ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
INFORMÁTICA
GRADO EN INGENIERÍA DEL SOFTWARE

MARCO DE INTEGRACIÓN DE SERVICIOS DE INFORMACIÓN EN EL ÁMBITO DE LA SALUD

FRAMEWORK FOR THE INTEGRATION OF INFORMATION SERVICES IN THE FIELD OF HEALTH

Realizado por
Francisco Miguel Soria López
Tutorizado por
Isaac Agudo Ruiz
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Febrero 2015

Fecha defensa:
El Secretario del Tribunal

Resumen:

Diraya es el sistema de información que utiliza el sistema sanitario andaluz como soporte de la información clínica de los pacientes. Integra toda la información clínica de cada ciudadano para su uso en cualquier momento y lugar.

La Subdirección de Tecnologías de Información del Servicio Andaluz de Salud ha definido un escenario de interoperabilidad basado en una estrategia SOA (Arquitectura Orientada a Servicios) permitiendo la integración de los distintos sistemas de información con Diraya.

Toda la información intercambiada por los distintos sistemas es guiada por un único bus de integraciones, el ESB (Enterprise Service Bus). El ESB debe publicar un servicio web por cada servicio del catálogo, haciendo uso del protocolo de comunicaciones SOAP.

Todo este intercambio de información entre sistemas se realiza a través del ESB corporativo y haciendo uso de mensajería HL7 (Health Level Seven), encapsulada dentro de los servicios web.

El objetivo de este TFG es el análisis, diseño, desarrollo y pruebas para el desarrollo de integraciones de sistemas de información con la historia digital de salud de Andalucía utilizando como motor de integración la herramienta Mirth Connect.

Como prueba de concepto se implementaran tres canales:

- Modificación de los datos de un paciente y grabación en base de datos.
- Creación de un fichero xml a partir de una información enviada en formato hl7.
- Creación de ficheros a partir de la información enviada a través de un servicio web.

Todo este diseño, desarrollo y pruebas se realizarán dentro de un entorno ficticio, simulando la puesta en práctica dentro del sistema hospitalario.

Palabras claves: Diraya, ESB, Mirht Connect, SOA, HL7, Servicio Web, SAS, SOAP, canal.

Abstract:

Diraya is the information system used by the Andalusian health system to support clinical patient information. It integrates clinical information of each citizen for use anytime and anywhere.

The Information Techonology Managment of the Andalusian Health Service has defined an interoperability scenario based on an SOA strategy (Service Oriented Architecture) enabling integration of different information systems. All information exchanged by the systems is guided by a single bus integration, the ESB (Enterprise Service Bus). The ESB must publish a web service for each service catalog, using the SOAP protocol communications.

All this information exchange between systems is via the corporate ESB and messaging using HL7 (Health Level Seven), encapsulated within web services.

The aim of this TFG is the analysis, design, development and testing for the development of information systems integrations with digital health histoy Andalucía integration engine using Mirth Connect tool.

As proof of concept three channels were implemented:

- Modifying the data of a patient and recording database.
- Creating an XML file from a format information sent HL7.
- Creating files from the information sent by a web service.

All this design, development and testing will be conducted within a fictional setting, simulating the implementation within the hospital system.

Keywords: Diraya, ESB, Mirth Connect, SOA, HL7, Servicio Web, SAS, SOAP, Canal.

ÍNDICE

Capítulo 1: Introducción

1.1 Introducción

La interoperabilidad entre diferentes sistemas de información, hoy día, es un aspecto muy importante en el ámbito de la informática y por extensión en el ámbito empresarial tanto del pequeño como gran comercio.

Dentro del entorno sanitario conlleva una gran importancia el proceso de interacción entre los diferentes sistemas de información debido al carácter de los datos tratados, siendo estos en muchas ocasiones datos críticos de los pacientes tratados.

En el ámbito sanitario coexisten múltiples sistemas de información, donde la información que reside en cada sistema es de vital importancia para la atención médica y para la gestión en todos los niveles de la organización. Dentro de esta estructura es frecuente que la información esté fragmentada en diversos sistemas de información independientes, formando estructuras independientes que determinan un acceso parcial a la información. Es por ello de la importancia que reside en estos sistemas independientes, donde reside la información y donde constituye un riesgo para los pacientes, pues las decisiones médicas se basan en la información parcial.

Se pueden encontrar diferentes definiciones del término interoperabilidad entre las que se puede destacar la ofrecida por el Institute of Medicine of the National Academies (IOM):

“Interoperabilidad es la habilidad de los sistemas para trabajar juntos, en general gracias a la adopción de estándares. La interoperabilidad no es solamente la habilidad de intercambiar información sanitaria, sino que requiere la habilidad de entender lo que se ha intercambiado”.

(Institute of Medicine, 2004)

“La habilidad de los sistemas de información computarizados y las aplicaciones software de comunicarse intercambiando datos en forma precisa, efectiva y consistente y de usar esa información intercambiada”. (HIMSS, 2005)

La oficina de interoperabilidad del servicio andaluz de salud la define como:

“la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos, de forma automática y manteniendo el significado en ambos extremos”

Y la integración como “la solución técnica para el intercambio de datos entre dos aplicaciones”

El presente proyecto está enfocado en la integración de sistemas de información dentro de un ámbito hospitalario, por ello, será necesario realizar un estudio de cómo se integran los distintos sistemas de información y en concreto en el caso estudiado

el sistema de información del servicio andaluz de salud. Para ello será necesario realizar un acercamiento DIRAYA, que es el sistema de información autonómico utilizado por el SAS. [1]

Diraya, es un sistema de información asistencial para el SAS, integrado por módulos relacionados y que comparten la misma información, que consiste en la integración en una Historia de Salud de cada ciudadano registrado por algún profesional del SAS. Gestiona las agendas de citaciones de consultas de primaria y consultas externas como de pruebas diagnósticas y permite el seguimiento de todos los tratamientos indicados y la extensión de la receta electrónica.

Independientemente de Diraya, que representa la columna vertebral del sistema sanitario andaluz, existen otras funcionalidades las cuales no están cubiertas por el mismo, lo que conlleva a que los centros sanitarios tengan que recurrir a otros sistemas de información adicional.

Todo esto conlleva a que la información en todos los sistemas deba ser coherente, de manera que un posible cambio en uno de los sistemas de información provoque la actualización en otro.

Para la integración de todos estos sistemas con Diraya, la Subdirección de Tecnologías de la Información del Servicio Andaluz de Salud ha definido un escenario de interoperabilidad basado en una estrategia SOA (Arquitectura Orientada a Servicios). [10]

Toda la información que es intercambiada entre los diferentes sistemas de información es guiada por el bus de integraciones ESB (Enterprise Service Bus) corporativo del Servicio Andaluz de Salud. Mediante este ESB se van produciendo continuamente los envíos y recepción de la información sujeta a alguna acción. El ESB publica un servicio web por cada servicio del catálogo, usando el protocolo de comunicaciones SOAP (Simple Object Access Protocol), y mediante un WSDL (Web Services Description Language), para que los sistemas proveedores de servicios puedan proveer la información.

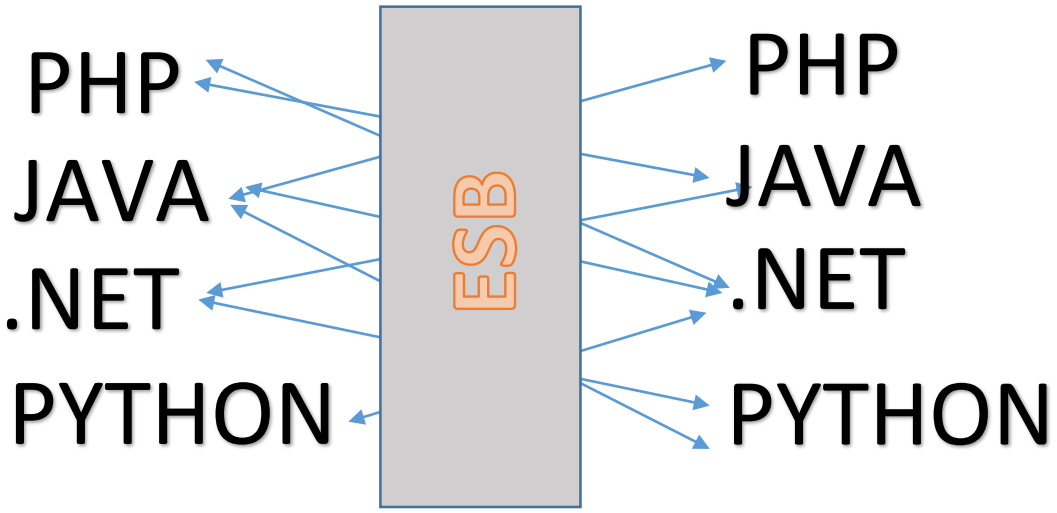


Ilustración 1. Posible escenario de interoperabilidad entre diferentes Sistemas de información.

El intercambio de información entre los sistemas se realiza mediante el ESB tal y como se ha comentado anteriormente y para el intercambio de la misma se hace uso de la mensajería HL7 (Health Level Seven), formato estándar dentro del ámbito sanitario y donde la información va encapsulada dentro de los propios servicios web.

El formato HL7 provee un conjunto de estándares para facilitar el intercambio electrónico de información clínica.

1.2 Necesidad

La importancia de la Integración:[1]

Evitar la fragmentación del conocimiento

Nuestro sistema de información posiblemente proporciona información útil para otros sistemas y supone una parte importante dentro del sistema global. El no disponer de esta información por parte de otros sistemas puede tener consecuencias muy graves.

Evita duplicidad de la información

Si se tiene un sistema que no se comunica con ninguno otro, entonces esta información se tendrá que replicar íntegramente en una base de datos propia.

Minimiza la entrada manual de datos

Al tener que duplicar la información, también se tendrá que introducir manualmente la misma, lo que conlleva grandes costes adicionales y puede provocar graves errores en su introducción**.**

[1] Extraído de informaticasana.com

Garantizar la calidad de los datos

Al introducir los datos manualmente nos exponemos a cometer errores en su entrada manual y no podremos reflejar los cambios directamente cuando se modifique en cualquier otro lugar de nuestra aplicación. Por lo tanto, sin una integración de sistemas será muy complicado tener la sincronización de la información.

Reducción de costes

Se podría pensar en soluciones más rápidas y económicas pero esto llevaría tarde o temprano a la aparición de nuevos problemas ya que al crecer estos sistemas, procesos y organizaciones estas soluciones serían muy complicadas de mantener.

Conforme el entorno hospitalario va incorporando nuevos sistemas de información, todos estos sistemas requieren la integración con el sistema de información global Diraya el cual a su vez hará de intermediario entre el nuevo sistema de información y los demás sistemas de información que se organizan en esta gran estructura del Servicio Andaluz de Salud.

Para ello los sistemas de información tendrán que implementar servicios web necesarios para consumir su mensajería HL7 (formato extendido dentro del mundo sanitario) desde su ESB corporativo.

La Oficina Técnica de Interoperabilidad de la Subdirección de Tecnologías de la Información del Servicio Andaluz de Salud proporciona toda la documentación técnica necesaria para consumir la información de cada uno de los servicios. Por lo tanto, es necesario que las empresas desarrolladoras de software sanitario que deseen que sus sistemas se instalen en los centros dependientes del Servicio Andaluz de Salud tengan conocimientos de HL7 y de servicios web para lograr con éxito la integración.

Para que las empresas se centren en su negocio, sin tener que preocuparse en aprender HL7, es deseable abstraer todo proceso de comunicación con el ESB. Esto permitiría utilizar mecanismos de comunicación alternativos y otros formatos para almacenar la información con los que el proveedor estuviera más familiarizado y que sus sistemas de información ya proveen. Mecanismos de comunicación pueden ser:

- Protocolos FTP o SMB para intercambio de ficheros.
- Actualización de bases de datos mediante procedimientos almacenados.
- Uso de GET Y POST para HTTP
- ….

Como ejemplos de formatos alternativos a HL7 pueden ser XML o JSON, adaptados a las necesidades de cada sistema de información.

1.3 Objetivo

El presente trabajo fin de grado está enfocado en la integración de un sistema de información particular de un hospital (el trabajo está centrado dentro de las especificaciones del Hospital Virgen de la Victorial de Málaga) dentro del sistema de información del SAS, Diraya, con la ayuda de la herramienta o motor de integración Mirth Connect.

Se planificará y desarrollará una serie de canales los cuales reproducirán de forma ficticia el funcionamiento dentro de un entorno hospitalario.

Se ha planteado el desarrollo de cuatro canales dentro de Mirth Connect. Una vez en funcionamiento, dichos canales realizarán el proceso de interoperabilidad entre los datos entrantes y los requerimientos exigidos por el sistema de información particular del hospital, dentro de un entorno simulado.

El desarrollo de los canales incluye:

- Inserción de los datos de un paciente en una base de datos postgresql a partir de los datos transferidos desde un canal Mirth.
- Creación de un fichero xml a partir de una información enviada en formato hl7.
- Creación de ficheros a partir de la información enviada a través de un servicio web.
- Envío de emails.

1.4 Contribuciones

Con este trabajo se pretende contribuir a la mejora de la calidad del servicio sanitario, a la integración de diferentes sistemas de información que puedan coexistir en este ámbito, a la actualización instantánea del historial clínico del paciente en todas las aplicaciones que hacen uso del mismo.

Con ello se contribuye a la mejora de las relaciones interdepartamentales dentro del ámbito sanitario y posibilita, en un futuro, una integración con los diferentes sistemas de información de otros hospitales y organismos autonómicos no pertenecientes al ámbito sanitario.

Dichos procesos automatizados implican el estudio del modelo actual de organización y cómo se puede producir una mejora de estos mismos gracias a las herramientas de integración.

Todo este análisis y desarrollo se realizará desde un entorno ficticio, posibilitando en un futuro la puesta en funcionamiento dentro del ámbito hospitalario.

1.5 Estructura de la memoria

La memoria de este Trabajo Fin de Grado ha sido planteada en 6 capítulos a través de los cuales se realizará un recorrido desde el análisis de la interoperabilidad entre el sistema información global y el sistema de información particular del Hospital Virgen de la Victoria de Málaga, todo esto dentro de un entorno simulado hasta el diseño de los canales encargados de realizar las operaciones especificadas en los objetivos. [13]

Una vez realizado el análisis del sistema a través del cual dicho hospital se integra con el sistema de información global del SAS Diraya, se procederá a la presentación de las tecnologías y estándares utilizados dentro del sistema.

Al concluir con el análisis del sistema se realizará una introducción al motor de integración Mirth Connect utilizado dentro del Centro hospitalario.

Después de la introducción a la herramienta Mirth Connect se procederá al desarrollo, prueba y funcionamiento de los canales planteados en el apartado de objetivos.

Para finalizar, realizaré una reflexión y ofreceré alguna de las conclusiones obtenidas del resultado planteado con sus posibles mejoras futuras.

Capítulo 2: Análisis del Sistema

En este capítulo se presentarán los conceptos más importantes sobre los que se sustenta el sistema de informático del SAS, la arquitectura desarrollada y la herramienta utilizada como integradora sobre la cual se realizarán todos los procesos de interoperabilidad entre sistemas.

2.1 Diraya

Diraya es el sistema informático que el Sistema Sanitario Público Andaluz utiliza como soporte de la información y gestión de la atención sanitaria. [2]

Los principales objetivos de Diraya son:

- Integrar la información de los usuarios independientemente de donde se produzca su generación en una Historia de salud única, que esté disponible donde y cuando se precise. Permitiendo así la creación de una Historia para su posterior consulta y anotación de datos en todos los dispositivos y niveles asistenciales: atención primaria, especializada, urgencias y hospitalización.
- Facilitar la accesibilidad a los servicios y prestaciones del sistema sanitario. Permite controlar el flujo de pacientes para coordinar todas las actuaciones requeridas en el diagnóstico y tratamiento de cada proceso.
- Lograr que toda la información relevante esté estructurada. Para ello utiliza tablas, códigos y catálogos comunes fruto del consenso profesional.

Diraya consta de un conjunto de módulos interconectados que intercambian información entre sí. Por ejemplo cuando un módulo de Diraya necesita identificar a un usuario, este se solicita a la Base de Datos Usuarios, si se precisa identificar un servicio hospitalario este se pide al módulo de estructura,..

Hay tres módulos principales sobre los que se sustenta Diraya:

El proyecto Diraya Atención Especializada (DAE), se encuentra enmarcado en el Programa de Sistemas de Información Asistencial del Servicio Andaluz de Salud y da cobertura a todos los dispositivos de asistencia, principalmente Atención Primaria y Especializada.[6], [7]

Su estructura se descompone en el siguiente gráfico:[2]

Ilustración 2.- Estructura de DIRAYA.

De la estructura gráfica anterior se pueden extraer los siguientes elementos principales: [1], [5]

1) Sistemas de Gestión de la Infraestructura de la Organización, Seguridad e Identificación del Ciudadano

- Módulo de Estructura.- Subsistema de Información Central que permite gestionar de manera corporativa tanto la estructura funcional (organización del sistema sanitario), la estructura física (recursos) y catálogos corporativos. Este módulo incluye los Servicios, Unidades Funcionales y ubicaciones físicas de atención primaria y especializada. Permite identificar cada servicio hospitalario, cada centro de atención primaria, cada dispositivo de urgencias,… es decir, la organización funcional de la asistencia. Permite identificar las ubicaciones físicas de los centros, establece la relación entre los niveles asistenciales y la realización de pruebas diagnósticas y gestiona los catálogos corporativos y las principales tablas maestras del sistema. En resumen, identifica los recursos y la oferta de servicios del sistema sanitario.
- MACO.- Subsistema de Información para la Gestión de los Operadores del Sistema, Perfiles (grupos de operadores) y permisos de acción a acceso a la funcionalidad del sistema, de modo que en el modelo de relación entre la estructura central y la estructura local Diraya Atención Hospitalaria (DAE) hace uso de MACO para gestionar los permisos funcionales locales de los Operadores del Sistema. Por decirlo de alguna manera MACO es la puerta de

[2] Modelo Corporativo de Implantaciones. MCI 2.0. Programa Diraya Atención Especializada Hospitalaria.

entrada a Diraya, cuando un profesional va a utilizar Diraya, este módulo identifica su clave de acceso y le permite utilizar las funciones de los diferentes módulos para las que está autorizado.

- BDU.- Subsistema de Información para la identificación de los Usuarios del Sistema Sanitario Andaluz a través de un número único (NUHSA) cuyo soporte físico es la Tarjeta Sanitaria (TAS). Por lo tanto, la función principal de BDU es dotar a cada ciudadano de un Número Único de Historia de Salud al que se vincula toda su información sanitaria.

2) Sistemas de Gestión Compartida de la Demanda

- Citaweb.- Gestor de la planificación de las consultas de Atención Primaria, Especializada, Radiología, Laboratorio,…
- AGD (Aplicación de la Gestión de la Demanda Quirúrgica).- Sistema de información puerta de entrada de toda la demanda quirúrgica.
- Receta XXI.- Sistema de Prescripción/Dispensación centralizado fuera del ámbito intrahospitalario.
- SIREGA (Sistema de información del Registro de Garantías).- Sistema que regula el cumplimiento de la garantía para la demora en las consultas y técnicas diagnósticas.
- Gestor de Peticiones.- Gestiona toda solicitud de pruebas diagnósticas y consultas que posteriormente se planificarán a través del sistema de gestión de citas.

Como se puede extraer de lo comentado anteriormente, se observa la complejidad de este gran sistema informático compuesto por diferentes módulos y como tienen que interoperar entre ellos para generar así un sistema en el cual todo usuario tiene su información perfectamente localizada y cualquier trabajador del servicio andaluz de salud puede acceder a los datos del mismo en cualquier lugar y momento. Para que todo este proceso de interoperabilidad se genere sin ningún tipo de problema a continuación se comentarán dos piezas claves para su funcionamiento como son el ESB (bus de integración de datos) por el que circulan todos los procesos de interoperabilidad del sistema y la estrategia de negocio a seguir, SOA, una arquitectura orientada al servicio. [4],[12]

Para encuadrar aún más la imagen que se tiene hasta el momento del sistema, tenemos que distinguir entre dos tipos de interoperabilidad que nos vamos a encontrar dentro de este sistema informático:

- La **interoperabilidad sintáctica**, hace referencia a los formatos y protocolos de los datos involucrados en los procesos de intercambio de información entre sistemas.
- La **interoperabilidad semántica**, busca que la información que se intercambia tenga la misma interpretación por los sistemas que intervienen, para ello

establece un modelo de referencia sobre el cual se unifican los conceptos terminológicos y de vocabulario.

2.2 ESB (Enterprise Service Bus)

Antes de introducir el concepto de ESB (Enterprise Service Bus) se dará una breve descripción de patrones de diseño de integración o mejor dicho métodos de trabajo en la forma de integración de aplicaciones.[3]

¿Por qué necesitamos la integración?

Porque las aplicaciones de negocio no pueden existir solas, en aislamiento. Antiguamente las aplicaciones no necesitaban de elementos externos, ni comunicaciones con el exterior, pero esa forma de trabajar ha cambiado y hoy día es casi imposible trabajar solo y exclusivamente dentro del mismo sistema informático (ilustración 3). Necesitamos que las aplicaciones se comuniquen entre sí debido a las necesidades de negocio, por lo tanto, es necesario disponer de una forma de diseñar aplicaciones estándar. Estas mejores prácticas se plasman en patrones de diseño.

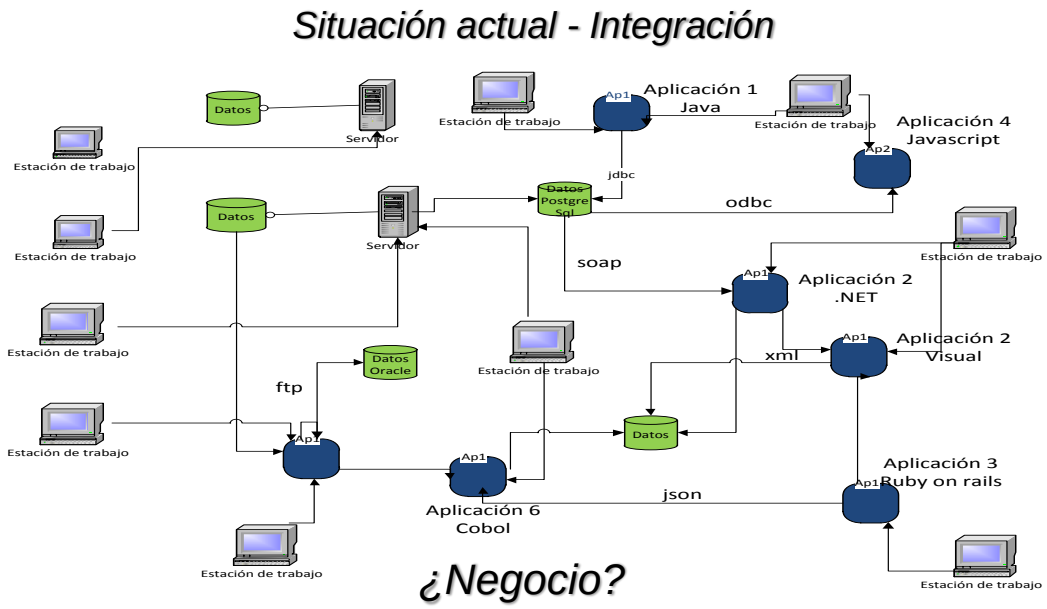


Ilustración 3.- Situación antes de la integración del sistema informático.

Estos patrones de diseño son métodos de trabajo ya probados para la solución de una determinada problemática. Estos patrones acumulan la experiencia de los desarrolladores que han pasado por la misma situación y por tanto no se inventan, más bien se descubren a partir de una problemática de negocio.

Si se habla de patrones de diseño de integración posiblemente el más importante sea el de Bus. El objetivo primordial del patrón de BUS es lograr el desacoplamiento entre

[3] http://pensandoensoa.com/

los dos extremos de la comunicación y para lograr esto se interpone otro elemento entre la comunicación de las dos partes, como si fuera un intermediario.

Por lo tanto, ¿Cuándo se necesita un Bus?

- Cuando se tienen numerosos puntos de integración.
- Necesidad de un futuro crecimiento de la arquitectura.
- Existencia de más de un protocolo de comunicación.
- Requerimientos de mediación entre dos extremos.
- Escalabilidad, gestión, monitoreo, transformación y seguridad.

Las ventajas de introducir una herramienta intermediaria entre las dos partes:

1) Desacoplamiento.- En un escenario de acoplamiento existe una red de conexiones punto a punto entre las mismas. Con el uso de un patrón de bus que desacople a las aplicaciones se tendrá un esquema como el siguiente:

Ilustración 4.- Interconexión entre Sistemas de Información y protocolos de comunicación distintos.

2) Adaptación de protocolo y formato.- Con el bus de integración que actúa como herramienta intermediaria entre el cliente del servicio y el proveedor, puede realizar el trabajo de adaptación de datos y protocolos. Todo este proceso de interacción se hará de una manera estándar y fácil.

3) Enrutador basado en contenido.- Dependiendo de los parámetros de entrada se puede invocar una lógica u otra, pudiendo transformar la mensajería dependiendo de estos mismos.

4) Configuración y no programación.- Los ESB se acompañan de herramientas para reducir la codificación. Gracias a su entorno visual se pueden realizar tareas o acciones mediante configuraciones sencillas y fáciles de usar.

5) Capa de abstracción de servicios.- Mediante esta herramienta intermedia el servicio final está oculto al usuario. De esta forma se podrá saber a qué servicios se está llamando, quién lo hace y con qué frecuencia.

6) Control de errores.- Con la capa ESB de integración es posible tener un punto centralizado donde gestionar los errores y proporcionar al cliente un interfaz único de estos errores.

7) Securización.- Se puede establecer un modo de securización común y homogéneo para los distintos backends que pueden a su vez tener una securización muy distinta.

8) Registro de servicios.- Se puede tener un registro de servicios (UDDI). Este registro contendrá el descriptor de despliegue del mismo (WSDL), el endpoint o punto de acceso del servicio (URL para invocar al servicio).

9) Validación, enriquecimiento.- El propio ESB proporciona API y funcionalidades de validación de los mensajes, pudiendo modificarlos.

Con todo esto se pretende obtener un sistema de Servicios hospitalarios tal y como se muestra en la ilustración 5, en la que se muestra un posible diagrama en el cual intervienen los servicios hospitalarios de una manera muy clara.

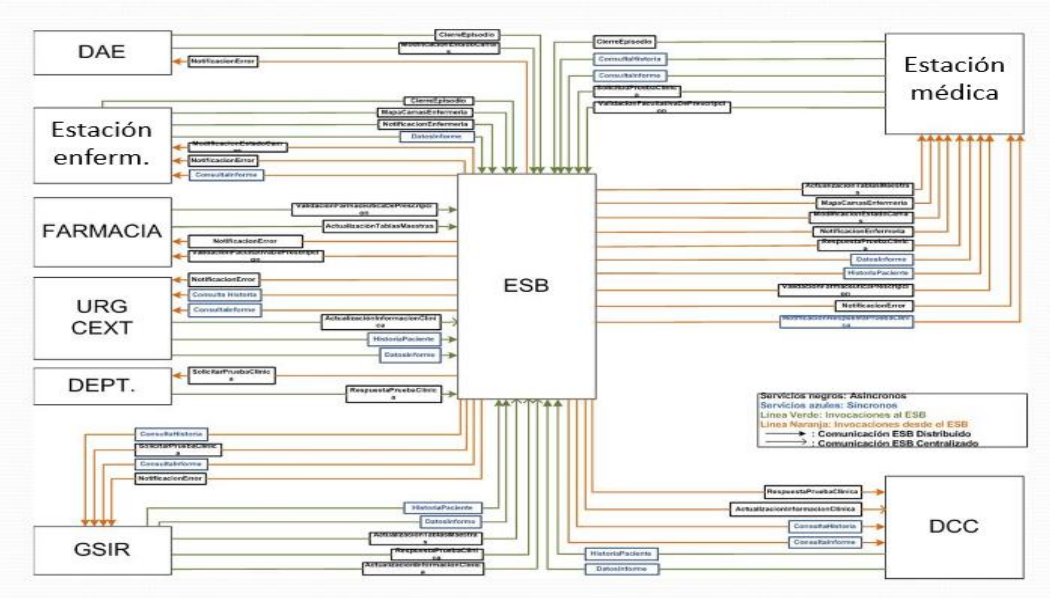


Ilustración 5.- Servicios Hospitalarios y como intermediario el ESB. Fuente SAS Subdirección Interoperabilidad.

En resumen, un Bus de servicios empresariales, es un software que actúa como intermediario, entre la comunicación de aplicaciones de diferentes sistemas. Siendo una herramienta donde se registran todos los servicios expuestos por todas las aplicaciones de un entorno empresarial y sobre el cual se puede construir aplicaciones para reaprovechar estas funcionalidades.

2.3 SOA

SOA representa un modelo que persigue mejorar la agilidad y los costes de una empresa reduciendo la carga de transferencia de información de la organización. Esto se logra ubicando los servicios como la forma de representar la lógica de negocio (la ilustración 6 muestra el esquema a seguir). [12]

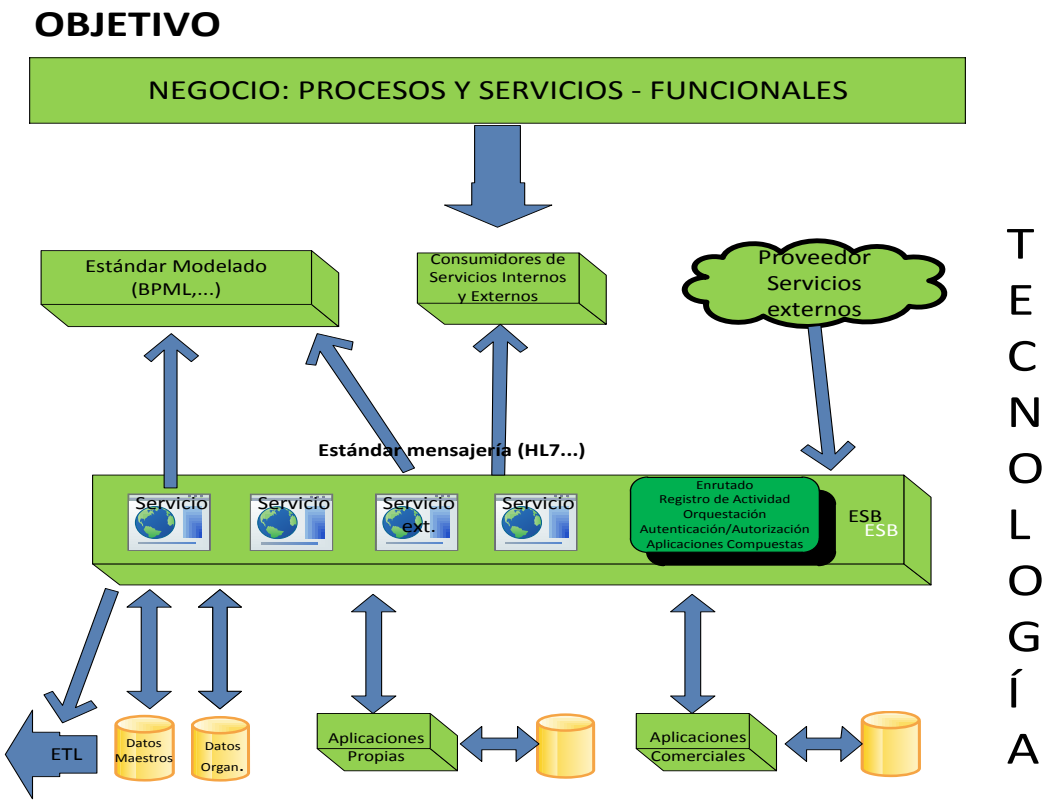


Ilustración 6.- Objetivo a alcanzar dentro de la subdirección de interoperabilidad del SAS. Fuente SAS Subdirección Interoperabilidad.

SOA, es un modelo de negocio, no tecnológico el cual desacopla negocio-tecnología, donde el conocimiento de la organización está en la organización, donde el conocimiento de los procesos y servicios evita multiplicidades, donde se garantiza la escalabilidad y la sostenibilidad y donde para poder llevarlo a la práctica se necesita:

- Modelar el negocio
- Definir gobierno del negocio y tecnológico
- Un modelo tecnológico basado en el uso de herramientas ESB -> integración, modelización y gobernanza.

En la ilustración 7, se puede observar un gráfico del esquema que se pretende alcanzar, mediante la arquitectura SOA. [3]

En dicha estructura se observa como el ESB representa la “columna vertebral” dentro de este modelo de negocio.

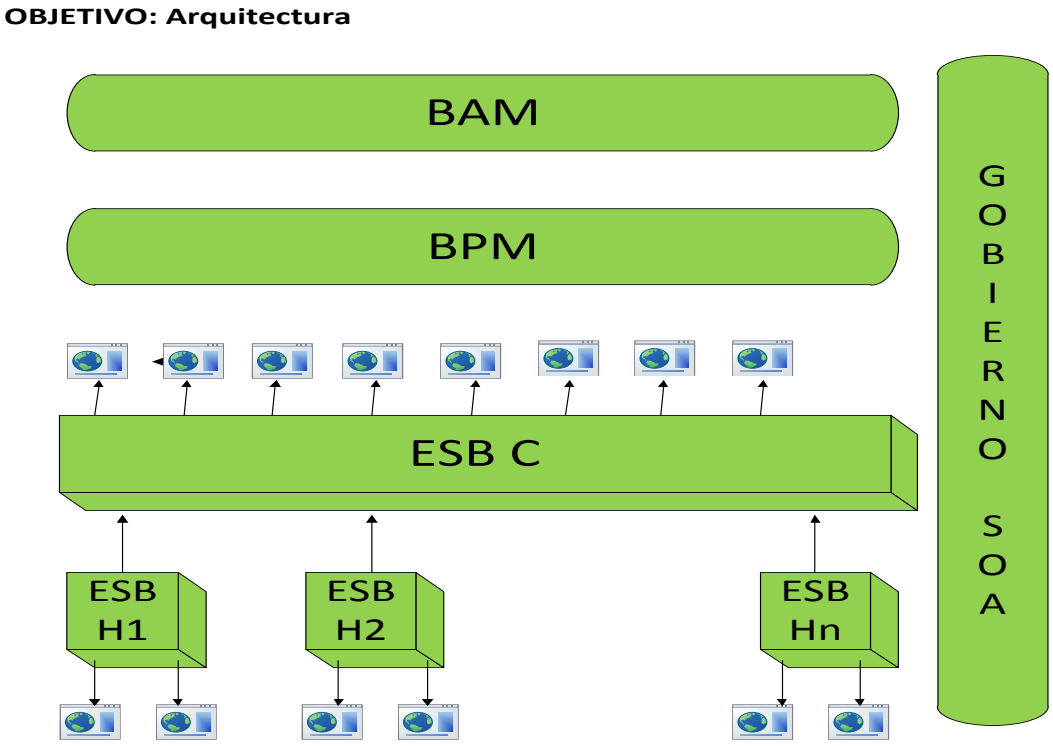


Ilustración 7.- Arquitectura deseada por la subdirección de interoperabilidad del SAS. Fuente SAS Subdirección Interoperabilidad.

De este gráfico se desprende, cómo a su vez cada subsistema de información de cada hospital puede a su vez hacer uso de otro ESB formando así una estructura en cadena en la cual como elemento principal se tiene un ESB general y a su vez cada subsistema de información hospitalario puede a su vez tener un ESB dentro de su arquitectura.

SOA es un modelo de arquitectura distribuida con características diferenciales en la realización de la orientación a servicios. Estas características son las siguientes:

Business Driven.- Estrategia dirigida por el negocio, a la arquitectura tecnológica con el fin de mantenerlas constantemente sincronizadas y evolucionar con el negocio a lo largo del tiempo. Es decir, que las arquitecturas tradicionales son creadas para soportar el estado actual de negocio pero sin pensar en la evolución del mismo. Pero que sucede cuando el negocio comienza a evolucionar y requerir nuevas exigencias, es cuando aparece la estrategia business-driven.

Vendor neutral.- La dependencia del vendedor de la plataforma reduce las oportunidades de innovación. Por lo tanto, si el modelo de arquitectura se ha diseñado para ser neutral respecto al vendedor, mantendrá la libertad de diversificar su implementación pudiendo aprovechar la innovación de múltiples vendedores.

Enterprise centric.- Fomenta la creación y reutilización de la lógica a través de diferentes lógicas de negocio y procesos. Para que un servicio sea reutilizable, debe ser capaz de participar en diferentes soluciones de arquitecturas distribuidas o composiciones de servicios.

Service composition architecture.- La característica de composición de un servicio permite que un mismo servicio pueda dar respuesta a nuevos requerimientos de negocio combinándolos de otra manera.

Finalmente se llega a la arquitectura mostrada en la ilustración 8, donde se representan los servicios que poseen cada uno de los Hospitales que a su vez se comunican con el ESB principal y este a su vez hace de interlocutor entre estos mismos haciendo uso para ello de los servicios ofrecidos por el sistema informático.

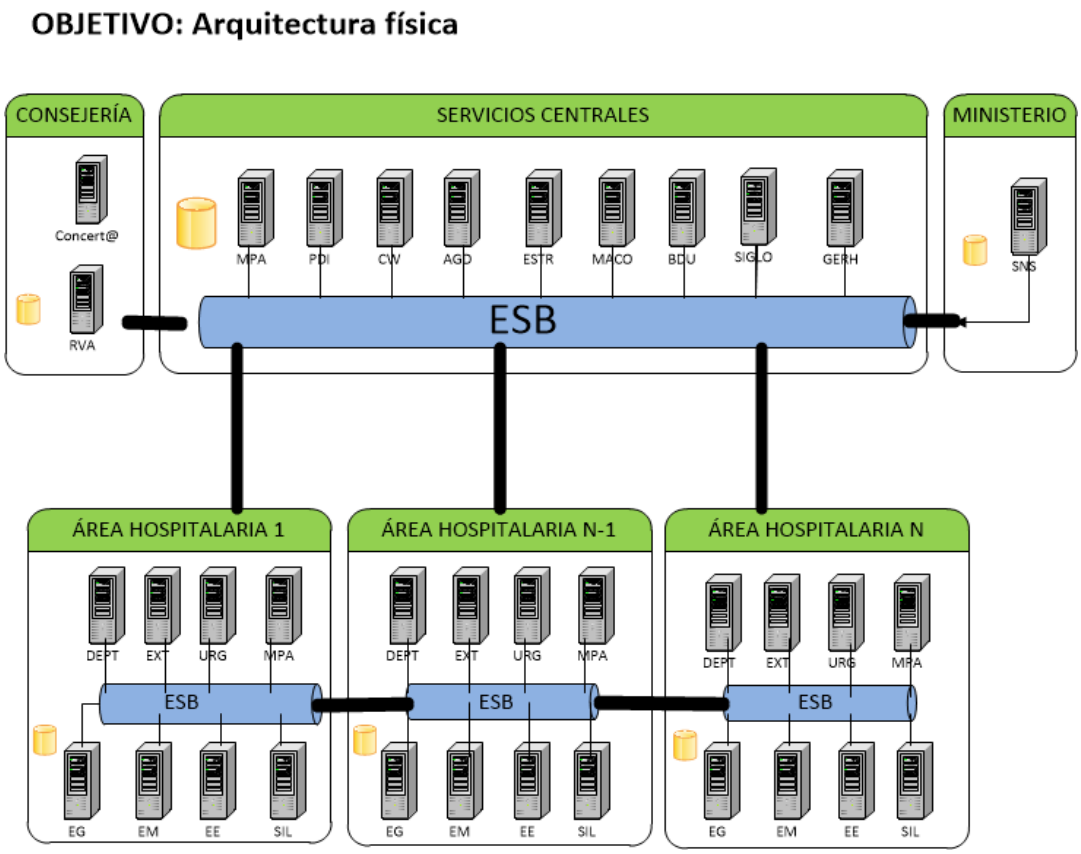


Ilustración 8.- Arquitectura final deseada por la subdirección de interoperabilidad del SAS. Fuente SAS Subdirección Interoperabilidad.

En la web de la subdirección de interoperabilidad del SAS se puede encontrar una descripción del catálogo de servicios de negocio SOA del SAS.[4]

En resumen, para el diseño y desarrollo de SOA la metodología de modelado y de diseño se tiene que enfocar hacia una arquitectura orientada a servicios. Esta arquitectura orientada a servicios es un marco de trabajo para el desarrollo de software y un marco de trabajo de implementación del mismo. Los desarrolladores de software deben orientarse hacia esta arquitectura creando servicios comunes para implementar los procesos de negocio.

[4] https://ws001.juntadeandalucia.es/unifica/interoperabilidad

Cuando se está hablando de arquitectura orientada a servicios se está hablando de una serie de servicios residentes en Internet, Intranet usando los mismos servicios web. El tipo de servicios web que podemos encontrar son xml, http, soap, rest, wsdl o uddi.

Con la descripción de la arquitectura SOA presentada en este último apartado se da por finalizado este capítulo en el cual se ha presentado el modelo de arquitectura utilizado en el sistema de información usado en el SAS.

Capítulo 3: Estándares y Tecnología

3.1 Introducción

En este capítulo se estudiarán y analizarán los principales estándares y tecnologías de uso dentro del sistema de información sanitario y en concreto en el caso del presente caso de estudio.

Se ofrecerá una visión de los formatos de ficheros más utilizados en el ámbito sanitario del SAS y de las implementaciones más utilizadas dentro de un entorno de interoperabilidad e integración.

Se comenzará con el estudio del formato HL7 (Health Level Seven), siendo este el estándar más usado dentro del ámbito sanitario.

A continuación se dará una visión del formato XML. Formato de uso extendido dentro de las aplicaciones utilizadas en el sistema de salud del SAS y el cual nos ofrece una alternativa al formato HL7 cuando la aplicación que hace uso de los datos de los pacientes no está planteada para el uso del formato HL7. Como se verá posteriormente el cambio del formato HL7 a formato XML se podrá realizar de forma automática por los canales una vez configurados. Este cambio de formato se realizará mediante funciones javascript.

Seguidamente se ofrecerá una visión de los servicios web y en concreto los servicios que utilizan como protocolo de comunicaciones SOAP (Simple Object Access Protocol).

Por último, se presentará una aproximación a la base de datos PostgreSQL, con la intención de su uso en capítulos posteriores.

3.2 HL7 (Health Level Seven)

HL7 (Health Level Seven) es un protocolo para el intercambio de información clínica, de uso generalizado dentro del ámbito sanitario y por supuesto dentro del caso de estudio del SAS. [11]

En la ilustración 9 (ver apéndice), se puede observar su ubicación en la capa de Aplicación dentro del modelo OSI.

Un mensaje HL7 es una cadena de texto que puede ser expresada en:

- Formato XML (v2.x XML)
- Nomenclatura de pipes (‘|’)

Siendo la nomenclatura de pipes la más utilizada dentro del ámbito sanitario, aunque ambas son equivalentes y se puede convertir de una a otra sin ningún tipo de problema.[5]

HL7 tiene estructura en función del tipo de mensaje y del evento disparador del mismo. Esta estructura se compone de segmentos los cuales constituyen una agrupación de campos. Un campo es una cadena de caracteres definida por un tipo de datos de HL7 y son los que finalmente contendrán la información específica de un determinado paciente.

Los segmentos dentro de un mensaje:

- Pueden ser requeridos u opcionales.
- Pueden ocurrir una vez o pueden repetirse.
- Son identificados con un código SEGMENT_ID.

Delimitadores estándar:

- |.- separador de campo.
- ^.- separador de componente.
- ~.- carácter de repetición.
- \.- carácter escape.
- &.- separador de subcomponente.
- <CR>.- terminador de segmento.

En el apéndice de la página web de HL7 se pueden encontrar todos los tipos de campos que se pueden incluir dentro de un mensaje. [6]

El nivel básico de definición dentro del estándar HL7 es el del mensaje abstracto asociado a cada evento particular. La definición del mensaje incluye:

- Datos.- Campos de datos a enviar dentro del mensaje.
- Respuestas.- Las respuestas válidas.
- Errores.- El tratamiento de errores de aplicación o fallos de comunicación.

Para saber interpretar un mensaje abstracto es necesario saber antes la terminología de símbolos utilizada dentro de los mismos:

- [ ].- significa 0 a 1, puede o no existir.
- { }.- significa 1 a N, puede haber 1 o más.
- [{ }].- significa 0 a N, puede haber 0 o más.
- …..- Debe existir solamente una vez en el mensaje.

En la ilustración 10 (ver apéndice) se puede observar la composición en segmentos de un mensaje correspondiente al alta de un paciente.

[5] Información protocolo de transporte hl7: http://www.interfaceware.com/blog/common- hl7-transports/

[6] Apéndice. Campos mensaje HL7. https://www.hl7.org/special/committees/vocab/V26_Appendix_A.pdf

El primer segmento, MSH, identifica el tipo de mensaje. Se puede observar el evento disparador que hizo que el mensaje sea enviado, en este caso, ADT^A28 (un alta de un paciente).

Tanto para el envío como para la recepción de mensajes se deben establecer unas reglas de codificación. Las reglas para el envío son:

- Codificar cada segmento en el orden especificado en el formato abstracto del mensaje.
- Poner el Id de segmento al comienzo del segmento.
- Cada campo debe ser precedido por el separador de campos.
- Codificar cada campo en el orden especificado por la tabla de definición de segmentos.
- Los campos que no están presentes no requieren ningún carácter.
- Los campos presentes con valor nulo deben codificarse “”
- Si los componentes, subcomponentes o repeticiones al final de un campo no están presentes, su separador puede ser omitido.
- Si no hay más campos al final de un segmento, los separadores restantes pueden ser omitidos.

Las reglas para recibir son:

- Si un segmento que se espera no se recibe finalmente, debe tratarse los campos involucrados como no presentes.
- Si se recibe un segmento que no se esperaba, ignorarlo.
- Si se reciben campos no esperados al final de un segmento, ignorarlos.

Los tipos de datos admitidos son variados:

- Alfanuméricos (ST, TX, FT) (string, texto, texto formateado)
    1. .Sp#.- fin de línea, saltar # líneas
    2. .br.- comenzar nueva línea
    3. .fi.- comenzar Word-wrap
    4. .nf.- sin Word-wrap
    5. .in#.- identar # espacios
    6. .ti#.- identar # espacios (temporalmente)
    7. .ce.- Terminar línea y centrar línea siguiente
    8. \H.- enfatizar
    9. \N.- Normal
- Numéricos (CQ, MO, NM, SI, SN)(cantidad compuesta, dinero, numérico, id de secuencia, numérico estructurado)
- Identificadores (ID, IS, HD, EI, RP, PL, PT) HL7 provee tablas para utilizar en los mensajes, tablas definidas por HL7 o tablas definidas por el usuario.
    1. ID.- valor para tablas definidas por HL7
    2. IS.- valor para tablas definidas por el usuario
    3. HD.- denominador jerárquico

4. EI.- identificador de entidad
5. RP.- puntero de referencia
6. PL.- ubicación de paciente
7. PT.- tipo de procesamiento

- Fecha/hora (DT, TM, TS)(fecha/hora, hora, time stamp)
- …

Las reglas de procesado de mensajes son:

Formas de procesamiento de mensajes:

- Original Mode Processing Rules
- Enhanced Mode Processing Rules

Reglas de proceso dentro del nivel OSI (Nivel 7 – Aplicación)

- El emisor construye un mensaje HL7 y lo envía al receptor
- El receptor recibe el mensaje y:
    1. Valida sintácticamente el mensaje de acuerdo a unas reglas de iniciación basadas en el segmento MSH. Si falla envía un mensaje de rechazo al emisor.
    2. Pasa el mensaje a la aplicación, la cual:
    - Crea un mensaje de respuesta
    - Crea un mensaje de error
    - Crea un mensaje de rechazo
    3. Envía el mensaje de respuesta, error o rechazo.

En el apéndice se incluyen algunos ejemplos de mensajería HL7.

3.3 XML

XML (Extensible Markup Language) (lenguaje de marcas extensible) es un lenguaje de etiquetado extensible de gran importancia para el intercambio de información. Desarrollado por el World Wide Web Consortium y utilizado para almacenar datos en forma legible.

XML proviene del lenguaje SGML permitiendo definir la gramática de lenguajes específicos para estructurar documentos grandes. XML da soporte a bases de datos, editores de texto, hojas de cálculo y cualquier otro uso. En el presente trabajo se hará uso de dicho lenguaje por su gran utilidad en los procesos de comunicación e integración de información entre aplicaciones.

XML fue propuesto desde sus inicios como un estándar para el intercambio de información estructurada entre diferentes plataformas y de ahí su importancia en este trabajo.

XML es similar a HTML aunque con la finalidad de describir datos en vez de mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas de los usuarios. Sirve para estructurar, almacenar e intercambiar información. XML hoy día tiene un papel muy importante en la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Las principales ventajas de XML son:

- Extensible.- Siendo posible la adición de nuevos elementos una vez diseñado y puesto en producción, facilitando así la continuación del diseño.
- Estructurado.- Permitiendo así su visualización y comprensión. Mejorando la compatibilidad entre aplicaciones. Se pueden comunicar aplicaciones de distintas plataformas. Cuando se dice que la información está estructurada quiere decir que la información se compone de partes bien definidas y que estas partes se componen a su vez de otras partes, estas partes son denominadas elementos y son indicadas mediante etiquetas.
- Validación.- La estructura de un fichero XML permite fácilmente su validación desde cualquier aplicación.
- Fácilmente procesable.- Su estructura es sencilla de entender y de procesar.
- Con XML se separa el contenido y el formato de presentación de los datos.
- Es un metalenguaje diseñado para cualquier lenguaje y alfabeto.

Componentes de un documento XML[7]

- Elementos.- Se representan con una cadena de texto (dato) encerrada entre etiquetas. Pudiendo existir elementos vacíos. Los elementos pueden contener atributos.
- Instrucciones.- Ordenes especiales para ser utilizadas por la aplicación que procesa.
- Instrucciones XML.- Comienzan por <? Y terminan por ¿>
- Comentarios.- Información que no forma parte del documento. Comienzan por <!—y terminan por -->.
- Declaraciones de tipo.- Especifican información acerca del documento. <!DOCTYPE >
- Secciones CDATA.- conjunto de caracteres que no deben ser interpretados por el procesador. <!CDATA[ ]>

[7] http://es.wikipedia.org/wiki/Extensible_Markup_Language#Historia

3.4 Servicios Web SOAP (Simple Object Access Protocol)

Servicio Web

Tal y como viene en multitud de definiciones sobre servicios web, un servicio web es una tecnología para el intercambio de mensajería entre aplicaciones y la cual utiliza un conjunto de protocolos y estándares para el intercambio de los mismos.[8]

En el uso de los servicios web se pueden utilizar diferentes lenguajes de programación y sistemas operativos. La interoperabilidad entre las diferentes aplicaciones se logra con la adopción de estándares abiertos. Para la mejora de la interoperabilidad entre los servicios web se ha creado el WS-I, el cual está encargado del desarrollo de perfiles para definir de una manera más exhaustiva estos estándares.

Los servicios web se pueden utilizar con HTTP sobre TCP en el puerto 80, esto significa que dichos servicios pueden atravesar los firewalls los cuales son los encargados de filtrar y bloquear parte del tráfico de Internet.[9]

Se puede decir que un Servicio Web es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Por lo tanto, distintas aplicaciones de software desarrolladas en diferentes lenguajes de programación, y sobre cualquier plataforma, pueden utilizar estos servicios para intercambiar datos en redes como puede ser Internet.

Un servicio web pone a disposición un conjunto de métodos que pueden ser invocados por alguna aplicación para la realización de alguna tarea.

SOAP (Simple object Access Protocol)

SOAP fue diseñado como un protocolo de acceso a objetos en 1998 y con la intención de solventar los problemas de interoperabilidad que tenían hasta el momento con los protocolos COM Y CORBA los cuales requerían que la comunicación se realizara entre máquinas que soportaran COM O CORBA, posteriormente con protocolos como DCOM o RMI donde la conexión entre máquinas dependía del lenguaje de programación. Por tanto, con esta problemática se plantea la necesidad de un protocolo de comunicaciones independiente del lenguaje de programación, de la máquina y del sistema operativo.[10]

Es un protocolo que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Paradigma de mensajería de una dirección sin estado y puede ser utilizado para formar protocolos más complejos y completos según las necesidades de las aplicaciones implementadoras.

[8] http://www.ibm.com/developerworks/ssa/webservices/tutorials/ws-understand-web-services1/

[9] Para más información consultar: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

[10] Información Extraída de: http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol

Basado en XML y estructurado en tres partes:

Envelope: donde se define el contenido del mensaje y la forma de procesarlo.

Conjunto de reglas de codificación: donde se expresan las instancias de tipos de datos.

Convención: para representar llamadas a procedimientos y respuestas.

Características principales de SOAP:

- Extensibilidad.- tanto en seguridad como en WS-routing los cuales son extensiones aplicadas en el desarrollo.
- Neutralidad.- pudiendo ser utilizado sobre cualquier protocolo de transporte (HTTP, SMTP, TCP O JMS).
- Independencia.- Se puede utilizar cualquier modelo de programación.

El uso que se suele dar a estos servicios web es el de integrar diferentes sistemas o componentes de una o varias plataformas.

La comunicación entre los sistemas de información y los servicios web se realiza a través del protocolo SOAP. SOAP es un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja.

SOAP es un protocolo para el intercambio de información en un entorno no centralizado y distribuido, normalmente usando HTTP y basado en XML. El framework ha sido diseñado para ser independiente de cualquier lenguaje. La importancia de SOAP reside en su simplicidad y extensibilidad.

El modelo de mensaje SOAP más utilizado es el RCP, donde un cliente envía un mensaje de solicitud a un servidor y el servidor responde el mensaje al cliente.

Los mensajes SOAP, son independientes del S.O y pueden transportarse en varios protocolos de Internet como pueden ser SMTP, MIME Y HTTP.

El mensaje SOAP está compuesto por una estructura en forma de “sobre” en la cual se incluye la cabecera del mensaje y el cuerpo, tal y como se puede apreciar en la ilustración 15 (ver apéndice).

Envelope.- Da la estructura, identifica al mensaje SOAP.

Header.- Permite enviar información relativa a cómo debe procesarse el mensaje. Herramienta para que los mensajes puedan enviarse de la forma más conveniente para las aplicaciones.

Este Header se compone de Header Blocks los cuales delimitan las unidades de información necesarias para el header.

Body.- Contiene información relativa a la llamada y respuesta.

Fault.- Contiene información relativa a errores producidos durante el procesado y envío del mensaje.

La manera más utilizada del protocolo SOAP es mediante el patrón petición-respuesta con el remitente y destinatario SOAP.

En cuanto a los lenguajes de programación que se pueden utilizar para la creación o consumo de servicios web se destacan Java, .NET, php, Python,… entre los cuales y con el uso de frameworks para el desarrollo de sistemas web agilizan el proceso de desarrollo de servicios web y de clientes que consuman a los mismos.

Para la descripción de los servicios web se utiliza WSDL, permitiendo que un servicio y un cliente establezcan la configuración en el transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL representa un contrato entre el solicitante y el suministrador, especificando la sintaxis y los mecanismos de intercambio de mensajería.

3.5 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido y con su código fuente disponible libremente. Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Entre las características a destacar están:[11]

- Estabilidad
- Potencia
- Robustez
- Facilidad de administración
- Implementación de estándares

Utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema, es decir, un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Ilustración 16 (ver apéndice)

Cliente.- Aplicación que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ser vía TCP/IP o sockets locales.

Demonio Postmaster.- Proceso principal de PostgreSQL. Encargado de escuchar por un puerto/socket las conexiones entrantes de clientes. Encargado de crear procesos hijos encargados de autentificar las peticiones, consultas y mandar los resultados a los clientes.

Ficheros de configuración.- Los tres ficheros principales de configuración utilizados son:

- Postgresql.conf

[11] Para más información consultar página: http://www.postgresql.org.es/

- Pg_hba.conf
- Pg_ident.conf

Procesos hijos Postgres.- Encargados de autentificar a los clientes, gestionar consultas y mandar los resultados a las aplicaciones clientes.

PostgreSQL share buffer cache.- Memoria compartida usada por PostgreSQL para almacenar datos en caché.

Write-Ahead Log (WAL).- Encargado de asegurar la integridad de los datos.

Kernel disk buffer cache.- Caché de disco del sistema operativo.

Disco.- Disco físico donde se almacenan los datos y la información para que PostgreSQL funcione.

Capítulo 4: Mirth Connect

4.1 Introducción a la herramienta Mirth Connect

Mirth Connect es un motor de integración enfocado para el ámbito sanitario y diseñado para la integración con mensajería HL7. Provee las herramientas necesarias para el desarrollo, testeo, despliegue y monitorización de interfaces de conexión realizadas. Al ser una herramienta de software libre dispone de una amplia comunidad de usuarios para su soporte.[8], [9]

Para el presente trabajo se ha optado por su versión 3 que es la más actualizada hasta la fecha.

Dicha versión está disponible tanto para entornos Windows como Linux y Mac OS X.

Para la puesta en funcionamiento Mirth Connect se requiere la máquina virtual de Java en su versión 1.6 o posterior. Aunque Mirth Connect dispone de una base de datos Apache Derby para el almacenamiento de su información, puede soportar otras como PostgreSQL, MySQL, Oracle 11gR2 o SQL Server.

En cuanto a las exigencias del hardware, todo dependerá de la actividad que tenga nuestra integración dentro del sistema. Por lo general, no son necesarios demasiados recursos para su funcionamiento.

¿Y qué tipos de sistemas y protocolos admite?

Mirth Connect permite desarrollar, configurar y desplegar interfaces de conexión sobre los protocolos más usados en el ámbito de la industria sanitaria, tales como:

- MLLP
- HTTP
- Database
- Email
- PDF
- JMS
- TCP/IP
- Web Services (SOAP)
- File System
- (S)FTP
- RTF
- DICOM

Ilustraciones 17 y 18 (ver apéndice)

En caso que Mirth Connect no pueda utilizar alguno de los protocolos anteriores, puede usar conectores estándar escritos en Java o JavaScript para transferir datos.

Mirth Connect permite crear interfaces arrastrando y soltando mediante scripts rápidamente.

Permite la creación de interfaces permitiendo el envío y recepción de mensajería creando mapeo y transformaciones arrastrando y soltando a través de la aplicación. Puede monitorizar y configurar las interfaces previamente creadas.

Una vez las interfaces han sido desplegadas, Mirth Connect provee una extensa variedad para monitorizar los canales desplegados. La ventana de despliegue de los canales ofrece una estadística muy variada de los estados de los mismos.

Un elemento clave sobre el que se soporta Mirth Connect es que es un software comercial libre, y porque es un software libre se pueden conseguir grandes beneficios en cuanto a la calidad y soporte de servicios sin el pago de licencia. Cada usuario puede unirse a la comunidad de Mirth Connect y obtener una gran cantidad de recursos incluyendo repositorios de código fuente y seguimiento de incidencias.

Mirth Connect puede transformar, filtrar y “enrutar” los mensajes, siendo estas características uno de los pilares sobre los que se sustenta su estructura. También se puede transformar entre los diferentes protocolos estándares.

Tal y como se ha comentado anteriormente Mirth Connect soporta numerosos formatos de datos utilizados en el ámbito sanitario como pueden ser:

- HL7 v2.x (comentado en capítulos anteriores)
- CDA (Arquitectura de Documento Clínico) pertenece al estándar HL7 y es el estándar más utilizado para el intercambio de documentos clínicos. Utilizando la tecnología XML, el modelo HL7 (RIM), la metodología de HL7 v3 y vocabularios controlados o locales. Se pueden enviar documentos con la mínima información contextual y documentos completamente codificados y referenciados.
- CCR (Continuidad del Registro del Cuidado) provee un formato estándar para la comunicación de información. Siguiendo la estructura:
    1. Identificación de Pacientes
    2. Historia Clínica
    3. Medicación
    4. Alergias
    5. Recomendaciones para el Plan de Cuidados
- DICOM (Imagen Digital y Comunicación en Medicina) estándar reconocido mundialmente para el intercambio de pruebas médicas, visualización, almacenamiento, impresión y transmisión.
- X12
- Delimited Text

- HL7 v3
- CCD (Continuidad del Documento de Cuidados) permite representar los datos del CCR en un CDA XML.
- XML
- NCPDP
- EDI
- Raw ASCII or Binary

Mirth Connect permite que cualquier tipo de datos sea procesado usando esta gran variedad de soporte de datos estándar.

Mirth Connect permite el almacenamiento de datos donde el usuario desee. Incluye una base de datos en la cual se pueden realizar las operaciones rápidamente. Una vez está en producción se puede almacenar la configuración y los datos de la mensajería en PostgreSQL, SQL Server, Oracle o MySQL.

Mirth Connect permite volver a procesar y gestionar los mensajes, es decir, cuando se tienen problemas con la interfaz o con los datos de los mensajes, Mirth Connect hace fácil este re-proceso y opcionalmente puede reemplazar mensajes desde el archivo.

Envía alertas y notificaciones
Mirth permite el envío de alertas email para detectar cualquier incidencia, lo que permite saber al administrador en todo momento cualquier problema con su interfaz.

4.2 Canales[12]

Un canal en Mirth Connect es un interfaz de conexión que permite realizar una acción determinada cuando se produce un evento que la pone en funcionamiento.

En la ilustración 19 (ver apéndice) podemos ver una muestra del Source (fuente) de un canal en Mirth. En este caso la fuente es un conector del tipo TCP Listener el cual estará “escuchando” en el puerto 6661, en el momento en que se produzca un evento, en dicho puerto se realizaran las acciones configuradas dentro de las etiquetas Filter y Transformer que darán paso a la ejecución en los canales de Destino.

Dentro de Mirth los canales están compuestos de una fuente y, al menos, un destino. Tanto la fuente y destino son conectores. Dichos conectores tienen como entrada un mensaje en un formato determinado y generan como salida el mensaje en otro formato tras aplicarle una serie de transformaciones. El formato de la entrada puede ser de otro sistema o como salida de otro conector.

[12] Información extraída de informáticasana.com

La salida del conector de la fuente está conectada a las entradas de los destinos y a su vez, el canal fuente puede responder al sistema que envió el mensaje usando la salida de alguno de los destinos configurados dentro del canal, de esta forma se cierra un circuito.

Un canal en Mirth Connect:

Tal y como se muestra en la ilustración 20 (ver apéndice) cada canal posee una configuración global para almacenar los mensajes encriptados en la base de datos y establecer el tipo o duración de almacenamiento de los mismos (flechas en rojo indicando las opciones más importantes a configurar dentro del canal).

Dentro del apartado Scripts se permite la ejecución de sentencias Javascript. Dichas sentencias pueden realizarse antes de que la fuente reciba el mensaje original (en este caso se selecciona la opción preprocessor) o bien justo después de que los destinos realicen su operación (postprocessor).

En la ilustración 21 (ver apéndice) se puede observar la ventana que se muestra para la ejecución de funciones Javascript dentro del canal Base de datos pdf y email.

En la ilustración 22 (ver apéndice) se puede observar una función javascript dentro del apartado Transformadores del conector fuente para el canal Base datos pdf y email. Dicha función javascript está implementada dentro de la variable Build observation list la cual incluirá dos variables a nivel del canal cuyos valores podrán ser tomados posteriormente desde los conectores destinos como variables del canal, tal y como se puede apreciar en la ilustración 23 (ver apéndice) marcadas con flecha en rojo.

Por lo tanto, el funcionamiento de los canales dentro de Mirth Connect depende bastante del tipo de los conectores que se esté usando, siendo esta variedad lo que hace de Mirth una herramienta tan versátil.

Con esto se da por finalizada una primera introducción a los canales en Mirth que posteriormente en los ejemplos que se realizarán en los capítulos posteriores se irán desarrollando más en profundidad.

4.3 Conectores

Mediante la configuración de los conectores tanto en fuente como en destino, se podrán realizar las operaciones requeridas para la interoperabilidad de sistemas.

En Mirth existen dos tipos de conectores principales:

- Conectores para las entradas fuentes.- donde a su vez estos conectores se pueden subdividir en:
    1. Reader.- este tipo de conector suele ejecutarse con cierta frecuencia (tal y como se puede apreciar en la ilustración 24 (ver apéndice) para la lectura de datos en una base de datos, lectura de ficheros, funciones Javascript, lectura de otro canal Mirth Connect,…

2. Listener.- este tipo de conector esta “escuchando” en un puerto continuamente y están basados en protocolos como:
   JMS (Java Message Service).- estándar de mensajería que permite a los componentes de aplicaciones basados en Java crear, enviar, recibir y leer mensajes.
   DICOM (Digital Imaging and Communication in Medicine).- estándar para el intercambio de pruebas médicas, visualización, almacenamiento, impresión y transmisión. Usa como protocolo de comunicación entre sistemas TCP/IP. Este tipo de ficheros solo pueden intercambiarse entre entidades que tengan la capacidad de envío y recepción de datos en formato DICOM.
   LLP (lower layer protocol).- protocolo de transporte más utilizado para el envío de mensajes HL7. A veces también llamado como MLLP, transmite mensajes HL7 vía TCP/IP.
   HTTP, TCP,…

- Conectores de Salida para los destinos.- estos conectores los podemos a su vez dividir en writer y sender.
  Conectores del tipo writer.- Permiten al conector escribir en una base de datos, fichero, otro canal del tipo Mirth Connect,…
  Conectores de tipo sender.- Permiten el envío de información utilizando protocolos como DICOM, LLP o TCP.

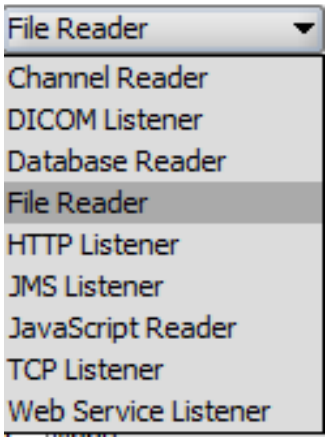
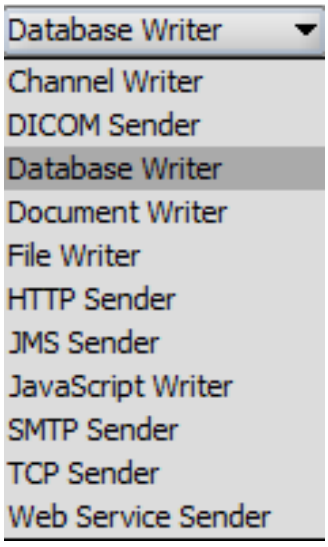
TIPOS DE CONECTORES FUENTE	TIPOS DE CONECTORES DESTINO
	

Ilustración 25. Tipo de conectores. Fuente y Destinos.

En la ilustración 25 se muestran los tipos de conectores fuente y destino que nos podemos encontrar en la nueva versión de Mirth Connect 3.0.3.

4.4 Panel de Administración (Dashboard)

El panel de administración o Dashboard nos permite conocer en todo momento el estado de los canales desplegados dentro de Mirth. Permitiendo conocer el estado de los canal o interfaces de conexión desplegados (arrancados, pausa, parados,..), su nombre, la revisión, la última vez que se desplegó, la mensajería recibida, filtrada, en cola, elementos enviados, errores y el estado de la conexión, estadísticas de uso de los canales, log del servidor y log de conexiones (ver ilustración 26 en el apéndice).

Esta información es importante para monitorizar el funcionamiento del sistema.

A su vez, para cada uno de los canales desplegados se puede acceder a otro panel donde podremos realizar operaciones sobre los mensajes recibidos, eventos del log que se han producido durante la transmisión/recepción de mensajería, errores, tipo de mensaje,…

En la ilustración 27 se puede observar el panel mostrado cuando se hace doble click en un canal dentro del Dashboard. En este caso, podemos monitorizar desde este panel el canal INFORMATICA_SANA_1_2_2_EXACTO que se ha creado y desplegado previamente.

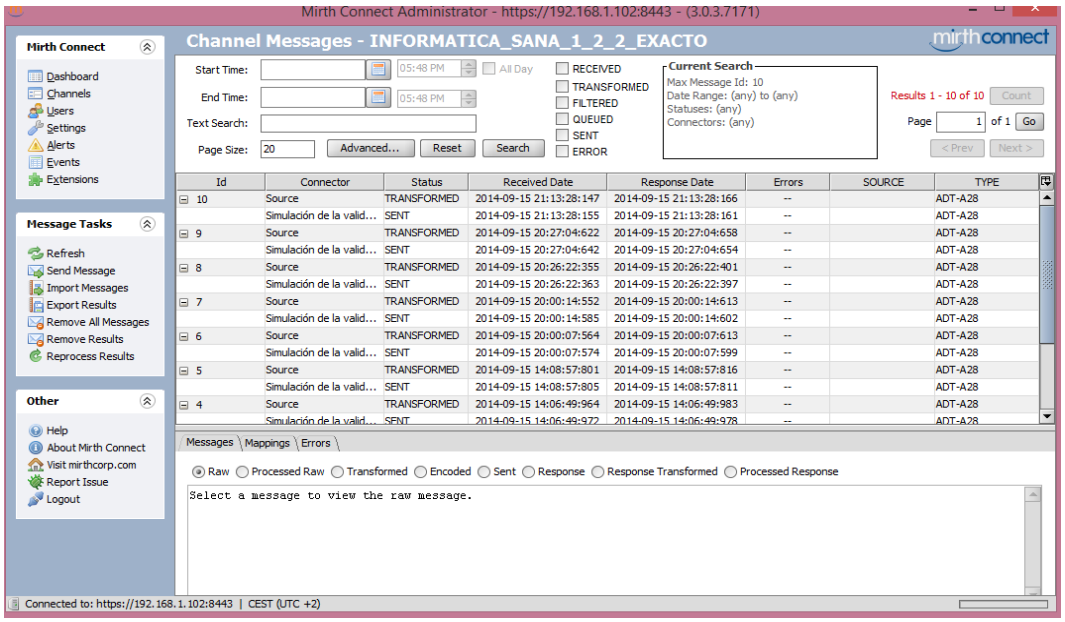


Ilustración 27. Muestra de la ventana que se visualiza dentro de un canal de Mirth.

Se concluye así una primera toma de contacto con la herramienta de integración Mirth Connect, en la cual se ha ofrecido una visión sobre el entorno y las configuraciones más específicas para desarrollar nuestra estructura de interoperabilidad. En el siguiente capítulo se procederá al diseño del sistema mediante la creación de una serie de canales los cuales simularán el funcionamiento e interacción entre diferentes sistemas de información.

Capítulo 5: Diseño del sistema

5.1 Introducción

En este capítulo se va a plantear el diseño y desarrollo de una serie de canales simulando su funcionamiento dentro del ámbito sanitario.

Para la creación del entorno ficticio se hará uso de dos máquinas virtuales una en Windows y otra en Ubuntu server recreando un entorno con entradas y salidas desde la herramienta Mirth Connect.

5.2 Máquinas Virtuales. Creación de máquinas virtuales en OracleVirtualBox

Oracle VM VirtualBox es una herramienta de virtualización para arquitecturas x86/amd64. Esta herramienta es desarrollada por Oracle como parte de su familia de productos de virtualización.[13]

Gracias a VirtualBox es posible la instalación de sistemas operativos adicionales, denominados sistemas invitados dentro de un sistema operativo anfitrión, cada uno con su propio entorno virtual, tal y como puede ser su entorno normal.

Los sistemas operativos soportados por VirtualBox son Linux, Mac OS X, OS/2, Windows y Solaris/OpenSolaris y dentro de los mismos se pueden virtualizar los sistemas operativos FreeBSD, GNU/Linux, OpenBSD, OS/2 Warp, Windows, Solaris, MS-DOS,…

Aunque inicialmente fue una herramienta bajo licencia, actualmente existe una versión gratuita para uso personal o de evaluación. La versión de software libre es la VirtualBox OSE.

En lo relacionado con el hardware y los requerimientos exigidos por las máquinas virtuales creadas, a través de VirtualBox se pueden configurar los discos duros de los sistemas invitados como discos duros dentro de los sistemas anfitriones como archivos individuales en contenedores llamados Virtual Disk image, incompatibles con los demás software de virtualización.

También se pueden montar imágenes ISO como unidades virtuales ópticas de CD o DVD.

Tiene un paquete de controladores que permite la aceleración en 3D, pantalla completa, hasta 4 placas PCI Ethernet, integración con teclado y ratón.

[13] Para más información consular: https://www.virtualbox.org/

A través de su ventana principal se pueden configurar y crear nuevas máquinas virtuales. En la ilustración 27 (ver apéndice) se puede observar la configuración que se ha establecido para la máquina Ubuntu Server 14.04 LTS.

Para el presente trabajo se han creado dos máquinas virtuales:

Máquina virtual Windows 7.- con un uso de memoria Ram de 512mb y con 8 Gb de disco duro, pudiendo ampliarse cuando sea necesario.

En dicho entorno se instalará la herramienta 7Edit, con la finalidad de recrear un entorno externo al de localhost dentro de la máquina anfitriona. En la ilustración 28 (ver apéndice) se puede observar dicha máquina virtual ejecutada dentro de la máquina anfitriona con el sistema operativo Windows 8.1 ilustración 29 (ver apéndice).

Dicha máquina invitada se comunicará con la máquina anfitriona a través de la dirección y puerto 192.168.1.102:21110, previamente configurados en los canales Mirth y a través de los cuales se recibirán y enviarán los mensajes HL7 y las respuestas a dichos mensajes.

Máquina virtual Ubuntu Server 14.04.

En esta máquina virtual se instalarán las herramientas siguientes:

- Máquina virtual de java jre en su última versión
- Mirth Connect 3.0.3
- Base de datos PostgreSQL 9.3.4

Dicha instalación y configuración se detallará en profundidad en el siguiente apartado.

5.3 Instalación de Ubuntu Server, Java jre y Mirth Connect

La instalación de Mirth Connect en su versión 3.0.3.7171 se realizará dentro de una máquina virtual Ubuntu server lts 14.04 lts, con 2gb de memoria ram y 8gb de disco duro como configuración genérica.

Instalación Ubuntu Server 14.04

Para la instalación de la máquina virtual Ubuntu server, una vez iniciado Oracle VM se pulsará en la opción Nueva, tal y como aparece en la ilustración 30 (ver apéndice).

Los siguientes pasos se configurarán dependiendo de los requerimientos que cada usuario desee establecer dependiendo del trabajo a realizar y de la configuración del equipo anfitrión.

Una vez se han configurado las características de la máquina virtual, se procede a la instalación de Ubuntu server dentro de dicha máquina, ver ilustraciones 31 y 32 en el apéndice.

Durante el proceso de instalación de Ubuntu server se van configurando los distintos elementos de configuración generales tales como: usuario, contraseña, configuración de red, idioma,…

Una vez instalado Ubuntu server dentro de la herramienta Oracle Virtual Box obtendremos una ventana similar a la que aparece en la ilustración 33 (ver apéndice).

Para que el acceso a Mirth Connect desde diferentes aplicaciones y base de datos PostgreSQL sea fijo, se configurará el archivo interfaces de Ubuntu server el cual se establecerá con una dirección ip estática, en mi caso 192.168.1.102.

La configuración a insertar dentro del archivo es la que aparece en la ilustración 34 (ver apéndice).

Como se puede observar en la configuración, las conexiones y accesos a esta máquina se realizarán de forma local al equipo anfitrión, aunque con direcciones ip distintas.

Instalación Java jre (máquina virtual de java)

Como nota importante a tener en cuenta:

Para el funcionamiento de Mirth Connect es necesaria la instalación de la máquina virtual de java, pare ello tendremos que instalar el jre de java en su versión 1.6 o más reciente (aunque con alguna versión anterior a 1.6 puede funcionar). En nuestro caso se instalará la versión 1.7. Para la instalación en Ubuntu server del jre se introducirá la siguiente instrucción:[14]

Sudo apt-get install default-jre

A continuación tendremos que configurar las variables de entorno para que Ubuntu haga uso de ellas en cualquier ubicación del servidor.

El path absoluto de java está en la ubicación /usr/lib/jvm, para identificar la localización de la instalación de java tendremos que introducir el siguiente comando (para saber dónde se encuentra el jre instalado dentro de nuestra máquina):

Ls –lah /usr/bin/java

Como resultado se obtendrá /etc/alternatives/java

Y a continuación introducimos el siguiente comando:

Ls –lah /etc/alternatives/java

Y finalmente obtenemos como resultado:

/usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java

Siendo esta ruta el path absoluto de la instalación de java.

Por lo tanto, para configurar el path de java_home para todos los usuarios tendremos que ingresar los siguientes comandos:

[14] Más información en http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

Declaramos la variable de entorno JAVA_HOME de la siguiente forma:

Export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java

No olvidemos poner la variable del PATH:

Export PATH=$PATH:/usr/lib/jvm/java-7-openjdk-amd64/jre/bin

Una vez finalizado estos pasos ya tenemos la máquina virtual de java instalada en nuestro servidor.

Instalación Mirth Connect

Una vez tenemos configuradas las variables de entorno, el siguiente paso es la instalación de Mirth Connect.

Para ello seguiremos una serie de pasos al igual que se ha realizado con la instalación del jre de java.

Introducir el siguiente comando en el terminal Ubuntu server:

W3m mirthcorp.com/community/downloads

Se busca la opción Linux el instalador denominado Installer, tal y como se muestra en la ilustración 35 (ver apéndice).

Una vez descargado el fichero con extensión .sh, introduciremos el siguiente comando para modificar los permisos del archivo permitiendo su ejecución:

Chmod 755 mirthconnect-3.0.3.7171.b1190-unix.sh

Y a continuación se ejecuta el comando:

./mirthconnect-3.0.3.7171.b1190-unix.sh

Para la instalación y configuración de la herramienta Mirth Connect.

Como puerto de acceso a Mirth Connect se deja por defecto el puerto 8443 para interacciones con Mirth y el puerto 8080 para arrancar Mirth y como usuario y contraseña se establece admin y admin respectivamente.

Para comprobar el funcionamiento de Mirth, desde la máquina anfitriona de las máquinas virtuales (Windows 8.1) se accede a la dirección:

http://192.168.1.102:8080

Finalmente aparecerá la ventana mostrada en la ilustración 36 (ver apéndice).

Al pulsar sobre la pestaña Launch Mirth Connect Administrator nos aparecerá la carga de un fichero webstart.jnlp el cual podremos ubicarlo posteriormente en el escritorio para acceder a la herramienta directamente, ilustración 37 (ver apéndice).

Una vez nos logueamos ya podremos ver la pantalla de administración de Mirth Connect tal y como se muestra en la ilustración 38 (ver apéndice).

Finalizada una sesión en Ubuntu server, se puede reiniciar el servidor Mirth Connect mediante el comando:

./mcserver start

./mcservice start ubicado en la carpeta /usr/local/mirthconnect

Y con ello iniciaremos el servidor para la utilización de la herramienta.

5.4 Instalación PostgreSQL en Ubuntu Server y PGAdmin III en Windows 8.1

Instalación de PostgreSQL en Ubuntu Server

Mediante la siguiente instrucción:

Sudo apt-get install postgresql postgresql-contrib

Instalaremos la base de datos PostgreSQL en nuestro servidor Ubuntu.

Para la creación de un nuevo usuario manejador de nuestra base de datos PostgreSQL lo realizaremos con el comando:

Su postgres –c “psql template1”

Y dentro del este nuevo “terminal” ejecutaremos la siguiente instrucción:

template1=# ALTER USER postgres WITH PASSWORD 'contraseña_deseada’;

Para la creación del usuario con su contraseña.

Para poder realizar conexiones desde equipos remotos habrá que configurar los ficheros postgresql.conf con la siguiente configuración:

Ejecutamos el comando:

Sudo nano /etc/postgresql/9.3/main/postgresql.conf

A continuación eliminar los comentarios # de las líneas que tienen incluidas las siguientes instrucciones:

listen_addresses='*'
password_encryption = on

Tal y como se puede apreciar en la ilustración 39 (ver apéndice).

También tendremos que configurar el fichero pg_hba.conf con las ip que accedan a la base de datos.

El fichero se ubica en la siguiente dirección dentro del servidor:

/etc/postgresql/9.3/main/postgresql.conf

Y se inserta la dirección de la máquina anfitriona tal y como se muestra en la ilustración 40 (ver apéndice).

Finalmente reiniciamos el servidor:

Service postgresql restart

Para la instalación del administrador de la base de datos PostgreSQL que se instalará en la máquina anfitriona accederemos a la página web de PostgreSQL:

http://www.pgadmin.org/download/windows.php

Y nos descargaremos su última versión 1.18.1 en mi caso. Los siguientes pasos se configuran sin ningún tipo de problema estableciendo un usuario y contraseña.

En la ilustración 41 (ver apéndice) se muestra la página de descarga de la base de datos y en la ilustración 42 (ver apéndice) se muestra la pantalla principal del administrador del PostgreSQL.

Una vez instalado el administrador pgAdmin III, se creará un nuevo servidor con la configuración que se puede observar en la ilustración 43 (ver apéndice) y que previamente se ha configurado en el servidor Ubuntu.

Llegados hasta este punto ya disponemos de las herramientas para poner a funcionar nuestros canales para realizar las operaciones de interoperabilidad entre distintos sistemas de información simulados.

5.5 Instalación del editor 7Edit en máquina virtual Windows 7

Para facilitar la labor en las pruebas de testeo de los canales que se van a crear en Mirth Connect se utilizará la herramienta 7Edit para la edición/creación y manipulación de mensajería HL7.[15]

De las múltiples herramientas encontradas en la red, se ha optado por esta debido a su facilidad de uso, y puesta en práctica dentro del centro hospitalario donde se ha realizado todo el estudio del presente trabajo. Aunque no es una herramienta de software libre, con la versión de prueba se puede hacer uso de la misma en un período cercano al mes.

Se ha hecho uso de una máquina virtual Windows 7 para la instalación de este software con la intención de simular el envío desde un entorno externo al de la máquina host y así realizar las pruebas en un entorno que no sea el local al del propio equipo.

7Edit es un software que nos permite la manipulación de mensajería HL7.

Que podemos hacer con 7Edit:

[15] Para más información consultar página http://www.7edit.com/home/index.php

- Vista de los mensajes HL7.- nos permite la creación y lectura de mensajes HL7 fácilmente, permite la identificación de segmentos y campos de forma simple pulsando sobre ellos.
- Edición de mensajes HL7.- edición de mensajes HL7 de forma simple. Toma los datos de forma estructurada, posicionados, con formato y los delimita automáticamente.
- Validación de mensajería HL7.- permite validar mensajes, dirigir perfiles, salvar reportes, deputar y redefinir mensajería fácilmente.
- Envío y Recepción de mensajería HL7.- permite simular y testear el intercambio de datos con sistemas HL7 mediante TCP/IP.
- Exportar HL7 a Excel y XML.- permite exportar mensajería HL7 a Excel, XML, y al formato HL7-XML 2.x.
- Personalización de definiciones y tablas.- permite personalizar definiciones existentes y tablas para adecuarlas a las necesidades del usuario, construcción de segmentos Z (en los casos en que HL7 no defina la información a intercambiar), exportación e importación entre la versión XML y la nomenclatura de pipes, comparación de mensajes,…

En la ilustración 44 (ver apéndice) se puede observar el entorno 7Edit, a través del cual el usuario podrá intercambiar mensajería HL7 entre sistemas del entorno hospitalario. Tal y como se aprecia en la ilustración tenemos una ventana con un navegador en forma de árbol, para explorar y editar mensajes HL7 según la estructura del mensaje. Dicha ventana está sincronizada con la ventana principal donde se encuentra el texto del mensaje HL7.x

Dentro de la ilustración 44 se observan 3 flechas indicando la ventana principal a través de la cual el usuario podrá configurar y realizar cualquier tipo de operación sobre el mensaje HL7.

La flecha 1 apunta a un mensaje HL7, que puede ser importado desde un fichero txt o bien puede ser creado desde su inicio de una forma rápida y sencilla. En este caso tenemos un fichero HL7 creado con anterioridad y se pueden observar los diferentes segmentos y campos de datos tal y como se ha comentado en capítulos anteriores.

La flecha 2 apunta al identificativo de cada campo dentro del segmento incluso si no está incluido en el mensaje HL7. De esta forma nos facilitará la labor al permitirnos ir introduciendo los datos conforme se van desplegando las pestañas incluidas en la ventana.

Por último la flecha 3 indica la ventana donde se configurará el nombre, la conexión y el tipo de acuerdo entre emisor/receptor aparte de otros parámetros de configuración general, en resumen, nos permite configurar perfiles de envío y recepción para conexiones TCP/IP, serial, MLLP,… siendo muy útil para la realización de pruebas cuando se está trabajando en un entorno de desarrollo.

En esta ventana también podemos ver otras pestañas como puede ser la configuración del receptor, es decir, el usuario de la máquina misma, una visión genérica del mensaje y un validador de mensajería.

La validación de mensajería es muy importante ya que 7Edit nos permite corregir de forma automática los mensajes HL7 en base a perfiles de validación configurables por los usuarios, siendo configurables a nivel de mensajes, segmentos, campos y definiciones. Muy útil al contrastar la mensajería real con la documentación proporcionada por el proveedor cuando se consume el mensaje HL7.

5.6 Instalación de SoapUI

SoapUI es una herramienta para ayudar en las pruebas y desarrollo de aplicaciones orientadas a los servicios web. Permite probar, simular y generar código de servicios web de forma ágil, partiendo del contrato de los mismos en formato WSDL y con vínculo SOAP sobre HTTP.[16]

Con una interfaz gráfica fácil de usar, permite fácilmente y rápidamente crear y ejecutar la funcionalidad de los servicios web de manera automática. Dentro de un entorno simple, SoapUI nos provee de una cobertura para realizar test y un gran soporte de protocolos y tecnologías.

En la ilustración 45 se puede observar la ventana que SoapUI nos ofrece para la creación de un proyecto tanto SOAP como REST, importación de proyectos, creación de entornos de trabajo,…

La versión usada para la realización de las pruebas es la 5.0.0. En la parte superior se introduce la dirección donde está ubicado el servicio web creado automáticamente por Mirth Connect.

Características principales de SoapUI:[17]

- Fácil uso.- el testeo de servicios web se realiza de forma muy simple, incluso para servicios web avanzados.
  Para realizar un testeo se comienza creando un proyecto y a continuación ir configurando las opciones del servicio a testear. También se puede insertar un WSDL, o crear una petición para todas los métodos/funciones incluidas en el servicio web, o incluso crear un mock (objeto simulado) y realizar su testeo. También se pueden testear operaciones sobre servicios web REST.
- Testeo automatizado, todo en uno.- es una completa y automatizada solución para el testeo de servicios. Las peticiones se realizan desde un entorno simple pudiendo testear los servicios web más estándares utilizados hoy día como

[16] Datos extraidos de SoapUI http://www.soapui.org/

pueden ser SOAP, REST, JMS,.. Todo esto se puede realizar de forma intuitiva desde una potente interfaz ofreciendo una configuración más personalizada si se prefiere dentro de sus opciones de configuración.

- Testeo para todo tipo de usuarios.- tanto para usuarios técnicos como para usuarios menos avanzados. Mediante un interfaz gráfico el trabajo con SoapUI se realiza de una forma simple.
- Ofrece funcionalidades avanzadas.- provee todas las herramientas que tú necesitas para testear servicios web, ofreciendo una visión global del proyecto y de sus contenidos. Usando HTTP Monitor para la grabación, análisis e incluso la modificación del tráfico entre cliente-servidor. Con Composite Projects se hace fácil el trabajo con proyectos en equipos y si la estructura de datos se va modificando SoapUI se va actualizando con ello.
- Permite la creación de reportes.- Reportes imprimibles, con datos exportables y reportes HTML.
  Se pueden imprimir los reportes en un formato estándar, incluyendo PDF, HTML, Word y Excel y personalizarlos según necesidades.
  En cuanto a la exportación de datos ofrece una impresión de reportes XML o CSV.

En la ilustración 46 (ver apéndice) se muestra un ejemplo de llamada al servicio web InformaticaSanaValidation el cual recibe como argumento un mensaje HL7. Realiza el envío del mensaje encapsulado en la estructura de SOAP al servicio web, este servicio web será el encargado de realizar la correspondiente acción con la ejecución del canal que hace uso del servicio web dentro de Mirth Connect.

5.7 Conexión desde Windows 8.1 a las máquinas virtuales

Para la conexión entre las dos máquinas virtuales y el pc anfitrión se configuraran las peticiones y conexiones mediante las direcciónes ip establecidas en cada una de las mismas.

Teniendo en cuenta que la máquina virtual Ubuntu Server tiene la dirección ip 192.168.1.102, la máquina virtual de Windows 7 y la máquina anfitriona podrán obtener otras direcciones libres.

5.8 El entorno de Mirth, ejemplos de uso

Una vez instaladas todas las herramientas de diseño y desarrollo de canales en Mirth Connect y configuradas previamente, se va a proceder a la creación de varios ejemplos a través de los cuales se probaran una serie de funcionalidades simuladas que se pueden producir tanto en un entorno de negocio como sanitario.[8]

5.8.1 Creación de ficheros

En este primer ejemplo voy a mostrar cómo se crea un canal básico en Mirth connect para la creación de un fichero XML a partir de uno en formato HL7.

Este canal será configurado de forma que esté en escucha en el puerto de entrada 21110 y posteriormente desde un conector de salida se guardará en una carpeta previamente configurada en el conector de salida.

Con la creación de este canal se simulará el envío desde un ESB de un mensaje en formato HL7 para que una aplicación que actualmente no puede procesar este tipo de mensajes, los pueda procesar en formato XML. Para ello se creará este canal con el objetivo de poder integrar esta aplicación "obsoleta" dentro del sistema de información actual y cambiando los ficheros de una ubicación inicial a una destino. De aquí la importancia de la herramienta Mirth Connect, tal y como se puede extraer de la explicación anterior el canal está actuando de elemento integrador entre diferentes sistemas de información, facilitando así la labor de interoperabilidad entre sistemas.

El canal propuesto se llama Escritura Fichero txt, en la ventana de configuración del tipo de datos se establecerá como entrada ficheros HL7 y para la salida del destino1 se indica extensión HL7 aunque se cambiará desde el destino a txt. En el destino2 se establece como entrada HL7 v2.x y para la salida se indica extensión como extensión XML, tal y como aparece en la ilustración 47 (ver apéndice).

En la ilustración 48 (ver apéndice) se puede observar la ventana Fuente del canal Escritura Fichero txt, se ha configurado como un conector tipo TCP Listener en el puerto 21110, puerto que se deberá configurar en la herramienta encargada del envío de mensajería, en mi caso, máquina virtual Windows 7 con la herramienta 7Edit.

En la ilustración 49 se muestra la ventana Destinations del canal Escritura Fichero txt. Tal y como se puede observar, consta de dos destinos conectores, El conector Destination1 recibirá un fichero en formato HL7 y lo ubicará con nombre pruebaGrabar.txt dentro de la carpeta /home del servidor en el cual está instalado Mirth Connect, en mi caso, máquina virtual Ubuntu server, tal y como se muestra en la ilustración 50 (ver apéndice).

El otro destino, descrito como Destinations2, al igual que el conector anterior recibe un fichero en formato HL7 y lo ubica en la dirección del servidor /home/prueba3 con el nombre ficheroxml${message.messageId}.xml, el parámetro $(message.messageId) contiene un valor dinámico que irá variando a medida que el destino va recibiendo mensajes, completándose así un rango de ficheros nombrados de forma incremental.

En el caso particular de este destino la entrada recibe un fichero HL7 y lo transforma en un fichero con formato XML, utilizando para ello una serie de transformadores para su modificación.

Este fichero XML, a su vez puede ser usado como entrada a una aplicación lectora de ficheros XML y que no pueda soportar la lectura de ficheros HL7 o bien podría ser la entrada a un servicio web para la realización de cualquier operación que a su vez pueda operar con otra aplicación que haga uso de ella.

Para la creación del fichero XML, tal y como se ha comentado anteriormente se han creado ocho transformadores, cada uno para cada campo que se introducirá en el fichero XML.

La ilustración 51 (ver apéndice) muestra los campos que configuran el mensaje XML, el número de historia clínica, nombre y apellidos del paciente, fecha de nacimiento, sexo, si ha fallecido y en su caso la fecha del fallecimiento. La plantilla está mostrada como un Outbound Message Template Tree.

La entrada de los campos anteriores se completará con los datos extraídos del mensaje de entrada HL7.

En la ilustración 52 (ver apéndice) se muestra la configuración del transformador encargado de obtener los apellidos del paciente. Para ello se obtendrá el primer apellido al cual se le concatenará el segundo apellido mediante funciones javascript. En la parte superior derecha de la ventana aparece el Inbound Message Template Tree el cual se tomará de plantilla para la recogida de la información del paciente.

Para la configuración del transformador fecha de nacimiento, cuya información de entrada viene estructurada en formato HL7 y se desea obtener un formato comprensible por el usuario, viene detallada en la ilustración 53 (ver apéndice). Para ello se hará uso de una función javascript global la cual se podrá utilizar en cualquier canal para convertir la fecha HL7 a una fecha con formato entendible por el usuario.

En la ilustración 54 (ver apéndice) se puede observar la configuración del script global. Dicha ilustración muestra la configuración del Script global encargado de la transformación de una fecha en formato HL7 que no es legible por el usuario a un formato legible por el mismo. El script está ubicado dentro de Code Templates dentro de la configuración general de Canales.

En este primer ejemplo se ha mostrado la creación de un canal para el envío de mensajería HL7 y su transformación/reenvío a otra carpeta dentro del servidor. En los siguientes ejemplos se mostrará la creación de una serie de canales Mirth los cuales proporcionarán diferentes funcionalidades.

5.8.2 Conexión con Servicio Web

En el siguiente ejemplo voy a desarrollar dos canales en Mirth para simular el funcionamiento a través de un servicio web del envío de ficheros desde una carpeta origen a una destinataria.

Se crearan dos canales, el primero expone el Servicio Web el cual recibe mensajería en XML, lo almacena en un directorio del servidor y el segundo canal leerá ficheros XML, los encapsula dentro de un mensaje SOAP y los envía al Servicio Web anterior.

Creación de un nuevo canal denominado Receptor – Web Service, este canal será el encargado de poner en funcionamiento el Servicio Web.

En la ilustración 55 (ver apéndice) se muestra la ventana principal de Mirth para la creación de un nuevo canal.

A continuación se introduce el nombre que queremos para este canal, en nuestro caso Receptor – Web Service. En la ilustración 56 (ver apéndice) se muestra la ventana para configurar los parámetros generales del canal, como pueden ser la duración de almacenamiento de los mensajes del canal, si el contenido tiene que ir encriptado, descripción de lo que hace el canal, duración en días del mensaje en la base de datos de Mirth,….

En mi caso, el único apartado para la configuración será la selección del tipo de los datos de entrada y salida para el conector fuente y destino.

En este ejemplo tanto entrada como salida recibirán y emitirán mensajes en formato XML, tal y como aparece en la ilustración 57 (ver apéndice).

Una vez que se ha configurado el tipo de datos a recibir y enviar, el siguiente paso será configurar el conector fuente del canal. Para ello nos situaremos en la ventana Source del canal y procederemos a su configuración. Tal y como he indicado anteriormente este conector pondrá en funcionamiento un Servicio Web el cual recibirá mensajería en formato XML y posteriormente lo escribe en una carpeta del servidor.

El tipo de conector para la fuente es Web Service Listener, es decir, un servicio web el cual atiende peticiones por el puesto 8081 tal y como aparece en la ilustración 58 (ver apéndice).

Una vez configurada la parte del tipo de conector y el puerto por donde recibirá las peticiones, pasamos a la creación del servicio web. Mirth Connect permite crear Servicios Web automáticamente, pudiendo así testear el canal en modo de desarrollo, para ello seleccionaremos por defecto Default service y posteriormente le daremos un nombre al Servicio Web, con este paso, automáticamente Mirth creará un servicio web con ese nombre el cual tendrá un método de para la recepción de mensajes denominado acceptMessage, el cual recibe un mensaje y emite otro nuevamente.

En el ejemplo el nombre que se le va a dar al Servicio Web es Mirth, se ubicará en la dirección http://192.168.1.102:8081/services/Mirth?wsdl y el método del servicio web al que tenemos que hacer referencia es String acceptMessage(String message), tal y como aparece en la configuración de la ilustración 58 (ver apéndice).

Independientemente de esta configuración, desde esta ventana podemos configurar un Servicio Web que tengamos previamente y utilizar sus métodos.

Una vez configurado el conector fuente del canal, pasaré a la configuración del canal destino. En la ilustración 59 (ver apéndice) aparece la configuración de conector de salida para el Canal Receptor – Web Service, tal y como aparece en dicha ilustración, se ha creado un destino denominado Destino 1, el cual creará un fichero denominado ${message.messageId}.xml y que se irá modificando automáticamente dependiendo de la petición id que realicemos, es decir, se irá cambiando de nombre conforme vayamos realizando peticiones.

El fichero se creará en formato XML en el directorio prueba2 del servidor. Por último, para introducir el contenido en el fichero XML, se arrastrará desde la sub-ventana Destinations Mappings el elemento Encoded Data el cual se refiere al contenido del mensaje a la sub-ventana Template la cual finalmente incluirá este contenido en el fichero.

Para concluir con la configuración de este canal, configuraremos la pestaña de Transformer, donde solamente incluiremos en el Inbound y en el Outbound el formato de los ficheros, simplemente seleccionamos un fichero XML para que Mirth interprete el formato XML internamente (esto es importante hacerlo para evitar errores en la generación de la mensajería). En la ilustración 60 (ver apéndice) se muestra el ejemplo de la puesta en escena del fichero XML tanto en la pestaña Inbound (entrada) como en el Outbound (salida).

Hasta aquí la configuración del canal creador del Servicio Web, para verificar si existe algún error de tipo script o de configuración general pulsaremos sobre la pestaña validate conector y lo validaremos. Una vez validado y se puede desplegar el canal para comprobar su funcionamiento desde la pestaña Channels.

Para comprobar el funcionamiento del canal, podremos testearlo introduciendo la dirección del Servicio Web en un navegador web y comprobar que aparece el servicio especificado, tal y como aparece en la ilustración 61 (ver apéndice).

Para el testear el envío y recepción de mensajes se utilizará la herramienta SoaupUi que se ha especificada previamente en este capítulo, se crea un nuevo proyecto SOAP y se introduce la url donde está ubicado el Servicio Web, a continuación se envía un mensaje de prueba con algunos datos de ejemplo tal y como aparece en la ilustración 62 (ver apéndice).

En la ilustración 63 se muestra un ejemplo del fichero creado por Mirth una vez que esta desplegado.

Una vez testeado el canal y comprobado su funcionamiento se procede a la creación del segundo canal que hará las funciones de recepción de mensajería y envío al Servicio Web.

Este nuevo canal se denominará Emisor Web Service y el tipo de datos de entrada al canal a través del conector fuente y envío de datos a través del conector destino serán de tipo XML, tal y como se muestra en la ilustración 64 (ver apéndice).

Una vez configurados el tipo de datos de entrada y salida, nos situamos en la pestaña fuente (Source) del canal para proceder a su configuración.

Este conector se configura del tipo Lectura de Fichero (File Reader) en la carpeta del servidor prueba1, se procesarán todos los ficheros con extensión XML y una vez finalizada la lectura del fichero o ficheros se eliminan automáticamente del servidor. La ilustración 65 (ver apéndice) muestra la configuración del conector fuente para este canal.

Una vez finalizada la configuración del conector fuente, se procederá a configuración del conector destino. Este conector se denomina Destination 1 y es del tipo Web Service Sender, es decir, es un conector el cual enviará información a un servicio web que posteriormente se introducirá y el servicio web ya realizará su proceso particular.

Dentro del apartado Web Service Sender Settings se introduciran todos la información necesaria del Servicio Web para que Mirth la localice. En concreto se introduce la dirección donde se ubica el servicio web, en este caso, http://192.168.1.102:8081/services/Mirth?wsdl. Pulsando en la pestaña Get Operations, Mirth automáticamente rellena los campos de Servicio y puerto.

Seguidamente se selecciona dentro del apartado Operation el método que deseamos que reciba la información dentro del Servicio Web. Y por último generamos el Envelope (sobre) pulsando sobre Generate Envelope el cual generará el mensaje en formato SOAP. Dentro del Envelope tendremos que introducir el mensaje leido previamente en el conector fuente.

La ilustración 66 (ver apéndice) muestra la configuración del canal destino para su uso con el Servicio Web Mirth, previamente creado en el canal anterior.

Al igual que se ha realizado en el canal anterior, para evitar posibles errores se editará la ventana de los Transformadores y se pondrá en escena el Inbound y Outbound seleccionando para cada uno un mensaje en formato XML.

Una vez desplegados los dos canales, podemos comprobar el funcionamiento de cada uno desde el Dashboard. Ilustración 67.

Para comprobar que realmente ha funcionado todo este proceso de interoperabilidad entre canales, iremos a la carpeta prueba1 del servidor Ubuntu e incluiremos un archivo en formato XML como el que se muestra en la ilustración 68 (ver apéndice).

A continuación y una vez que están desplegados los canales en Mirth Connect, iremos a la carpeta destinataria a través de la cual el Servicio Web realizará su proceso de recogida de información y creación de fichero por parte del conector destino.

En la ilustración 69 (ver apéndice) se puede observar la carpeta prueba3 del Servidor. También se observan una serie de ficheros con extensión XML, los cuales representan llamadas al Servicio Web y la posterior creación de ficheros en el formato indicado.

Finalmente en la ilustración 70 (ver apéndice) muestro un posible ejemplo del contenido del fichero 22.xml (que tal y como he comentado anteriormente se ha generado automáticamente indicándole por nombre el número de la llamada al Servicio Web). Tal y como se puede apreciar en la ventana mostrada, el contenido es el mismos que el que teníamos en la carpeta origen.

Hasta aquí el diseño y desarrollo de este ejemplo el cual ha mostrado como con el uso de un Servicio Web (simulado por un canal Mirth) se pueden traspasar ficheros de un sistema a otro sin necesidad de realizar ningún tipo de actividad. Todo esto con la configuración de los canales dentro de Mirth para finalmente conseguir el objetivo de traspase de ficheros entre sistemas.

5.8.3 Actualización de Base de Datos PostgreSQL

En este ejemplo se simulará la inserción en la base de datos PostgreSQL de los datos de un paciente, tales como el nombre y apellido. Una vez que Mirth detecta la aparición de un fichero en formato hl7 lo que realizará será la obtención de los datos del paciente y su posterior introducción en la base de datos PostgreSQL.

Creamos un nuevo canal llamado Grabar en BD Postgres tal y como se puede apreciar en la ilustración 71 (ver apéndice).

A continuación dentro de la casilla Name se introduce el nombre identificador del canal. Ilustración 72 (ver apéndice).

En esta configuración los tipos de datos se dejaran tal y como Mirth Connect los establece por defecto, que es en formato HL7, ya que tanto en la entrada como en la salida los datos son HL7. En la ilustración 73 (ver apéndice) se puede observar la configuración predeterminada por Mirth Connect.

A continuación nos ubicamos dentro de la pestaña Source para la configuración del conector fuente de los datos de entrada. Ilustración 74.

En esta ocasión, el canal estará “en escucha” (cada 5000 ms) dentro del directorio home del servidor Ubuntu y cuando aparezca un fichero con extensión HL7, realizará la acción especificada en el conector destino.

Notar que una vez se produce la lectura del fichero HL7 Mirth Connect se encargará de eliminar el fichero HL7, ya que se ha configurado de esta manera.

En caso de error, Mirth Connect detectará algún error en la lectura y automáticamente creará un fichero de error dentro de la carpeta /home/prueba2 dentro del servidor Ubuntu con la información relativa al error.

En cuanto a la configuración del conector de destino se establecerán los parámetros indicados en las ilustraciones 75 y 76 (ver apéndice).

Tal y como se puede apreciar en la ilustración anterior el canal se configurará de la siguiente forma:

Nombre: Destination 1.

Tipo de conector: Database Writer (Base de datos grabador).

Driver: PostgreSQL (Se selecciona la base de datos a la cual se accederá)

URL: jdbc:postgresql://localhost:5432/postgres , la dirección en la cual se ubica la base de datos PostgreSQL, previamente configurada. Se introduce localhost ya que tanto Mirth Connect como la base de datos PostgreSQL están ubicados dentro del mismo servidor. El puerto es el que se instala por defecto 5432 cuando se instala PostgreSQL y la base de datos es la creada para la realización de esta simulación, denominada postgres con la tabla “pacientes”.

Username, Password: como nombre de usuario y contraseña se introducen los valores que se han establecido al instalar la base de datos.

SQL: para la generación de la instrucción SQL se puede hacer uso del botón insert el cual nos facilitará la creación del insert con la generación automática de la instrucción:

INSERT INTO pacientes (nombre, apellidos) VALUES (${nombre}, ${apellidos})

Tal y como se puede apreciar en la ilustración 75 (ver apéndice).

Notar que dentro de la sentencia SQL VALUES se incluyen los parámetros ${nombre} y ${apellidos} los cuales se recogerán automáticamente de las variables Channel Map recogidas dentro de Transformer, tal y como se aprecia en la ilustración 74 (ver apéndice).

En la ilustración 76 (ver apéndice), se pueden observar tres variables a las que se podrá acceder dentro del canal al configurarlas de tipo Mapper y al añadirlas al Channel Map.

Como muestra de ejemplo, la variable identificada como nombre, para acceder al valor del mensaje se incluye la siguiente instrucción javascript:

msg['PID']['PID.5']['PID.5.2'].toString()

Los campos incluidos como PID son extraidos de la pestaña Message Trees en un mensaje ejemplo dentro del segmento PID y campo PID 5, identificador de los datos personales del paciente tal y como se ha descrito en capítulos anteriores. Ilustración 77.

Una vez finalizado con la configuración del canal se procede a su despliege en el Dashboard.

Tal y como se puede observar en la ilustración 78 (ver apéndice) se han recibido tres entradas al canal y se han producido otros tres envíos.

Las tres entradas corresponden a tres ficheros con extensión HL7 incluidos en el home del servidor y Mirth Connect automáticamente los elimina y los inserta en la base de datos PostgreSQL.

La ilustración 78 (ver apéndice) muestra el despliegue del canal Grabar en BD Postgres y se muestra la recepción y envío de tres ficheros HL7.

En la ilustración 79 (ver apéndice) se muestran tres registros de pacientes, los cuales han sido insertados por la inclusión de tres ficheros con extensión HL7 dentro de la carpeta home del servidor Ubuntu.

5.8.4 Envío de emails

En el siguiente ejemplo de creación de canales se creara un canal para el envío de emails en caso de modificación de la información de información relativa a un paciente que previamente se ha recibido desde un servicio web o bien a través de un correo que pasa directamente a través de Mirth Connect.

Tenemos un escenario de interoperabilidad donde un servicio web hace de puente entre la recepción de mensajería del ESB y su posterior envío a correo previamente configurado en Mirth.

Tal y como se ha visto en anteriores capítulos la mensajería circula por el ESB y esta herramienta será la encargada de encaminar el mensaje desde el servicio web en cuestión hacia el puerto en el que Mirth estará “escuchando”.

Para ello se crea un canal denominado MirthSoap tal y como se muestra en la ilustración 80 (ver apéndice). La configuración de esta entrada se dejará por defecto tal y como la configura Mirth.

En la ilustración 81 (ver apéndice), se muestra la configuración del conector fuente. En dicha configuración se establece como tipo para el conector un Servicio Web del tipo Listener, en el puerto 8081. El nombre del Servicio Web es Mirth y la descripción de dicho Servicio se obtiene en la dirección http://192.168.1.102:8081/services/Mirth?wsdl.

El Servicio Web tiene un método el cual acepta como entrada una cadena de texto y devuelve esa misma cadena, la configuración de este método se realizada de la siguiente forma:

String acceptMessage(String message)

La función de este Servicio Web es conectora entre fuente y destino, es decir, recibe un mensaje a través del ESB, el canal que está a la escucha también detecta que el Servicio Web ha recibido un mensaje y en ese momento entra en acción el conector destino el cual será encargado de realizar la operación especificada.

Hacer notar que desde la pestaña Edit Transformer se puede modificar el texto recibido y añadirle alguna información procedente de otro canal o bien simplemente añadirle información relativa al paciente en cuestión desde una base de datos.

A continuación se procederá a la descripción del conector destino. Dicho conector es denominado Destination 1 y es del tipo SMTP Sender, es decir, conector para el envío

de correo. En este caso se ha procedido al uso de Gmail para el envío de la mensajería y por ello se establece como el Host de SMTP: smtp.gmail.com, dicho servicio de Gmail se configura en su puesto 465 para las conexiones SSL y se realiza una espera de 5sg para su envío. Ilustración 82.

En las opciones de nombre de usuario y contraseña, se establecerá la información en particular de la cuenta del administrador o bien de la cuenta particular del usuario que configure este canal, en mi caso he puesto mi cuenta en Gmail.

Seguidamente se introducen los correos desde donde hasta donde se desea recibir la mensajería en caso que se envíen correos entre sí, o bien hacia donde se enviará el mensaje recibido desde el servicio web, en mi caso, el mensaje se enviará al correo de Hotmail.

En definitiva, se utiliza la cuenta de correo Gmail como transportador y como receptor de correo la cuenta de Hotmail.

Una vez desplegado el canal, ilustración 83 (ver apéndice), vemos cómo se van realizando los envíos y recepciones de correo a través del canal MirthSoap, el cual tal y como se ha comentado anteriormente va recibiendo mensajes del servicio web puesto en funcionamiento.

Una vez desplegado el canal y realizando un envío desde la herramienta SoapUI, tal y como se muestra en la ilustración 84 (ver apéndice). Se simula el envío desde el ESB central que interopera en todo el sistema. En dicha ilustración se muestra el mensaje enviado a Mirth. El mensaje está en formato SOAP tal y como se puede apreciar, donde el campo arg0 contiene la información que posteriormente se volverá a enviar al correo destinatario final.

Una vez enviado el mensaje SOAP, el canal MirthSoap se encargará de reenviar la información especificada en el arg0 al correo destinatario especificado anteriormente.

En la ilustración 85 (ver apéndice) se observa el mensaje enviado al correo destinatario de Hotmail.

5.8.5 Interacción entre ruby on rails, 7 Edit y base de datos PostgreSQL

En este ejemplo se va a realizar una interacción entre rails y 7Edit a través de la base de datos PostgreSQL. El objetivo del ejemplo es dar a conocer como a través de un canal en Mirth Connect y previamente configurado podemos hacer de la interoperabilidad entre diferentes sistemas de información un proceso eficaz e integrado.

Se ha creado una simple aplicación para el uso de la base de datos PostgreSQL en ruby on rails (no es pretensión del trabajo la descripción del funcionamiento de ruby on rails ni el desarrollo de aplicaciones en el mismo) como una aplicación externa a otro sistema de información y con el objetivo de simular un entorno de trabajo diferente.

Tenemos funcionando la herramienta 7Edit en la máquina virtual de Windows 7, que simula su funcionamiento desde otro sistema o simplemente desde el ESB central por el que llega la mensajería.

Por otro lado, la base de datos PostgreSQL está funcionando dentro del servidor Ubuntu server, situado en un entorno distinto al de la aplicación ruby y al de la mensajería HL7.

Funcionamiento del entorno simulado

En la ilustración 86 (ver apéndice) se puede observar la aplicación que hará uso de la base de datos PostgreSQL.

Es una aplicación sencilla de desarrollar para el ingreso de nombres en la tabla posts dentro de la base de datos postgres.

Se puede simular un entorno en el cual el administrativo del sistema hospitalario está introduciendo datos en una base de datos a través de una aplicación situada en el sistema de información privado del hospital y ubicándose la base de datos en un entorno diferente.

La ilustración 87 (ver apéndice) muestra un cuadro donde aparecen los mismos datos que se pueden obtener en la aplicación ruby on rails.

En la ilustración 88 (ver apéndice) se muestra el entorno donde se está ejecutando la herramienta 7Edit a través de la cual se van a enviar la mensajería HL7 a la base de datos PostgreSQL, teniendo muy en cuenta que como medio de integración/interoperabilidad estará Mirth Connect mediante la creación de un canal tal y como se puede observar en las ilustraciones 89, 90 y 91 (ver apéndice).

Se ha creado un nuevo canal denominado llp 2 restful RAILS, el cual en su ventana Source se establecerá como tipo conector para la fuente TCP Listener en el puerto 21110.

Se ha creado un destino denominado Destination 1, el cual será del tipo Database Writer con la finalidad de ingresar el nombre del usuario/paciente que recibe de la mensajería enviada y cuya configuración aparece en la ilustración 89. Se puede observar en dicha figura la sentencia SQL de inserción en la base de datos con una variable con ámbito dentro del canal y que previamente se ha configurado en la pestaña de Transformer, tal y como se puede apreciar en la ilustración 90 (ver apéndice).

En la ilustración 91 (ver apéndice) se puede observar la sentencia javascript para obtener el nombre del usuario/paciente que se puede obtener desde los mensajes HL7 enviados al puerto 21110 del canal llp 2 restful RAILS.

Con la creación y puesta en funcionamiento de esta serie de canales se puede dar por finalizado este apartado en el cual he desarrollado una serie de canales con la finalidad de poder mostrar la importancia de una herramienta de integración en

cualquier tipo de entorno, en el caso del presente trabajo ha sido enfocado al ámbito sanitario pero este tipo de arquitectura/estilo de desarrollo puede extrapolarse a cualquier ámbito de negocio.

Con la simulación y diseño de este último canal se da por finalizado este capítulo y el groso del presente trabajo en el cual se ha ofrecido una visión de la importancia de la interoperabilidad entre distintos sistemas y como a través de diferentes herramientas este proceso se puede realizar de forma automática y oculta al usuario final.

Para ello se han ido presentando los diferentes conceptos relativos a la estructura del escenario en el cual se desarrollan algunas de las operaciones del Servicio Andaluz de Salud.

Capítulo 6: Conclusiones y Posibles Mejoras

6.1 Conclusiones

Con el presente trabajo fin de grado se ha mostrado la importancia hoy día de la interoperabilidad e integración de la diversidad de sistemas de información existentes dentro del entorno sanitario y por su extensión dentro del entorno empresarial.

Tal y como se ha mostrado en los capítulos iniciales del trabajo, inicialmente existía una red de sistemas heterogéneos y aislados, donde la información quedaba en los propios sistemas. Surge la necesidad de intercambiar toda esta información (información muy valiosa para el personal sanitario) y por ello comienzan a integrarse estos sistemas. A través de la implantación del modelo de interoperabilidad orientado a servicios SOA, se ha logrado una integración de sistemas que ha contribuido a una mayor eficiencia, reducción en costes y errores humanos, reducción en los tiempos de trabajo, mejor aprovechamiento de los recursos y agilidad de procedimientos dentro del SAS que ha contribuido a mejorar la asistencia sanitaria en cuanto a calidad seguridad y eficiencia.

Para ello se ha ofrecido desde el inicio una visión global del sistema de información utilizado por el SAS y se ha ido profundizando a lo largo del trabajo hacia un entorno ficticio particular dentro un hospital.

Una vez planteado el entorno de trabajo existente y la importancia de la información sanitaria se ha presentado una herramienta integradora capaz de conectar todos estos sistemas de información creando un gran sistema de información global donde todas las aplicaciones están interconectadas. En el caso del presente trabajo se ha utilizado Mirth Connect (herramienta utilizada en el Hospital Virgen de la Victoria) y se ha mostrado su capacidad de integración con los diferentes subsistemas de información que pueden existir en un hospital.

Mediante la integración e interoperabilidad entre los diferentes sistemas de información en el ámbito hospitalario, se pretende minimizar el impacto que supone el cambio hacia nuevas aplicaciones, la interacción entre aplicaciones de diversos departamentos del hospital y la posibilidad de cometer errores humanos en la utilización manual de la información del paciente. Con todo ello el personal médico puede tener accesible en todo momento la información clínica actualizada de cada paciente.

La integración entre sistemas de información hace que el personal clínico tenga una visión global de la información y todo esto constituye un valor fundamental y de gran utilidad para poder realizar el proceso asistencial del paciente y la toma de decisiones.

Tal y como se ha comentado anteriormente este trabajo se ha estado enfocado al ámbito sanitario aunque toda esta arquitectura de trabajo puede ser utilizada en

cualquier ámbito de trabajo empresarial que por su extensión y diversidad de estructura en cuanto a nivel de software lo permita, haciendo de ello un entorno más eficiente y útil dentro del mismo.

En resumen, con esta arquitectura de trabajo se pretende dar un giro a la integración, interoperabilidad y desarrollo de sistemas ofreciendo una gran evolución respecto a arquitecturas estáticas encajadas en un ámbito local y sin posibilidad de integración con otros sistemas.

6.2 Posibles mejoras

Como posible mejora y puesta en práctica de los canales desarrollados se podrían incrementar sus funcionalidades añadiéndoles nuevas características para la integración de diferentes sistemas de información y/o aplicaciones que pueden coexistir dentro de un complejo hospitalario.

Dicha mejora conllevaría el estudio en profundidad de las necesidades interdepartamentales de cada subsistema de información implantado en el centro sanitario y de los requerimientos de información de cada uno de los servicios clínicos del hospital.

Para ello se seleccionaría una sección clínica para un estudio de integración. Esto conllevaría un estudio de los procesos que existen actualmente interactuando dentro de la sección, las necesidades que se desean cubrir, las nuevas aplicaciones existentes y los formatos utilizados por las mismas.

Sería necesario realizar reuniones con los/as coordinadores/as de la sección para extraer las necesidades del departamento. Hay que tener en cuenta que todo este procedimiento integrador debe ir supervisado por la unidad de interoperabilidad de cada centro sanitario y por el coordinador y supervisor del departamento en estudio.

Una vez seleccionada el área clínica de estudio se podría proceder al estudio de todas las aplicaciones utilizadas dentro del departamento e ir identificando cada comunicación interna y externa de dicho departamento. Una vez identificados los procesos comunicadores, se procedería a la creación de los correspondientes canales dentro de la herramienta integradora utilizada por el centro.

En el caso que nos ocupa, el centro hospitalario Virgen de la Victoria utiliza a nivel interno como herramienta integradora, Mirth Connect. En otros centros sanitarios se utilizan las herramientas integradoras Iguana y Ensemble.

Con este desarrollo y/o modificación de canales se contribuye a la mejora de la calidad sanitaria, mejora de la cooperación entre profesionales de diferentes disciplinas, cooperación entre médicos y pacientes, coordinación entre áreas diversas y por último colaboración entre organizaciones públicas y privadas.

Bibliografía

Documentación aportada por el SAS:

[1]
Diraya Atención Especializada inforsalud'2013 V1
[2]
DossierDiraya2010_Es.
Dossier sobre el Sistema de Información Diraya.
[3]
Technologies in the Andalusian Healthcare ministry
Presentación sobre el Sistema Sanitario Andaluz.
[4]
Presentación 20110720_SOA-IOP (Subdirección de Tecnología e Información)
[5]
Interoperabilidad, Integrados: Interoperabilidad, Gobernanza y Casos Prácticos. Apisa 2014, Junio

Webgrafía

[6]
http://www.juntadeandalucia.es/repositorio/usuario/listado/fichacompleta.jsf?idProyecto=626
[7]
http://www.juntadeandalucia.es/servicioandaluzdesalud/principal/documentosacc.asp?pagina=pr_diraya
[8]
http://www.informaticasana.com/
[9]
http://www.mirthcorp.com/products/mirth-connect
[10]
http://www.seis.es/documentos/informes/secciones/adjunto1/15_Interoperabilidad.pdf
[11]
http://www.hl7spain.org/documents/tutoriales_HL7/SemHL7_Detalles_V2.pdf
[12]
http://pensandoensoa.com/
[13]
https://ws001.juntadeandalucia.es/unifica/interoperabilidad

Apéndice.

Ilustraciones.

Ilustraciones capítulo 3

Ilustración 9.hl7spain.org.

Ejemplos de mensajería HL7.
Extraídas de www.hl7spain.org

Ejemplo de mensaje ADT^A01 (Admisión de un paciente)

MSH|^~\&|EPICADT|DH|LABADT|DH|201301011226||ADT^A01|HL7MSG00001|P|2.7|
EVN|A01|201301011223||
PID|1||MRN12345^5^M11|52299881^^^^DNI|TORRES^ANTONIOIII||19710101|M||C|1 CATALYZE STREET^^MADISON^WI^53005-1020|GL|(414)379-1212|(414)271-3434||S||MRN12345001^2^M10|123456789|987654^NC|
NK1|1|TOLEDANO^ANTONIA|WIFE|||||NK^NEXT OF KIN
PV1|1|I|2000^2012^01|||004777^GOOD^SIDNEY^J.|||SUR||||ADM|A0|

Ilustración 10. Ejemplo de mensaje para la admisión de un paciente.

Ejemplo de mensaje ORM^O01 (Del tipo Orden)

MSH|^~\&|*HIS*|MedCenter|*LIS*|MedCenter|20060307110114||ORM^O01|MSGID20060307110114|P|2.3
PID|||12001||Torres^Francisco||19670824|M|||30 c/Cordoba Málaga^CO^80020^ESPAÑA|||||||
PV1||O|OP^PAREG^||||2342^Torres^Benito|||OP|||||||||2|||||||||||||||||||||||||20060307110111|
ORC|NW|20060307110114
OBR|1|20060307110114||003038^Urinalysis^L|||20060307110114

Ilustración 11. Ejemplo de mensaje del tipo orden.

Ejemplo de mensaje ORU^R01 (Imágenes)

MSH|^~\&|LCS|LCA|LIS|TEST9999|199807311532||ORU^R01|3629|P|2.2
PID|2|2161348462|20809880170|1614614|20809880170^TESTPAT||19760924|M|||^^^^
00000-0000||||||86427531^^^03|SSN# HERE
ORC|NW|8642753100012^LIS|20809880170^LCS|||||19980727000000|||7280^MEDICO^FEDERICO
OBR|1|8642753100012^LIS|20809880170^LCS|008342^COLUMNA
LUMBOSACRA^L|||19980727175800|||||SS#634748641 CH14885 SRC:THROA
SRC:PENI|19980727000000||||||20809880170||19980730041800||BN|F
OBX|1|ST|008342^IMPRESIÓN DIAGNÓSTICA^L||FINALREPORT|||||N|F||| 19980729160500|BN
ORC|NW|8642753100012^LIS|20809880170^LCS|||||19980727000000|||HAVILAND
OBR|2|8642753100012^LIS|20809880170^LCS|997602^.^L|||19980727175800||||G|||
19980727000000||||||20809880170||19980730041800|||F|997602|||008342
OBX|2|TX|0^Informes^TIAR||Se ha efectuado una RNM de la columna lumbosacra en cortes multiplanares. El examen realizado muestra: Espondilolistesis grado I de L4-L5||||||F|||19980729160500|BN

Ilustración 12. Ejemplo de mensaje el cual identifica imágenes.

[…] opcional,	{…..} permite repetición
MSH	Encabezado de Mensaje
EVN	Tipo de evento
PID	Identificación del paciente
[PD1]	Datos adicionales demográficos
[{ NK1 }]	Familiares a cargo
PV1	Información del episodio
[PV2]	Información adicional del episodio
[{ DB1 }]	Información de discapacidades
[{ ALG }]	Información sobre alergias
[{ DG1 }]	Diagnóstico
[DRG]	Grupo relacionado de Diagnóstico
[{ PR1	Procedimiento
[{ ROL }]	Rol
}]	
[{ GT1 }]	Garante
[{ IN1	Datos de la obra social
[IN2]	Datos de la obra social – Adicionales
[IN3]	Datos de la obra social – Adicionales
}]	
[ACC]	Información de Accidente

Ilustración 13. Ejemplo de mensaje abstracto. (hl7spain.org).

```
MSH|^~\&|NewModernHIS||PharmaCoolX 1.0||20140701123154||ADT^A28|1-alta-adt-a28|T|2.5|||AL|NE
EVN|A28|20140701123153
PID|1||NHC||Carmona^Antonio^Lopez||19840123000000|M|||||||||||||||||||||N
PV1|1|N
```

Ilustración 14. Ejemplo de Mensaje: Al detalle.

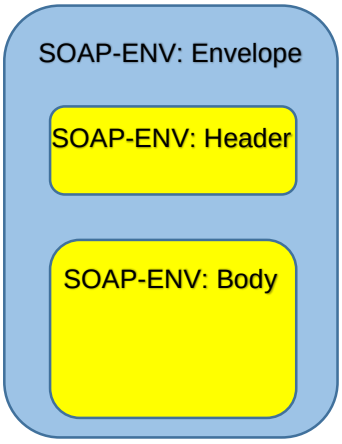


Ilustración 15. Estructura mensaje SOAP (Wikipedia)

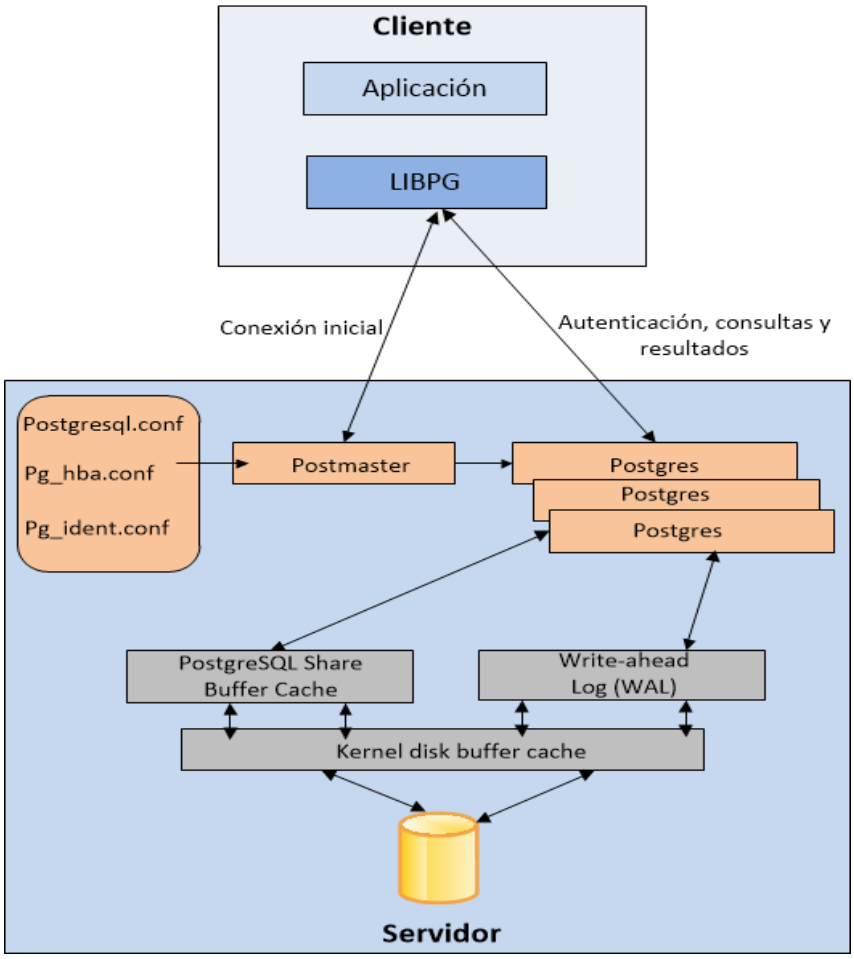


Ilustración 16. Componentes en un sistema PostgreSQL.

Ilustraciones capítulo 4

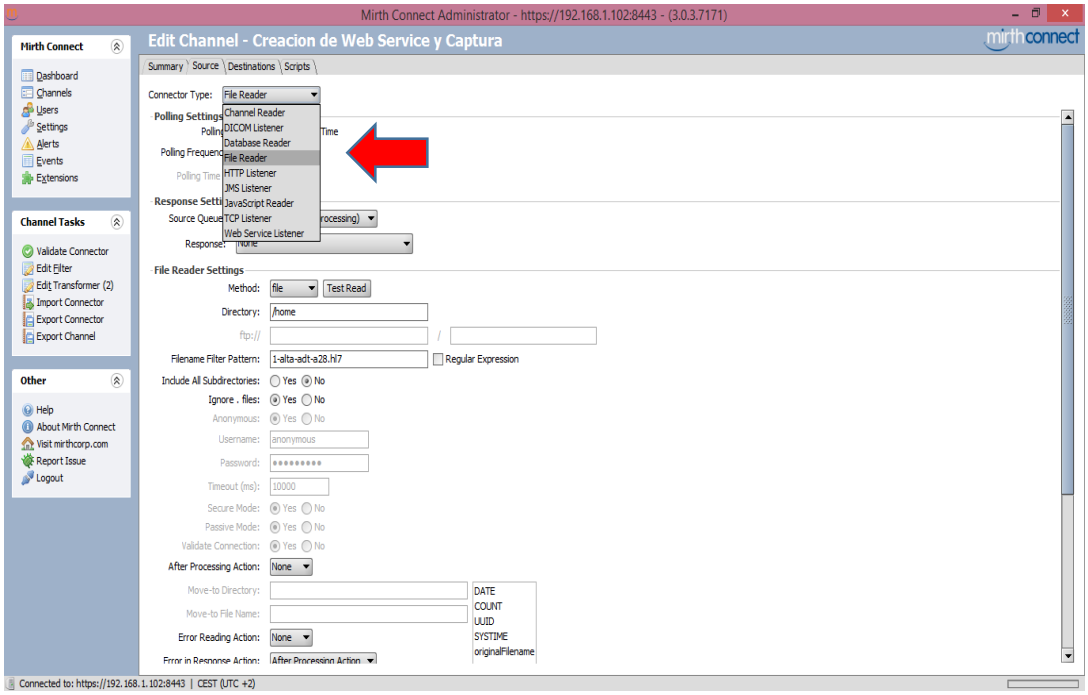


Ilustración 17. Tipos de conexiones dentro de un canal en su pestaña de fuente (Mirth Connect V3.0.3)

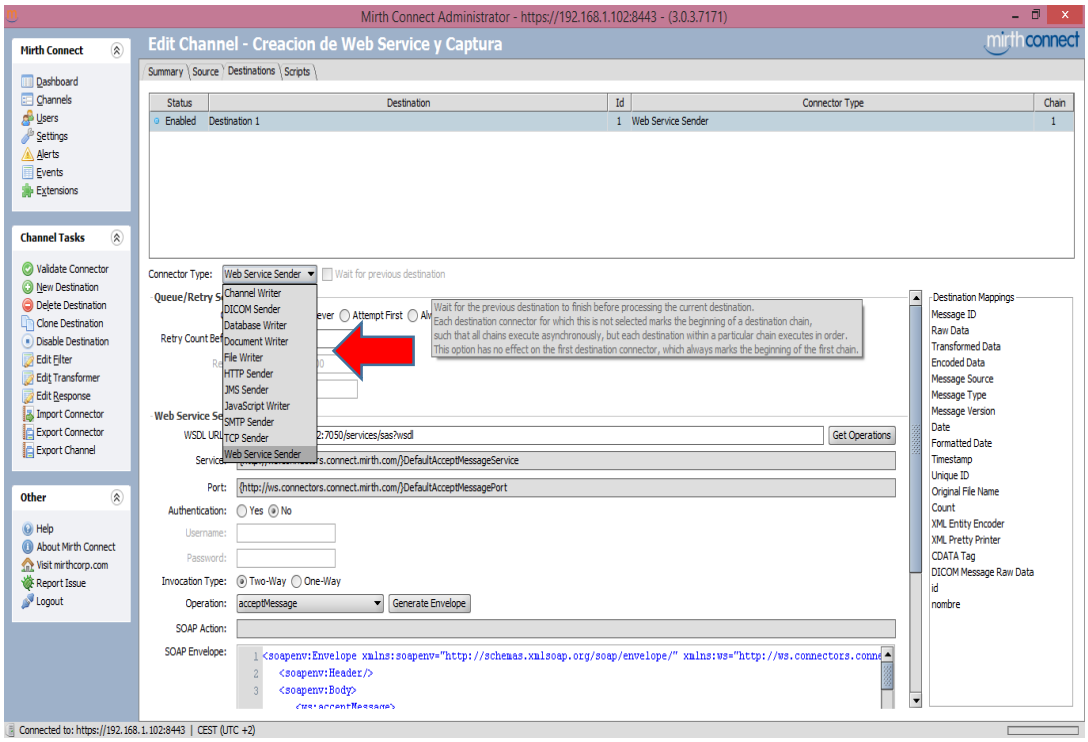


Ilustración 18. Tipos de conexiones dentro de un canal en su pestaña Destino (Mirth Connect V3.0.3)

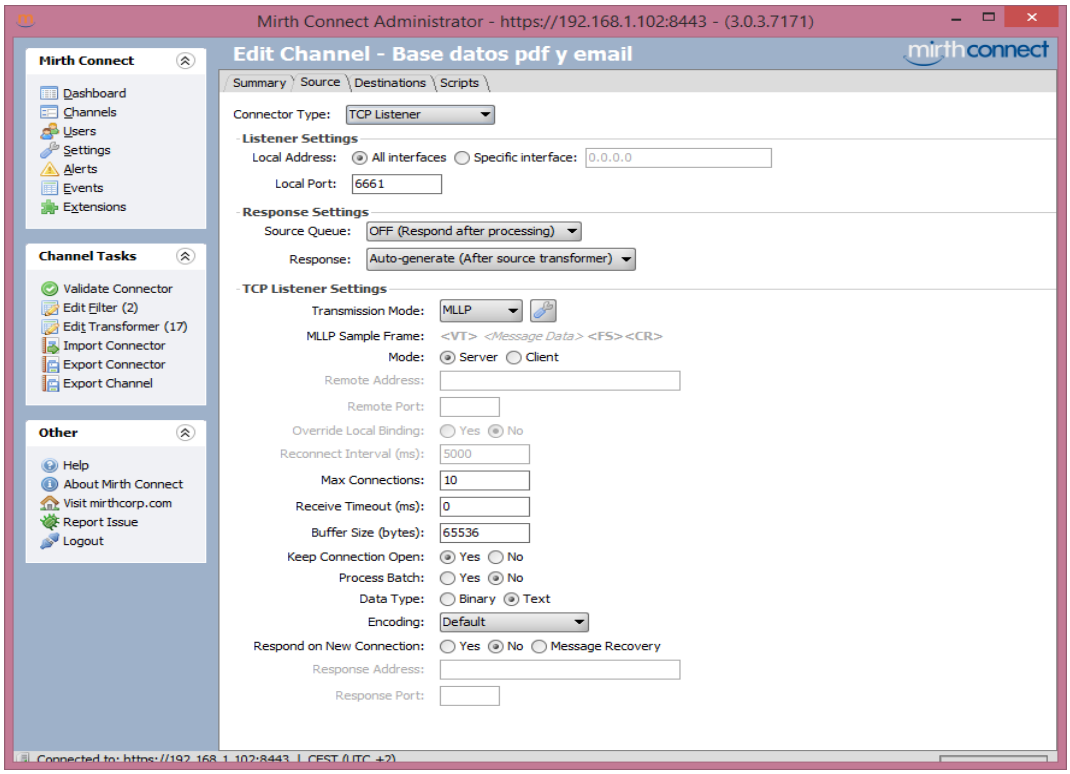


Ilustración 19. Muestra de la ventana mostrada en Mirth del canal Base datos pdf y email.

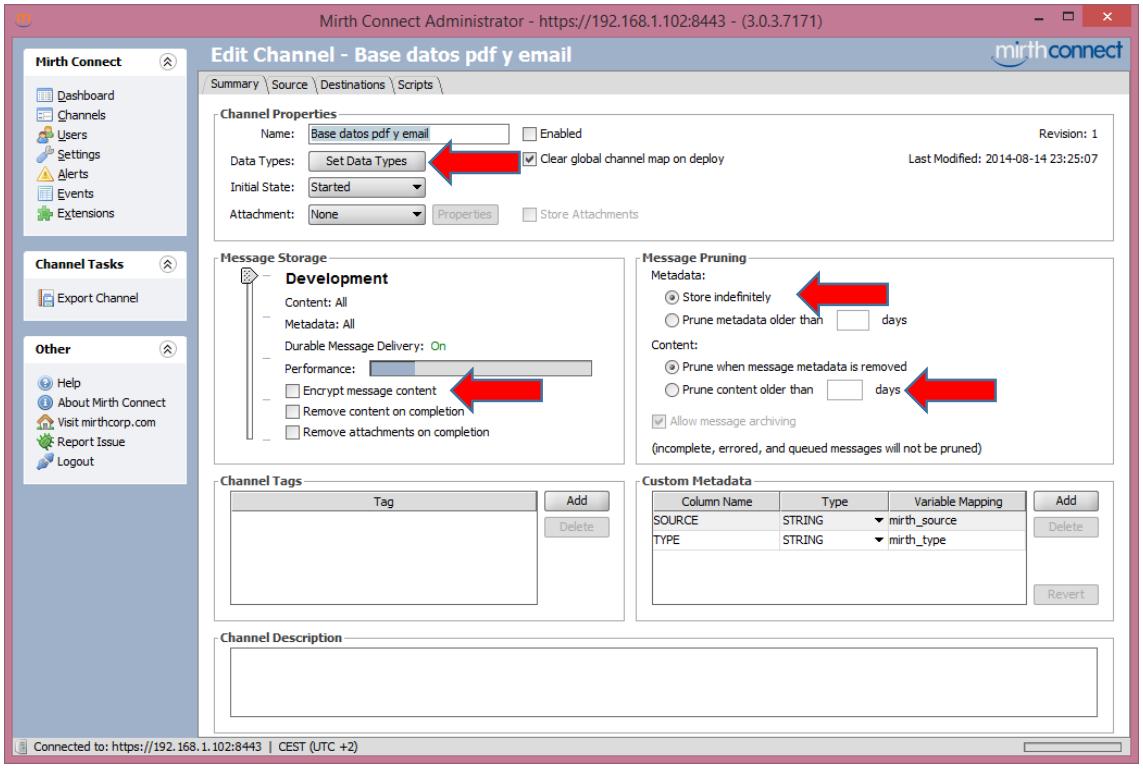


Ilustración 20. Ventana de configuración global del canal Base datos pdf y email.

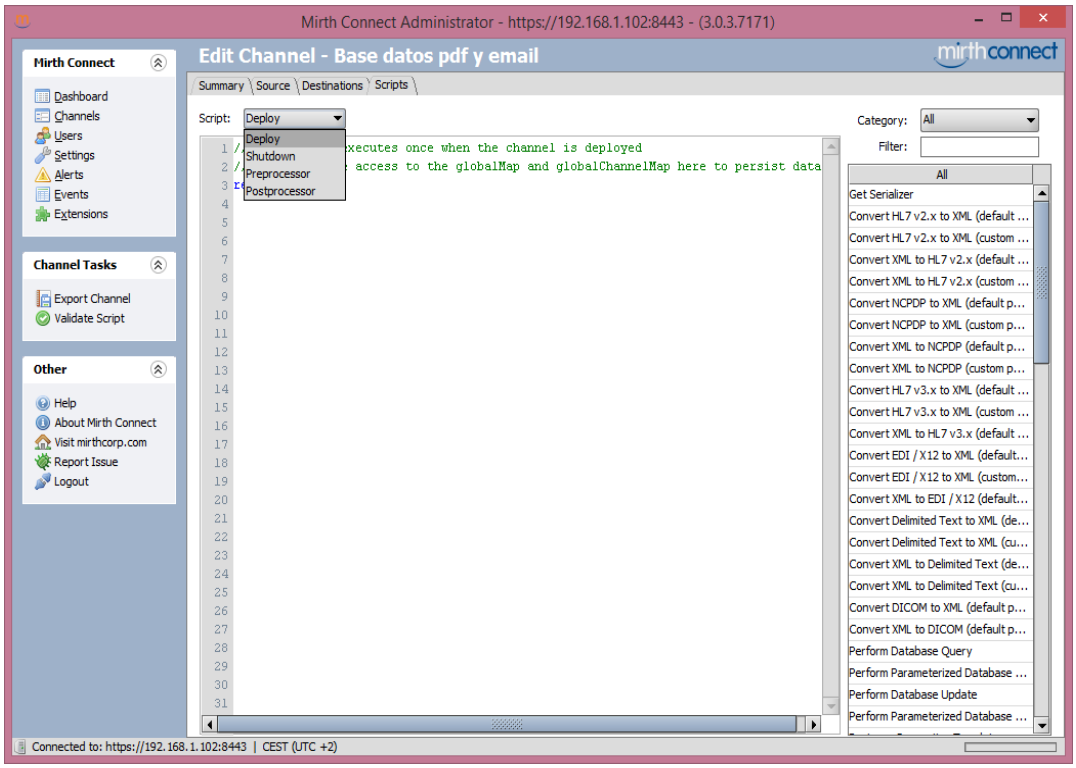


Ilustración 21. Ventana configuración de scripts Javascript.

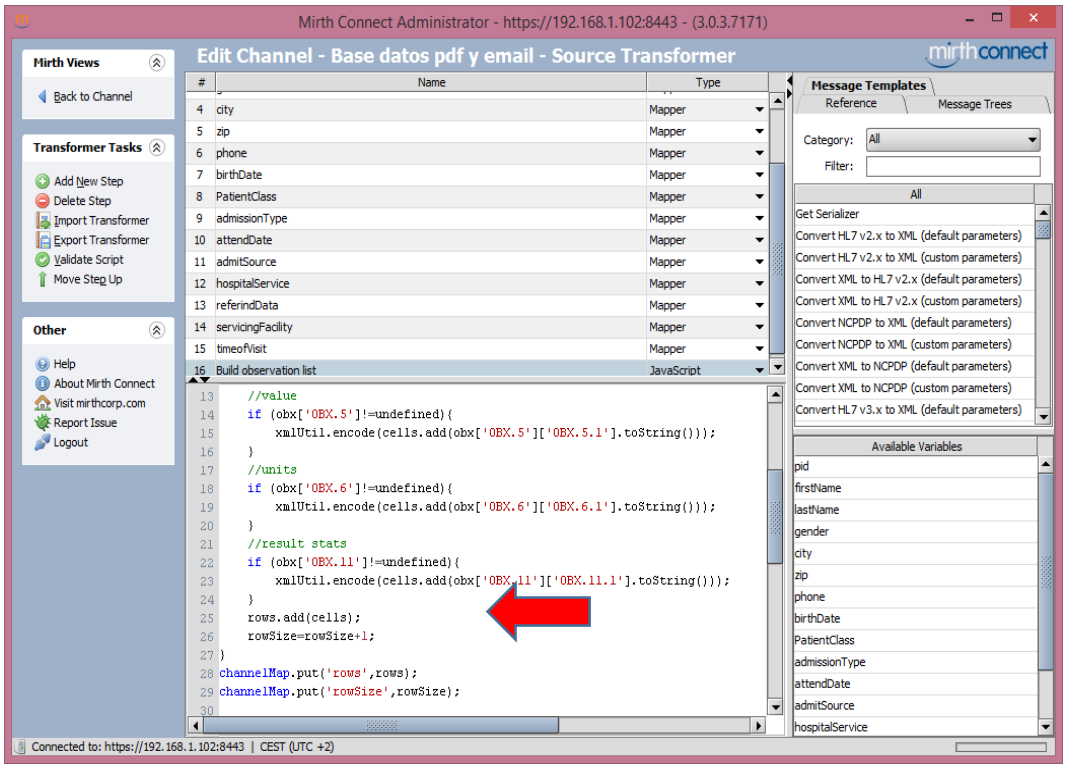


Ilustración 22. Muestra de función javascript dentro del canal fuente en su opción Transformadores.

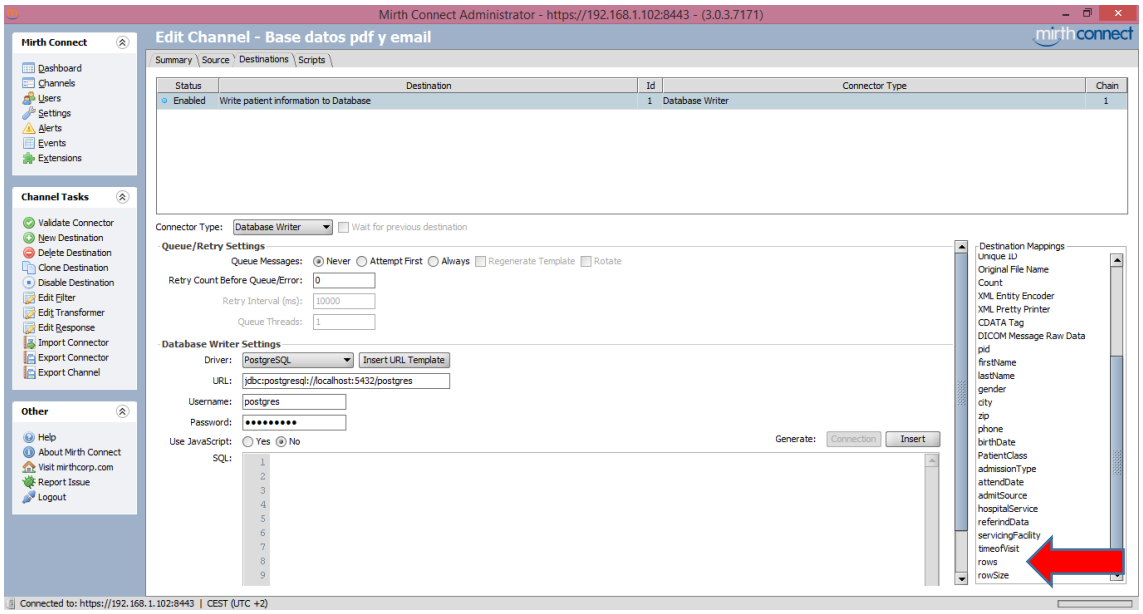


Ilustración 23. Conector destino del canal Base datos pdf y email.

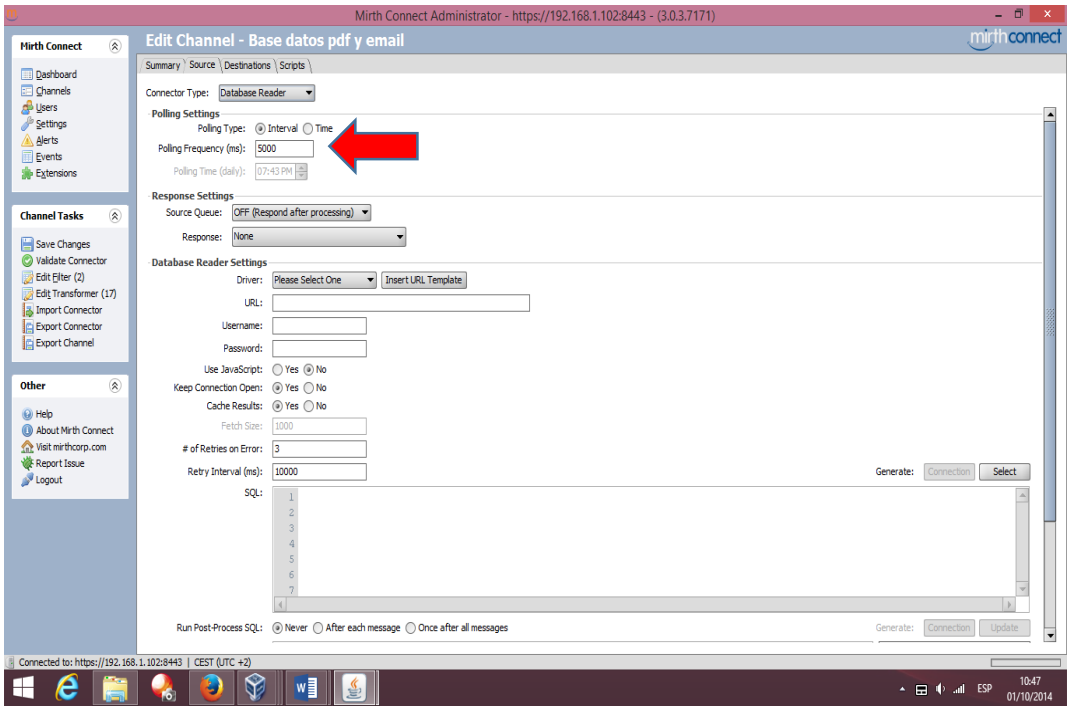


Ilustración 24.- Panel Source del Canal Base datos pdf y email. Con flecha rojo la frecuencia de escucha.

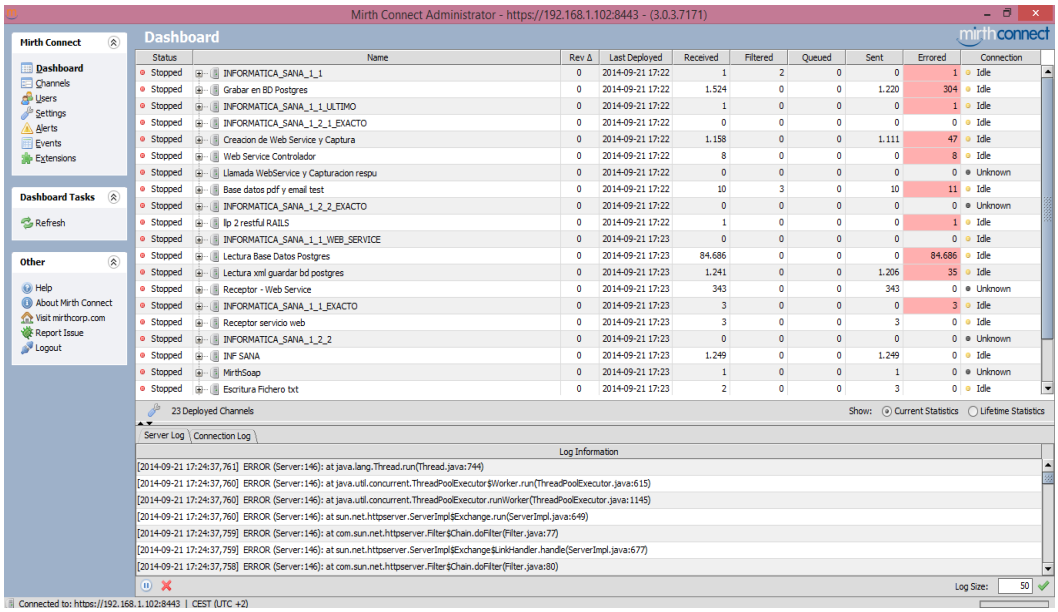


Ilustración 26. Muestra de la pantalla que se puede observar desde el Dashboard.

Ilustraciones capítulo 5

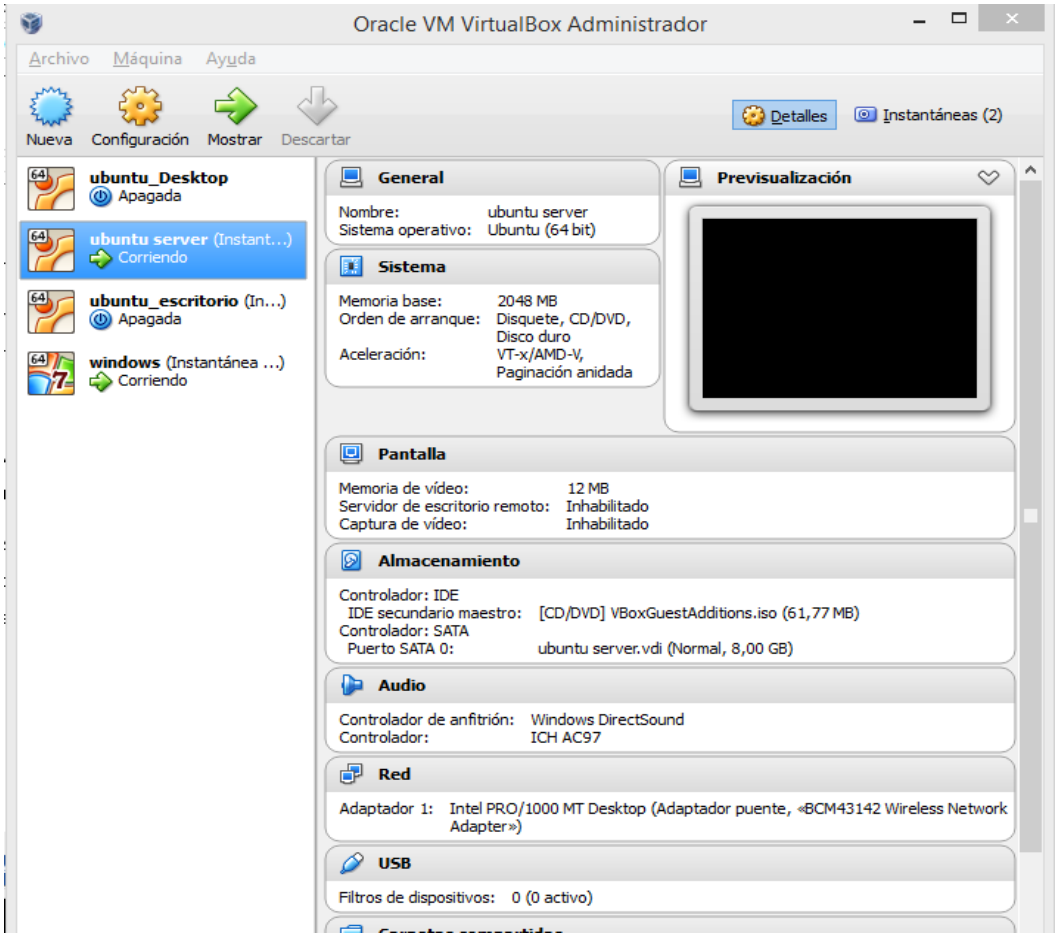


Ilustración 27. Pantalla principal del entorno Oracle VirtualBox VM.

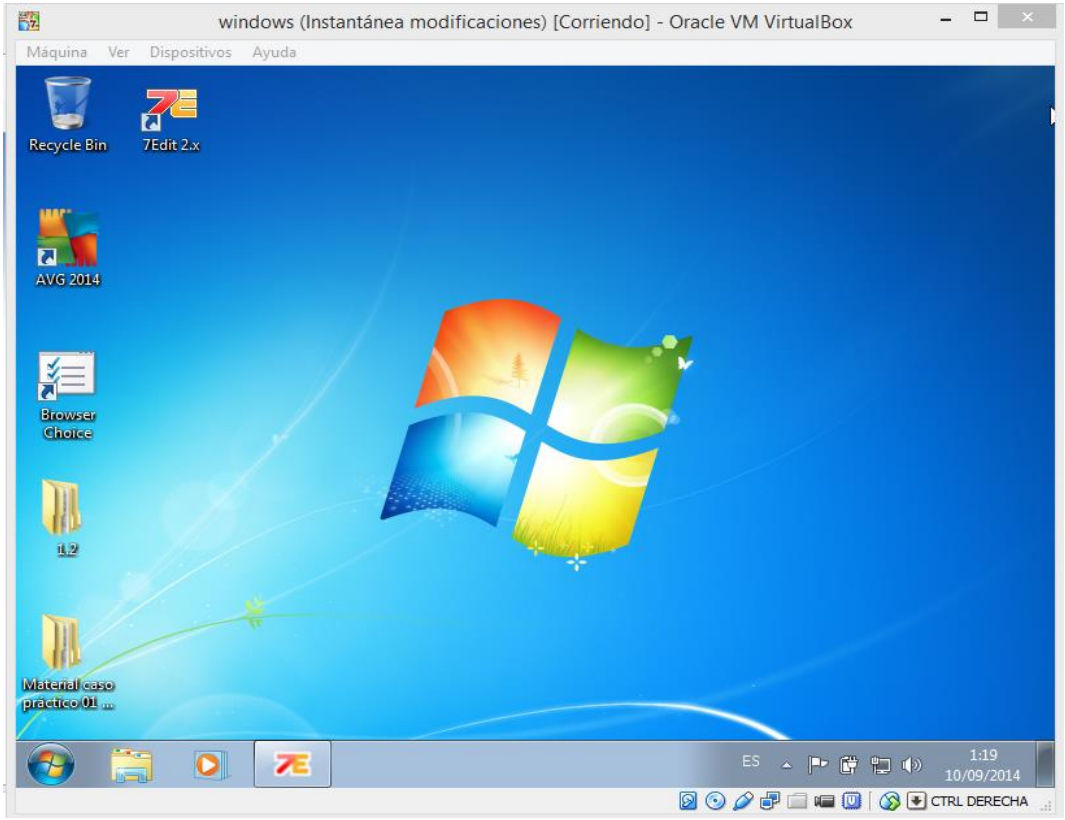


Ilustración 28. Máquina virtual Windows 7.

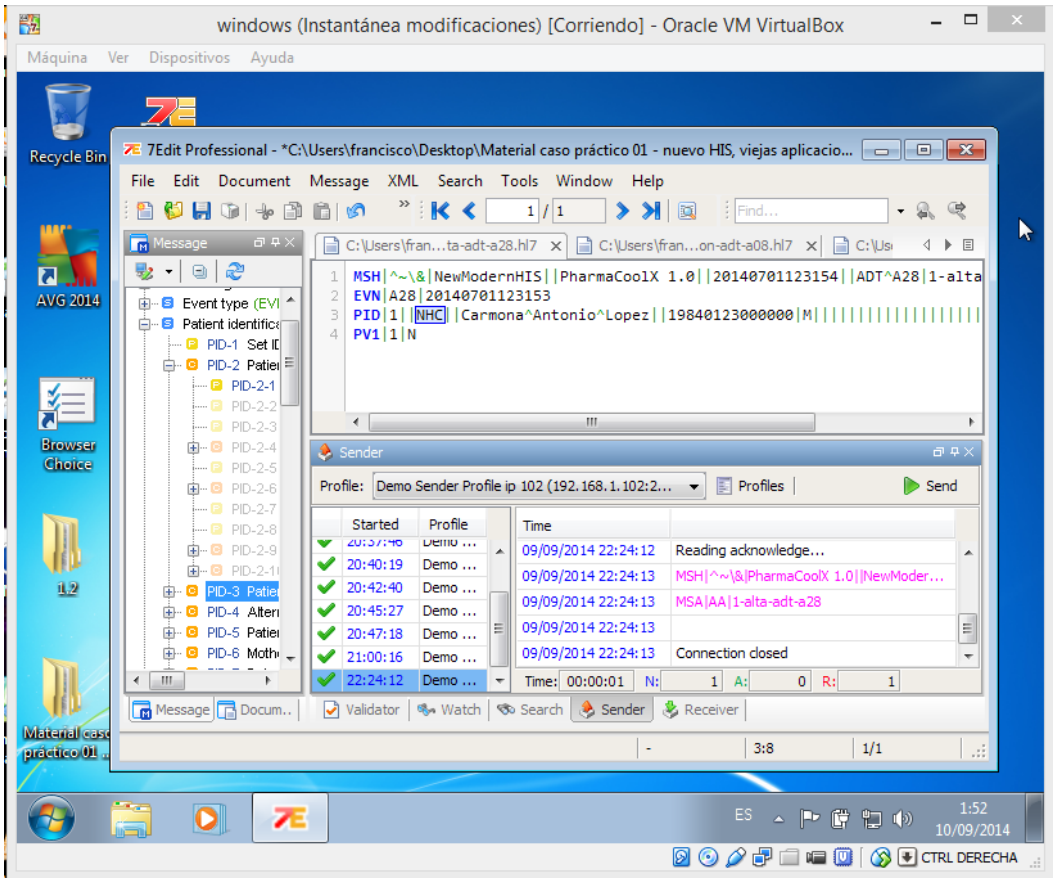


Ilustración 29. Herramienta 7Edit instalada en la máquina virtual Windows 7.

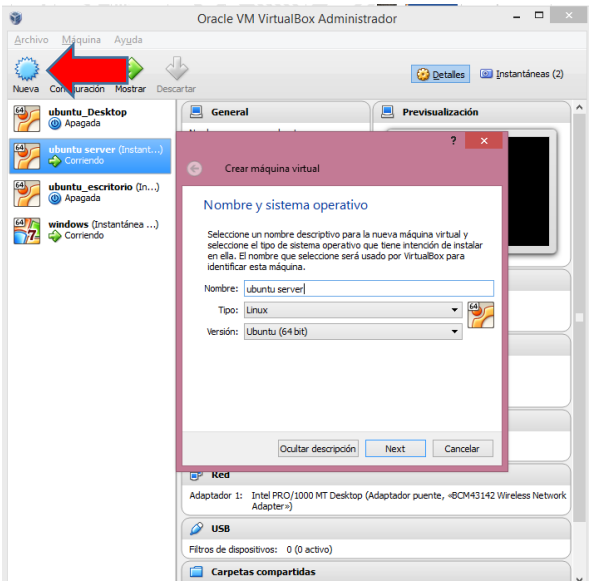


Ilustración 30. Creación máquina virtual Ubuntu server.

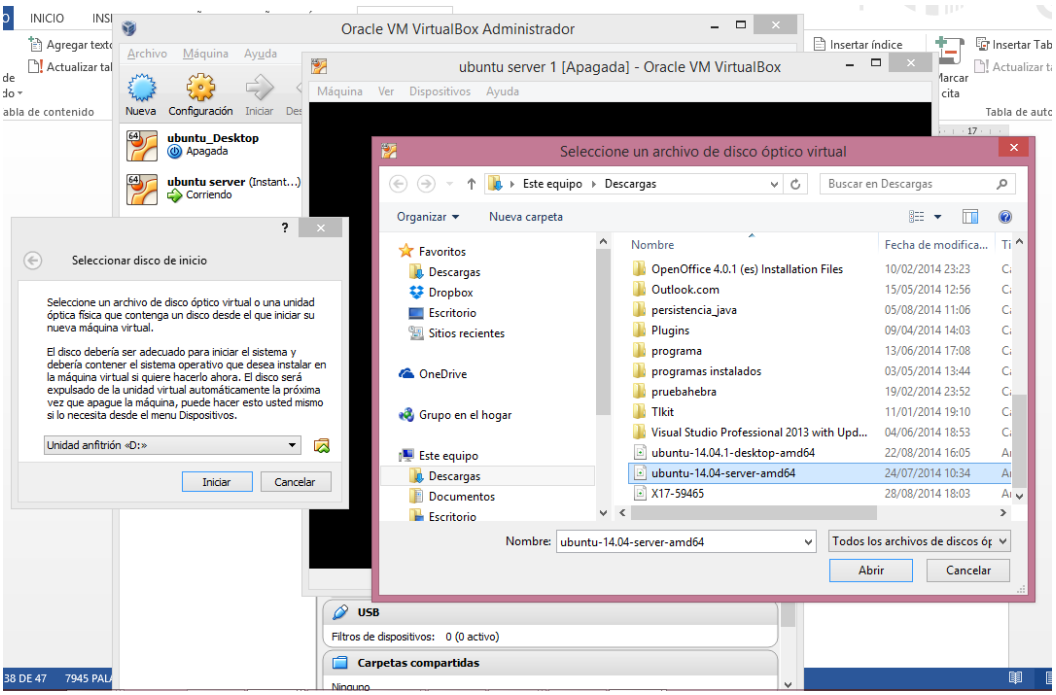


Ilustración 31. Instalación de Ubuntu server 14.04 lts.

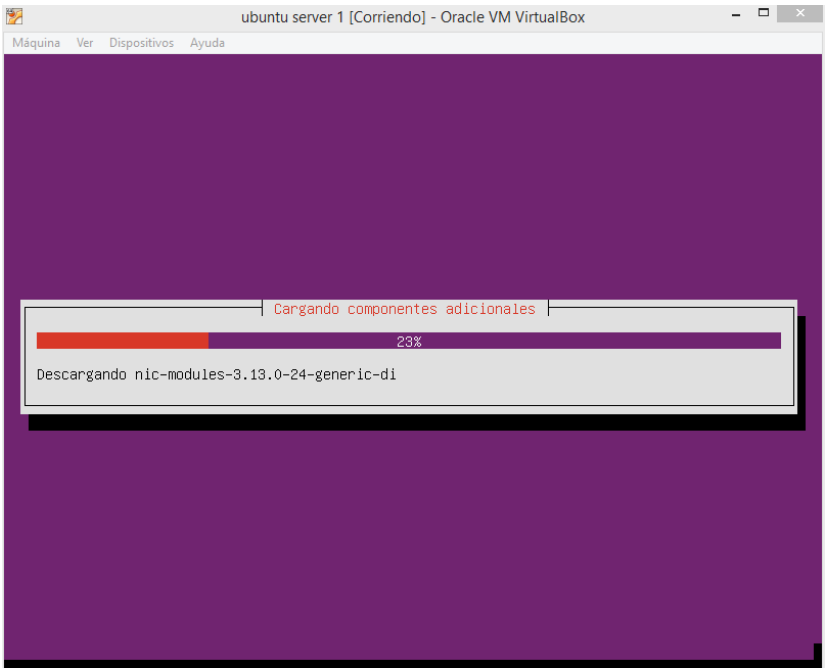


Ilustración 32. Configuración del servidor Ubuntu server 14.04 lts.

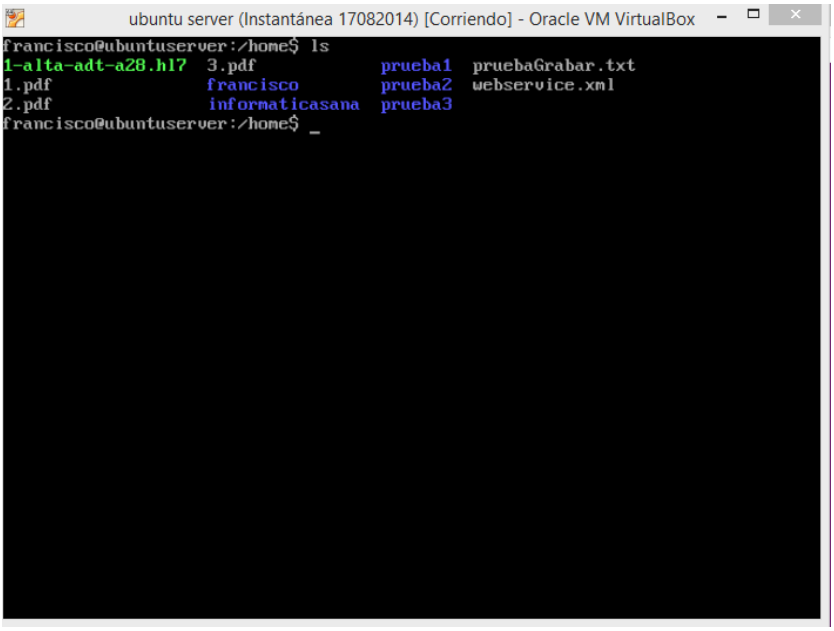


Ilustración 33. Ventana Ubuntu server.

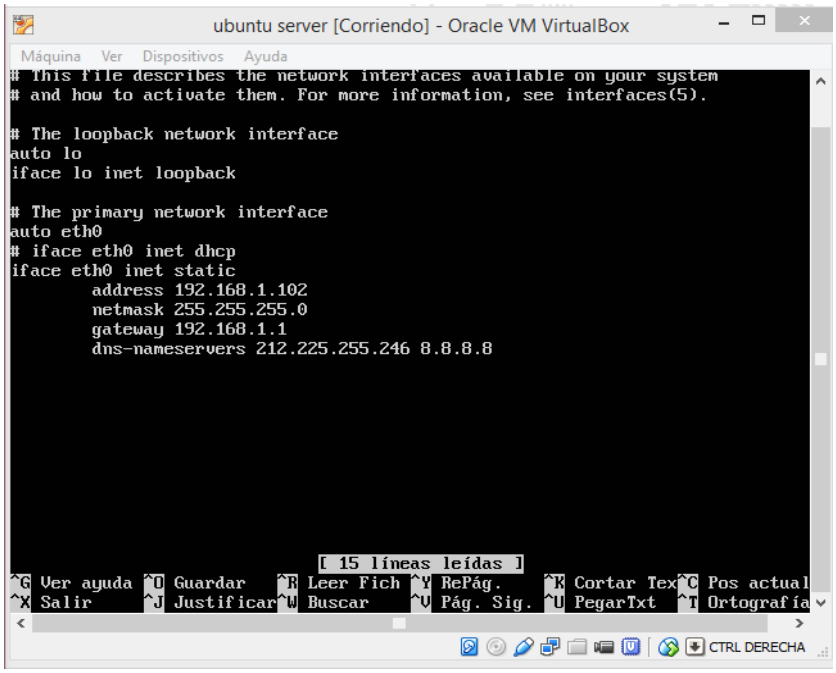


Ilustración 34.- Configuración del archivo interfaces.

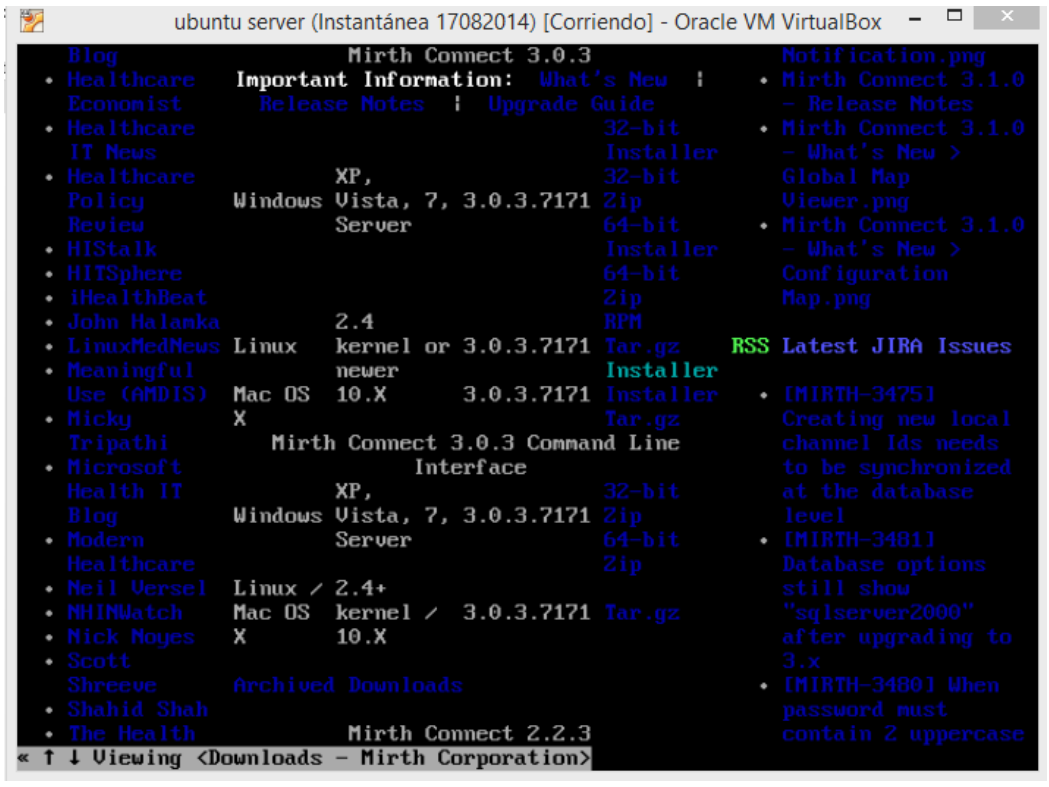


Ilustración 35. Página de descarga del instalador Mirth Connect.

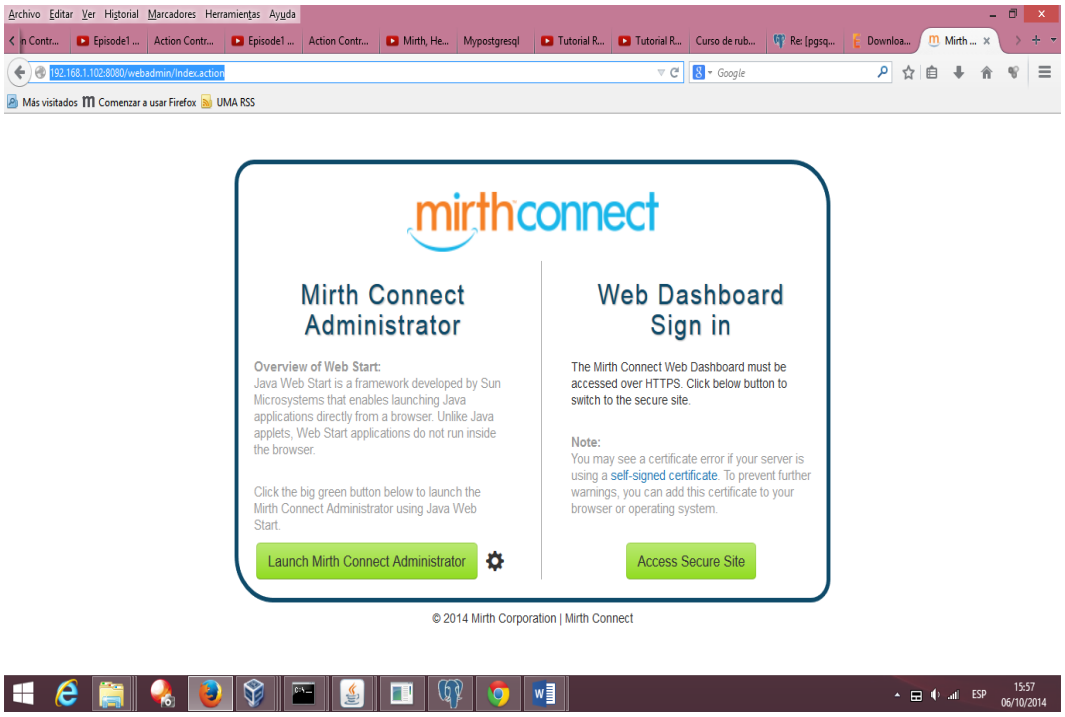


Ilustración 36.- Ventana de acceso a Mirth Connect.

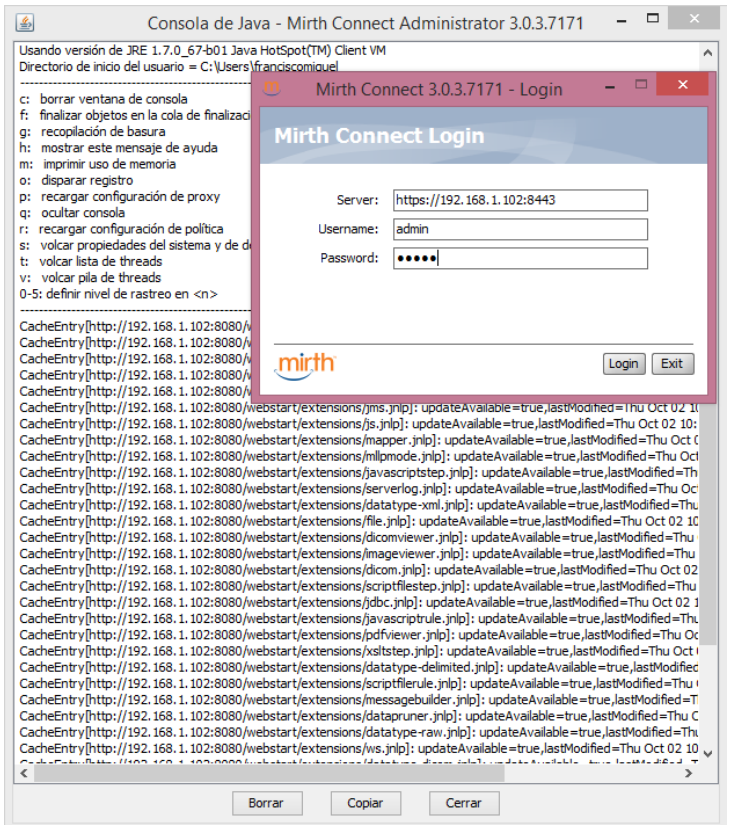


Ilustración 37.- Ventana de login para iniciar el administrador de Mirth Connect.

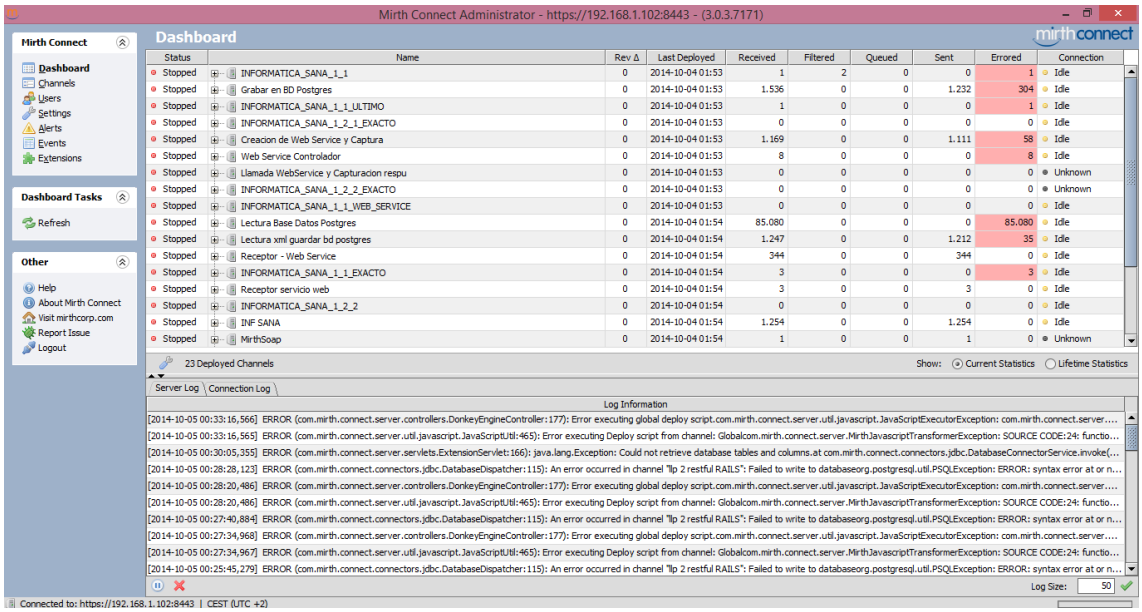


Ilustración 38.- Ventana de Mirth Connect. Dashboard.

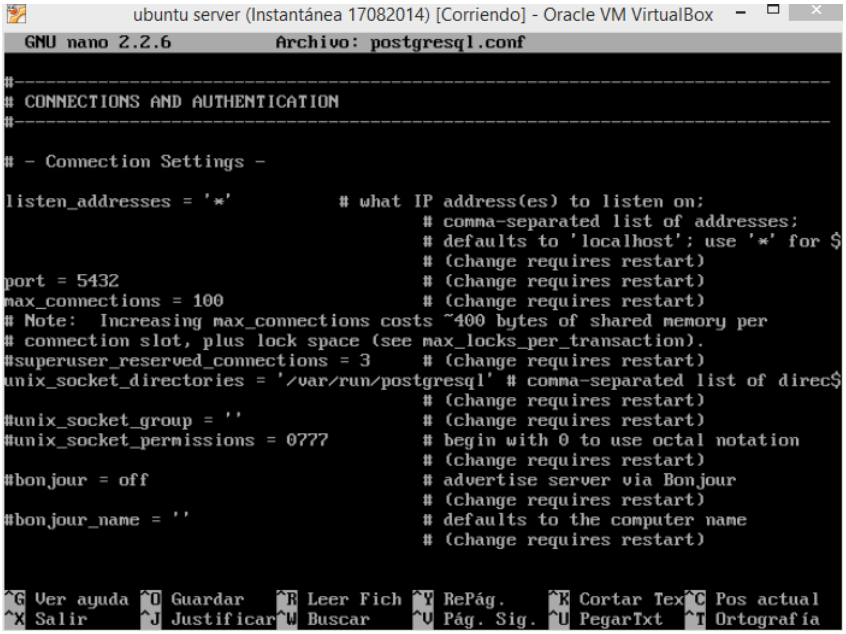


Ilustración 39.- Contenido del fichero postgresql.conf

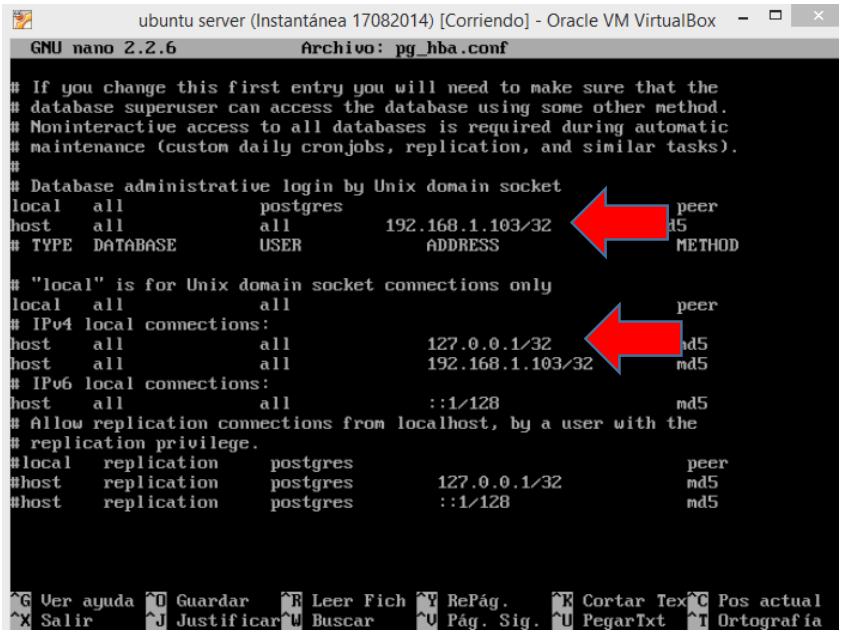


Ilustración 40.- Configuración fichero pg_hba.conf.

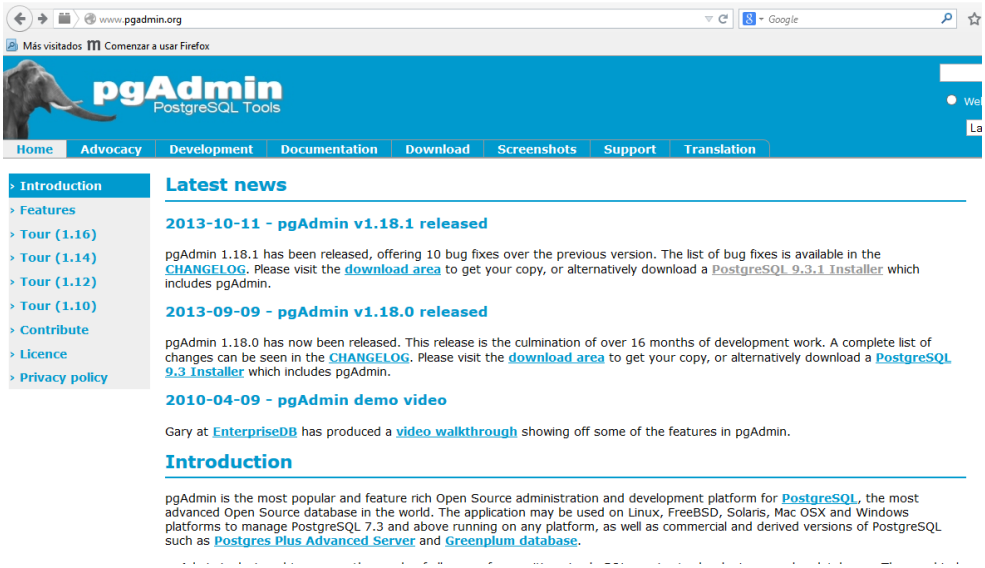


Ilustración 41.- Página de descarga del administrador de PostgreSQL.

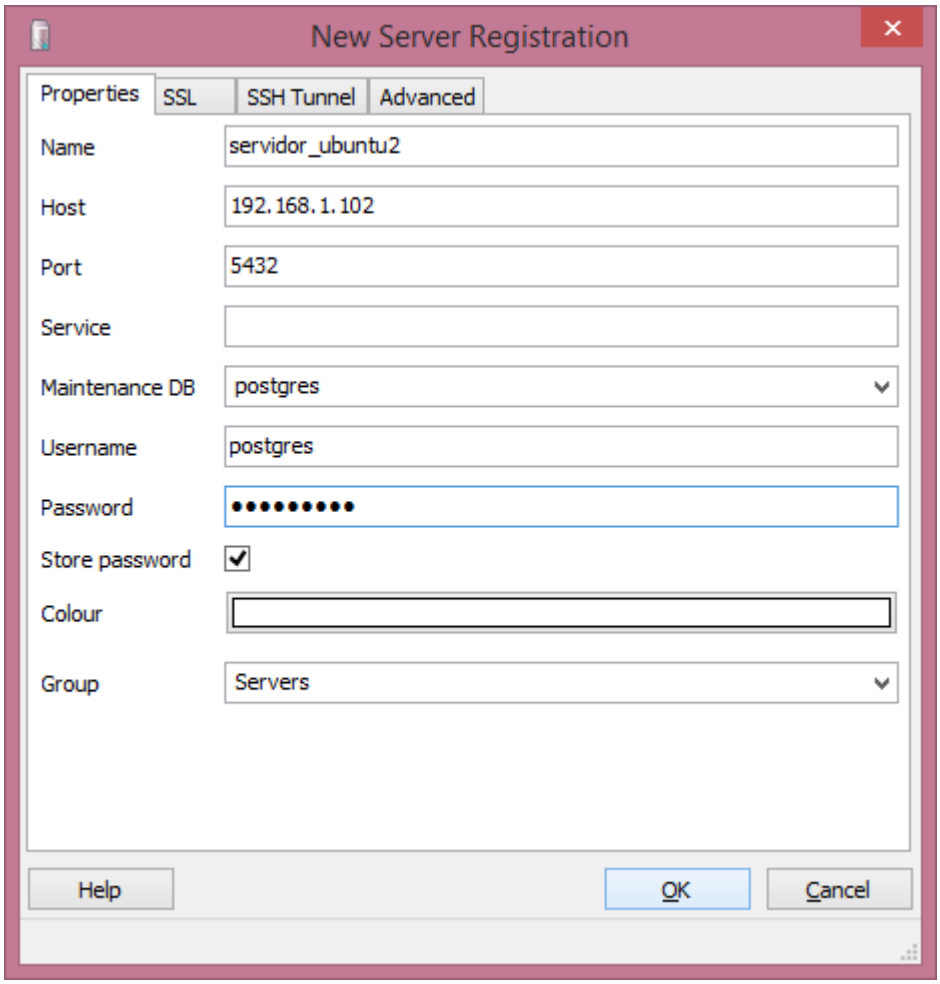


Ilustración 42.- Configuración de un servidor en PgAdmin III.

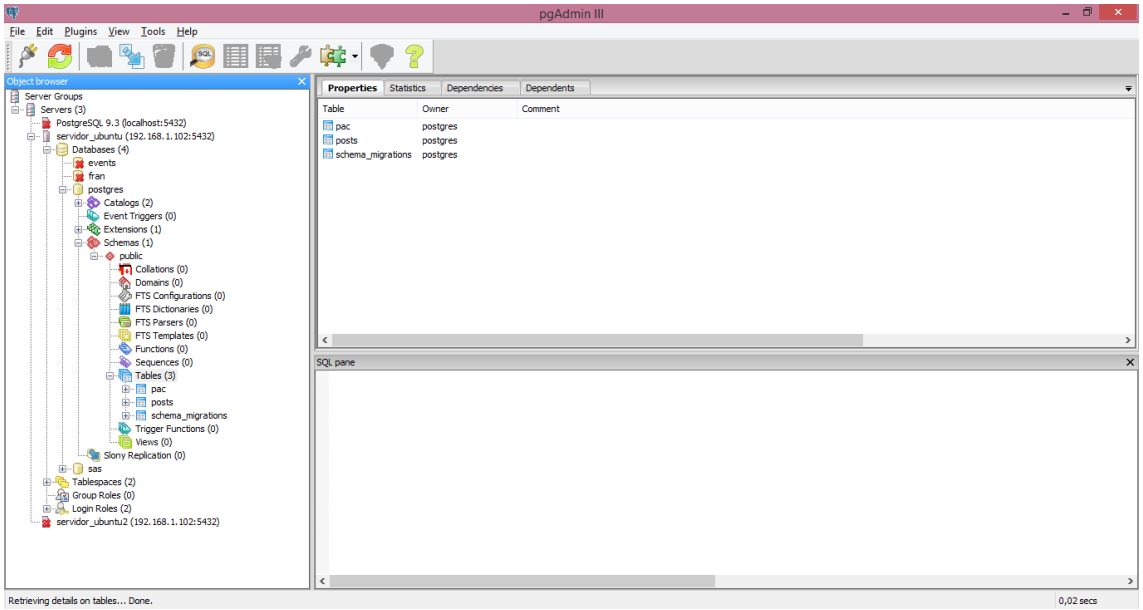


Ilustración 43.- Pantalla principal del Administrado de PostgreSQL.

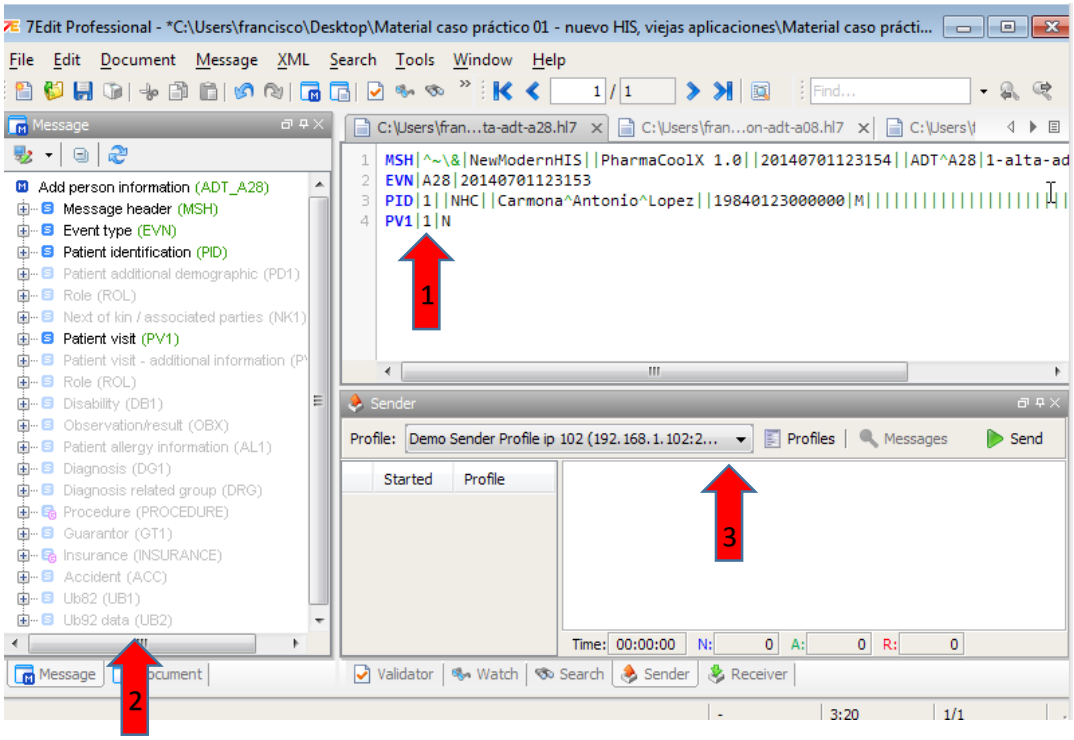


Ilustración 44. Ventana principal 7Edit.

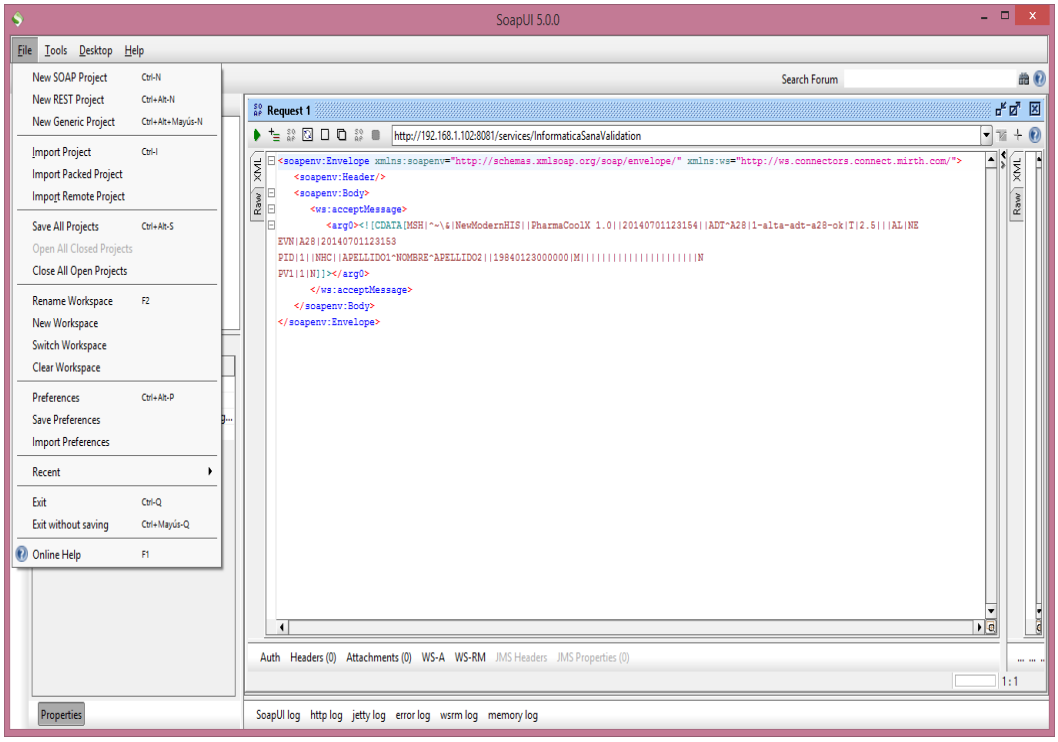


Ilustración 45. Ventana para la creación de nuevos proyectos y entornos de trabajo.

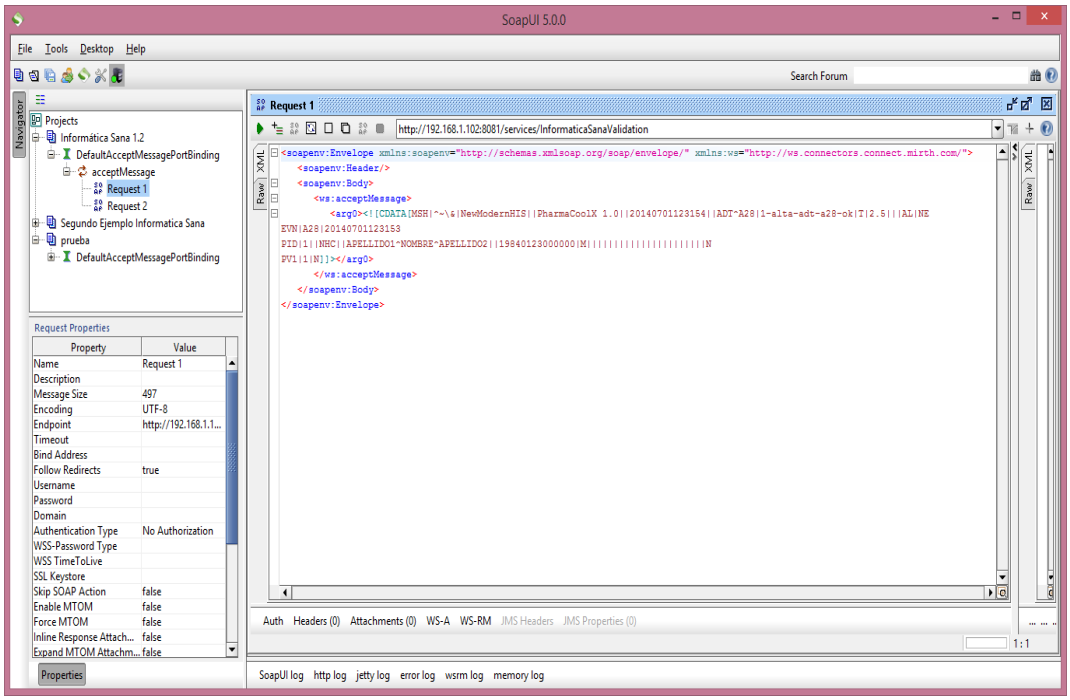


Ilustración 46. Muestra de una llamada al WS InformaticaSanaValidation.

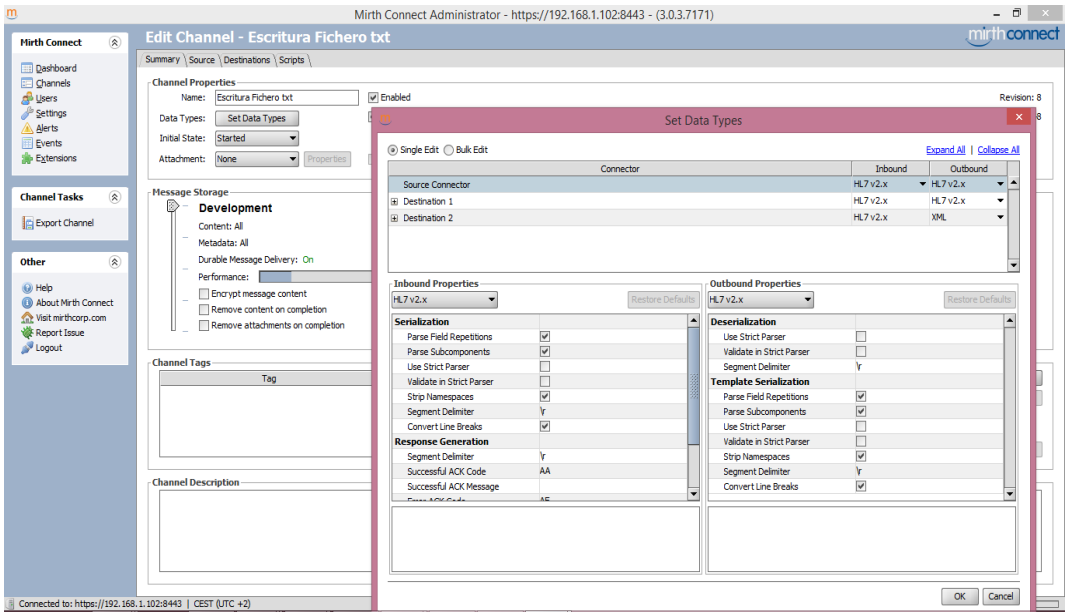


Ilustración 47.- Ventana configuración del tipo de datos para la entrada y la salida.

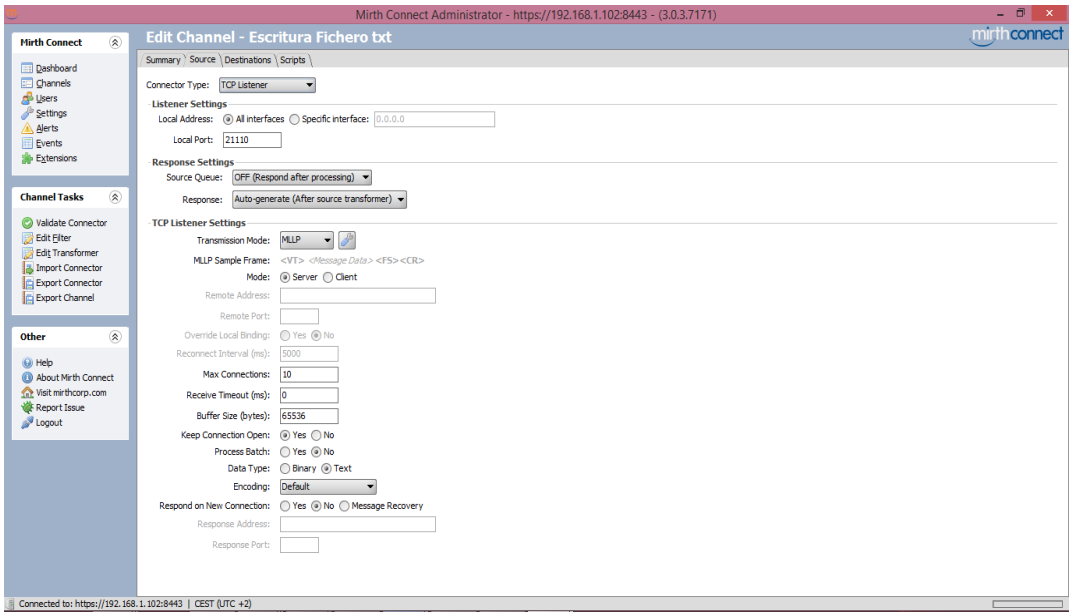


Ilustración 48.- Ventana configuración del conector fuente.

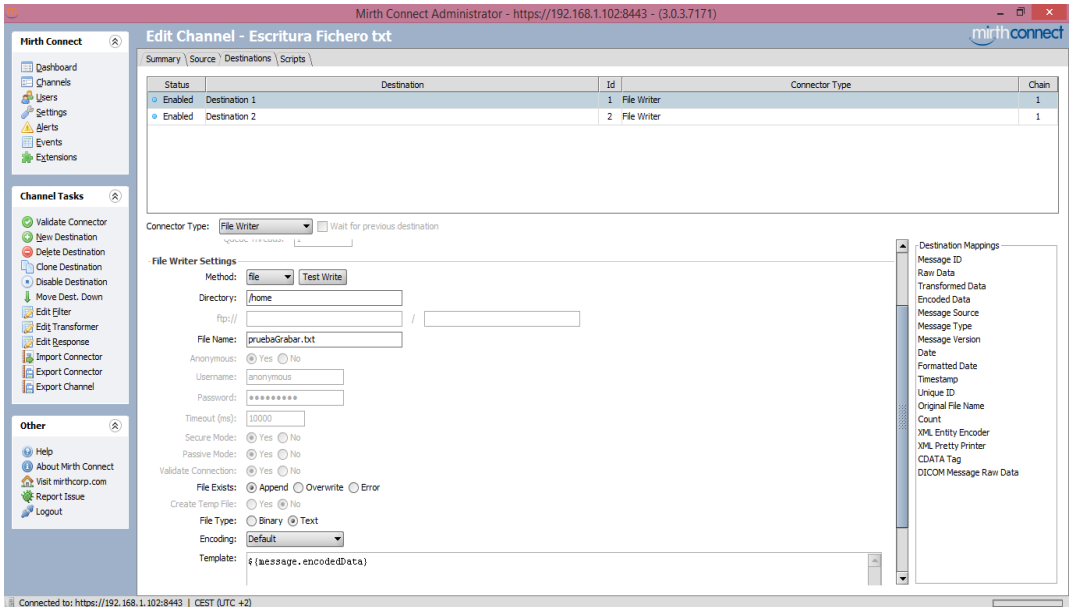


Ilustración 49.- Ventana configuración del conector destino1 para el canal Escritura Fichero txt.

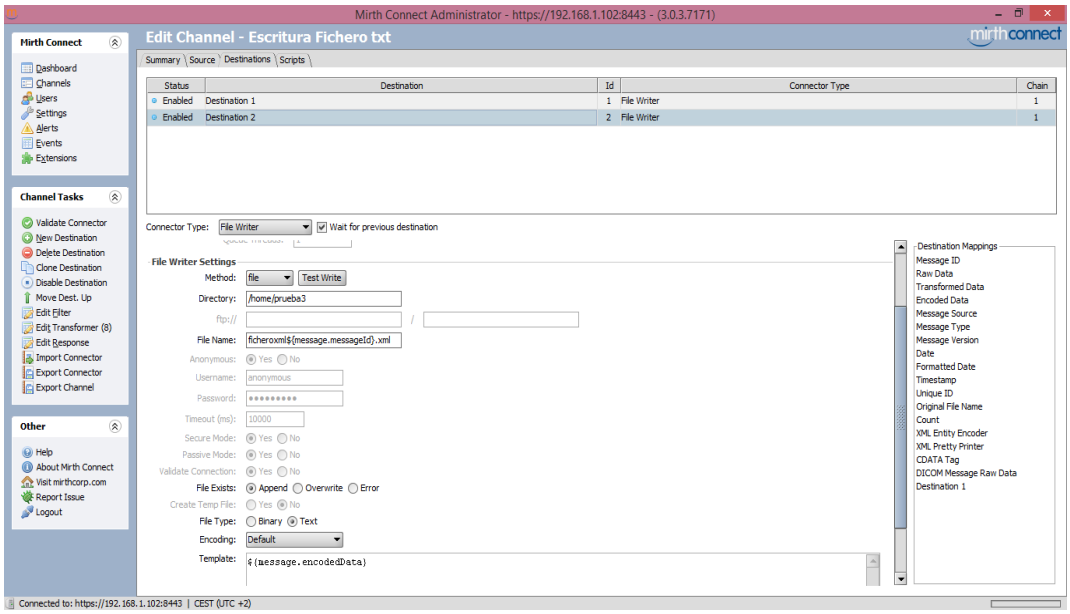


Ilustración 50.- Ventana configuración del conector destino2 para el canal Escritura Fichero txt.

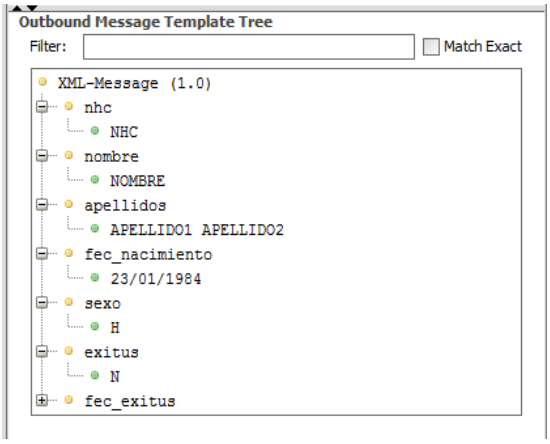


Ilustración 51.- Campos configuradores del mensaje XML.

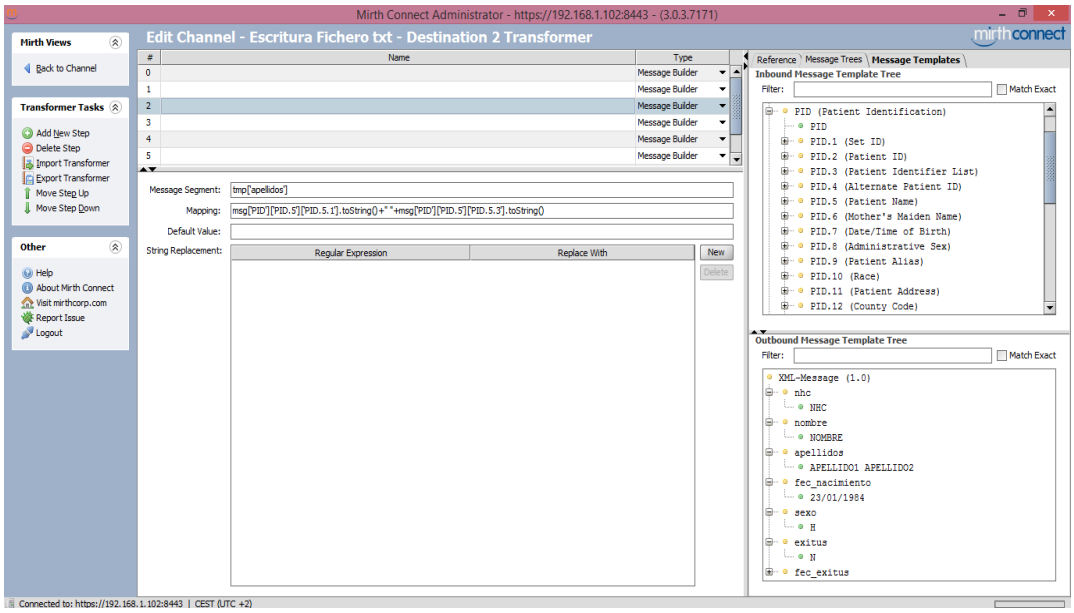


Ilustración 52.- Transformadores usados para la creación de la salida del conector destino2. Entrada HL7, salida XML.

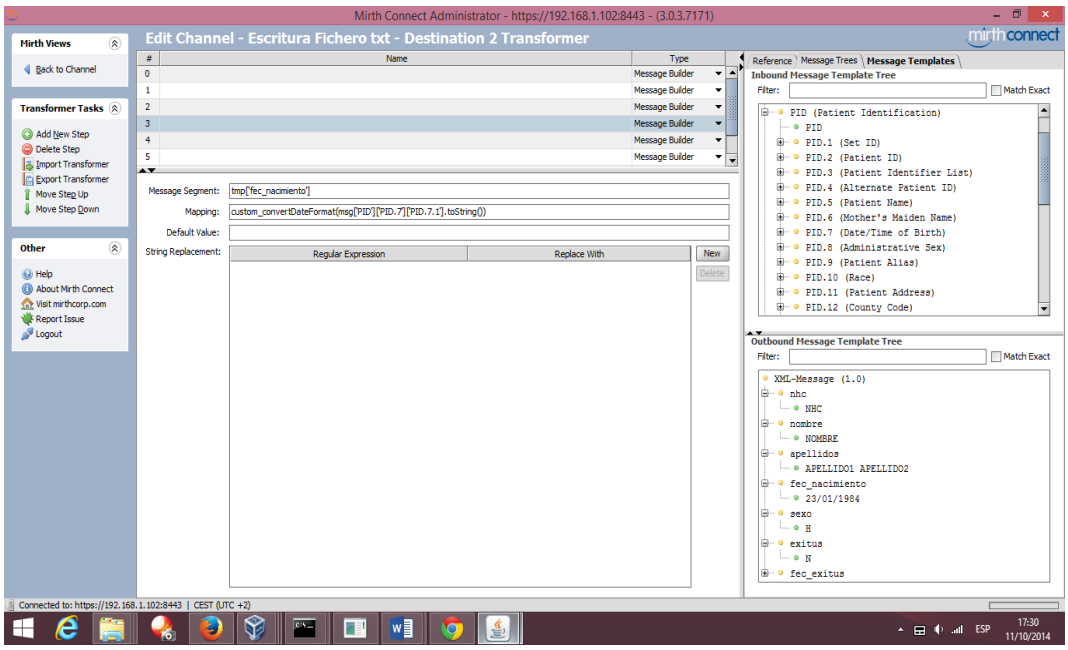


Ilustración 53.- Configuración del transformador fecha nacimiento para el formato español.

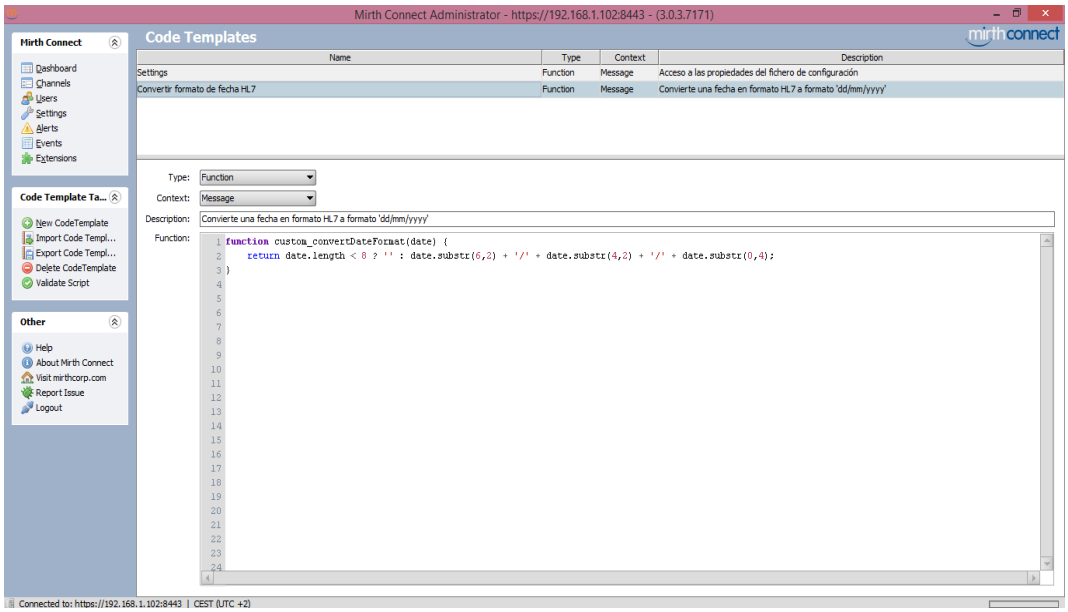


Ilustración 54.- Script global para convertir la fecha introducida desde el mensaje HL7 a formato español.

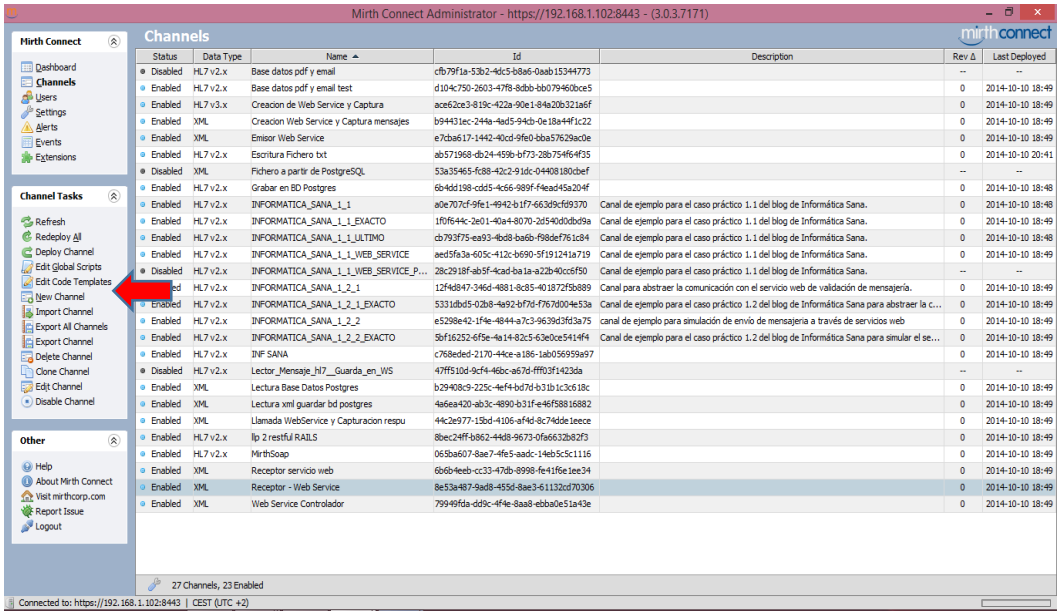


Ilustración 55.- Ventana Mirth para la creación de un nuevo canal.

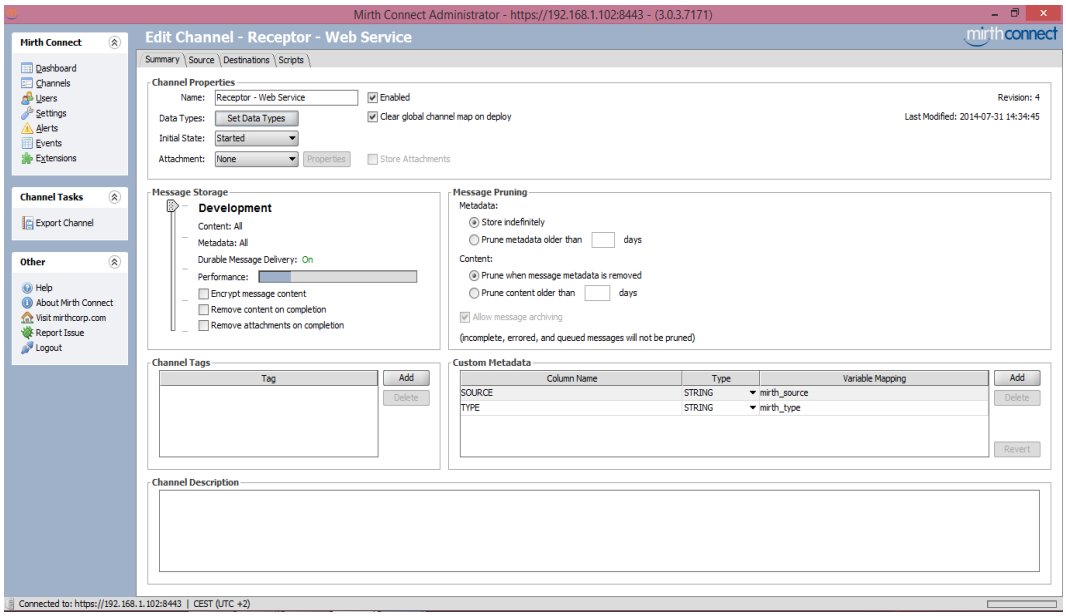


Ilustración 56.- Ventana para la configuración del nombre del canal, datos de entrada a los conectores,…

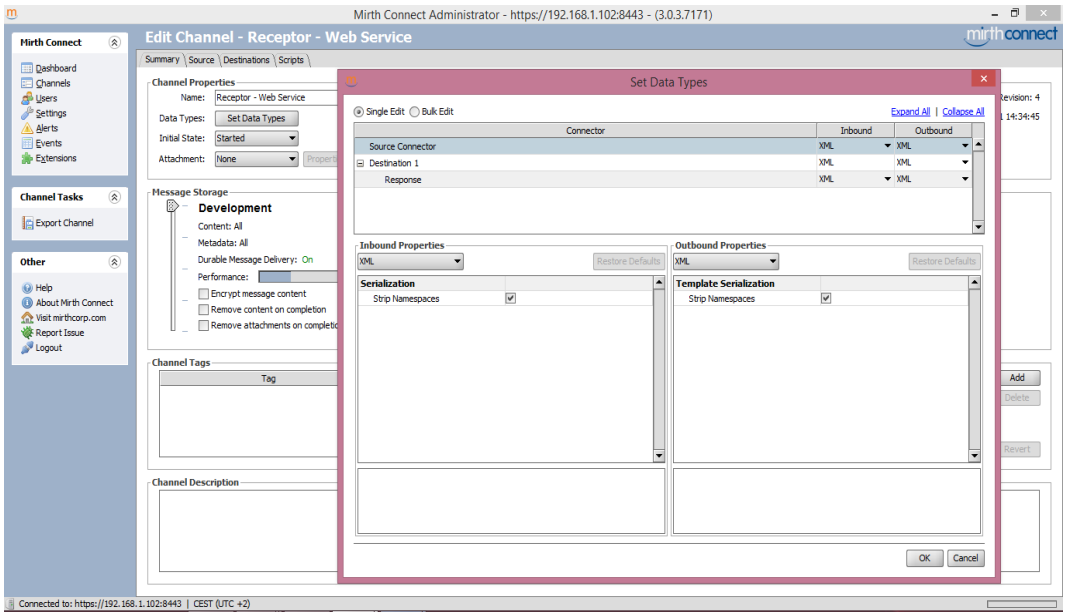


Ilustración 57.- Ventana de configuración del tipo de dato para el conector de entrada y de salida.

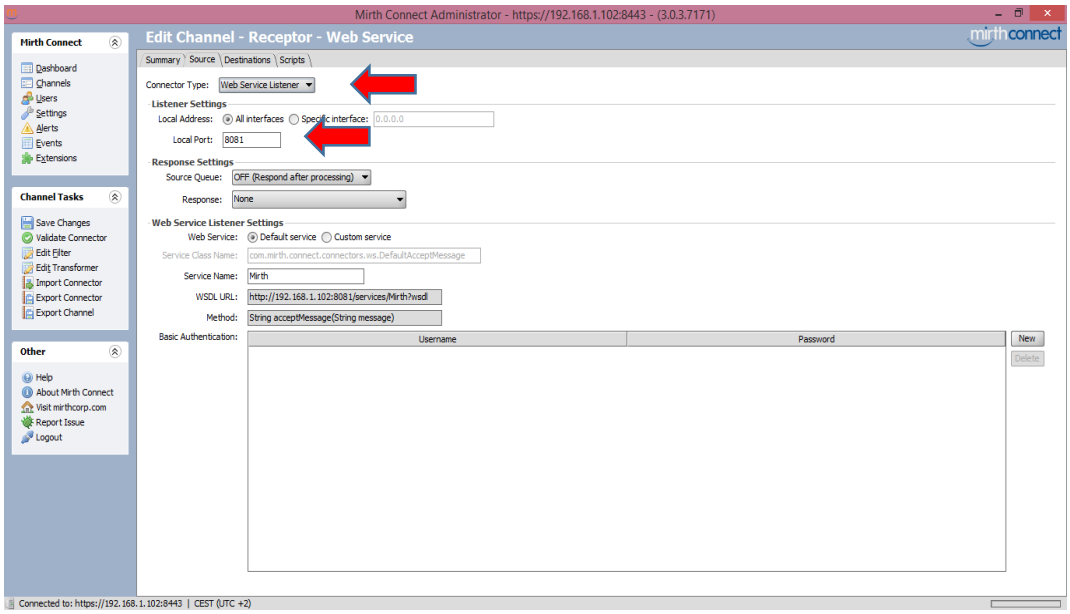


Ilustración 58.- Configuración del conector fuente para el canal Receptor – Web Service.

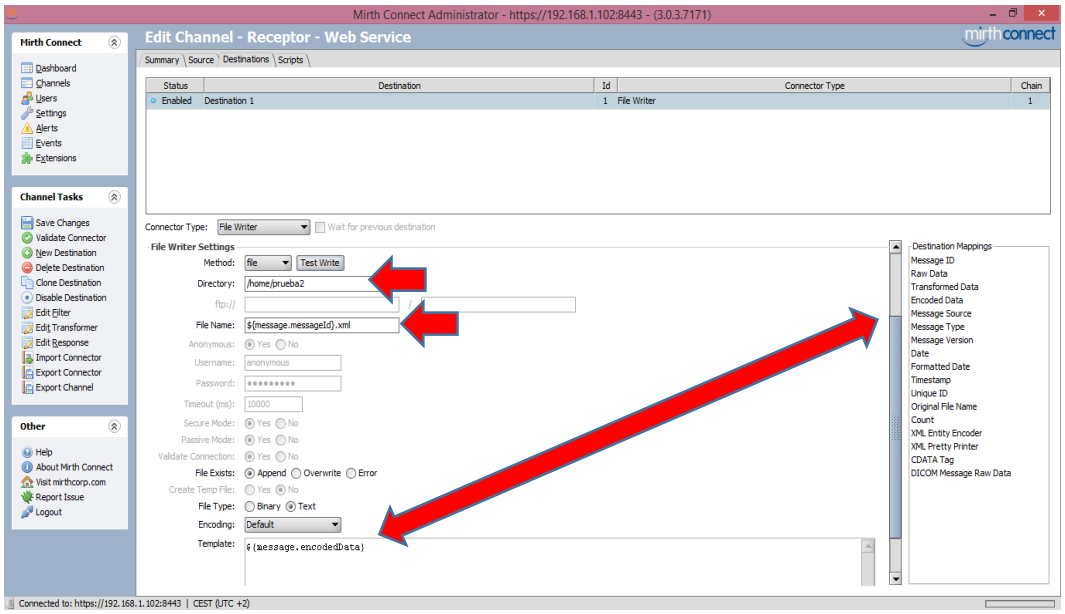


Ilustración 59.- Configuración del conector destino del canal Receptor – Web Service.

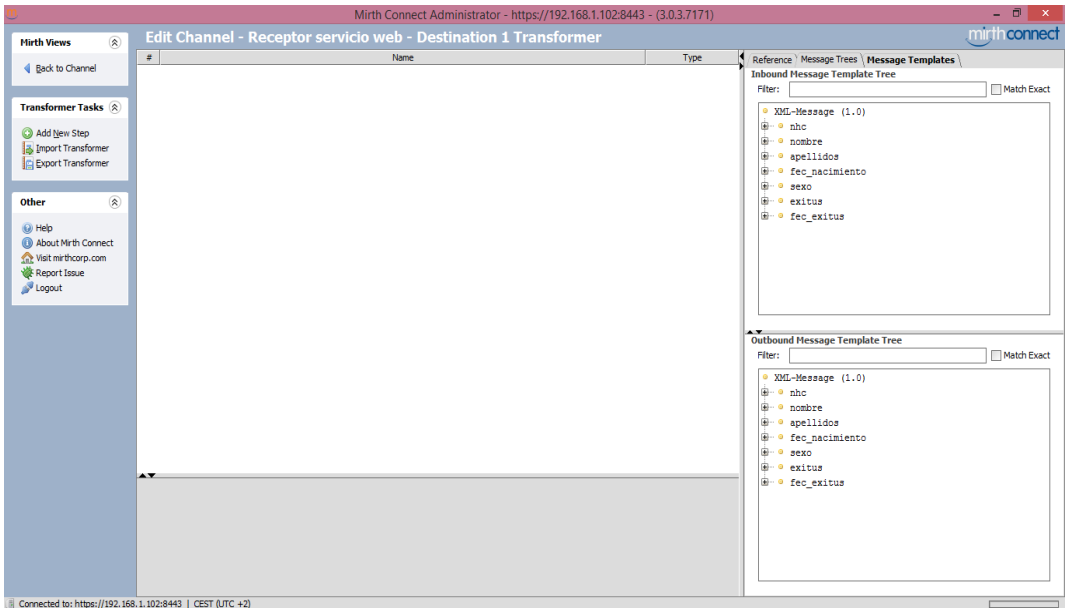


Ilustración 60.- Puesta en escena de XML en el Inbound y Outbound.

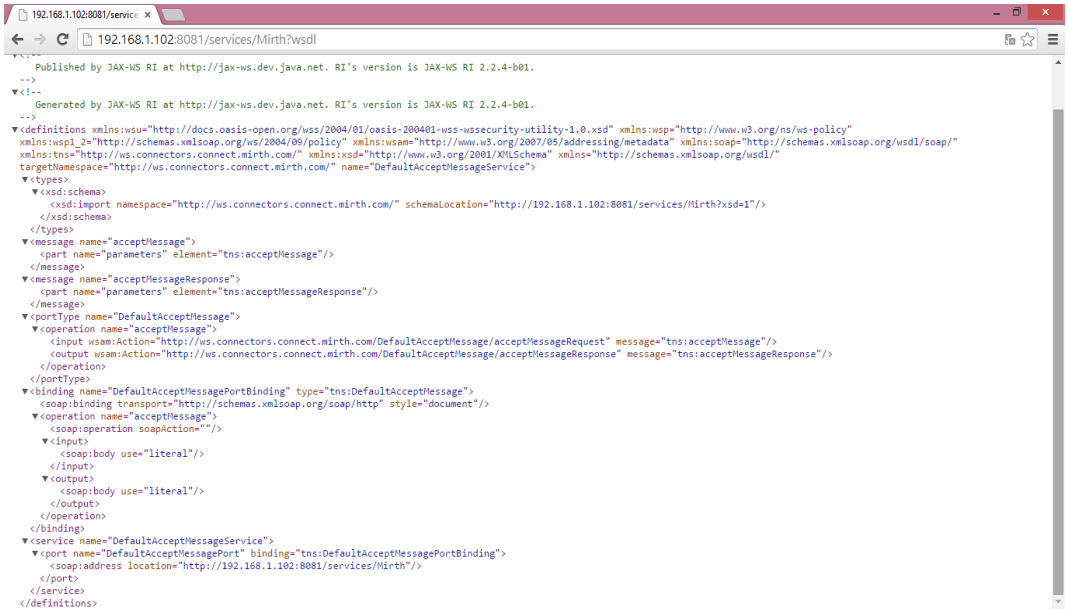


Ilustración 61.- Servicio Web creado automáticamente por Mirth.

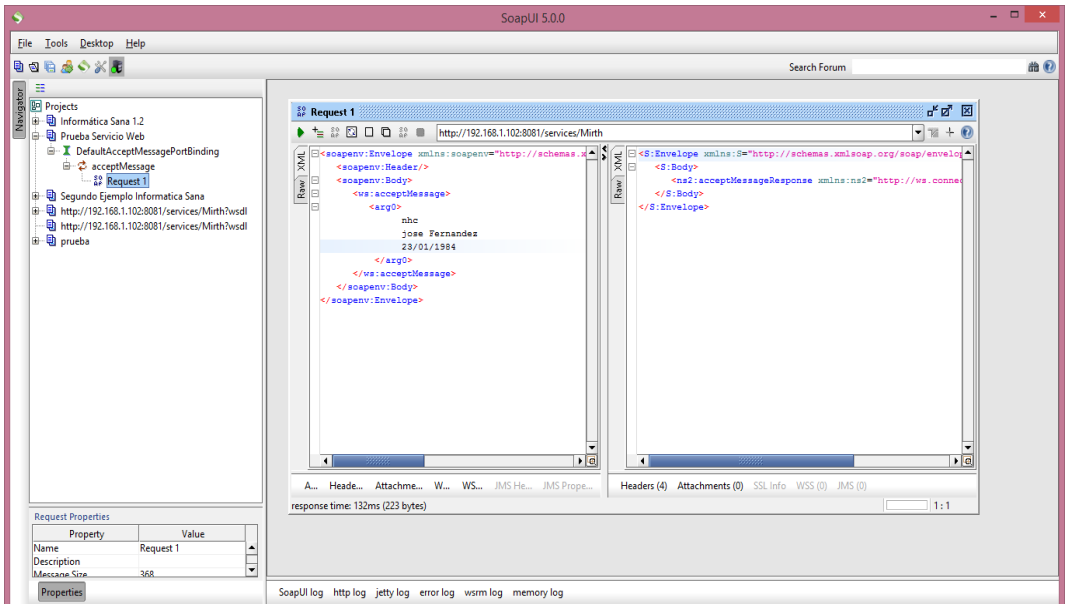


Ilustración 62.- Prueba de funcionamiento del Servicio Web denominado Mirth.

Ilustración 63.- Fichero creado por Mirth en el Servidor.

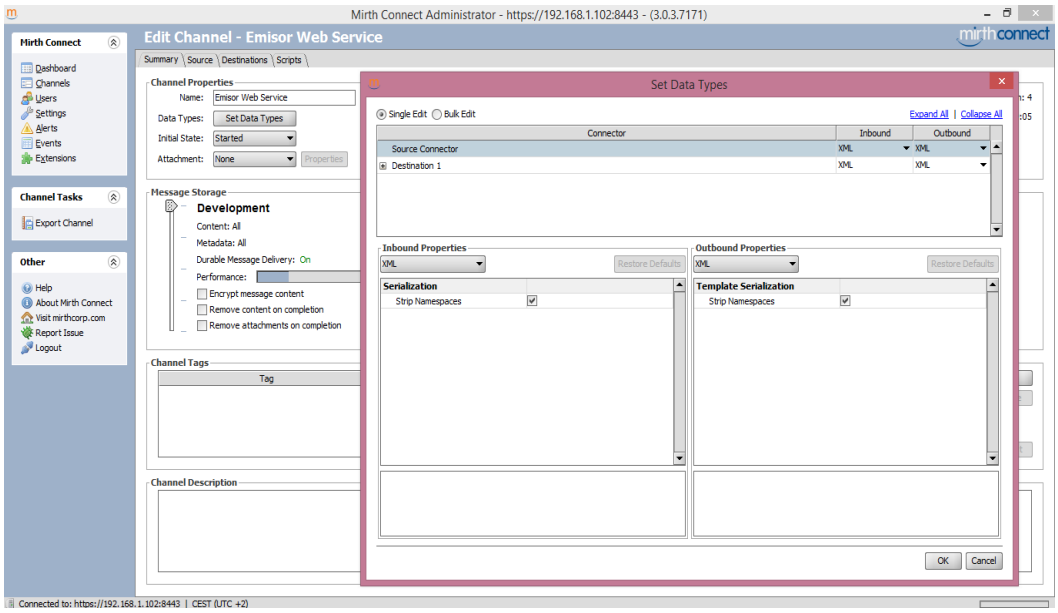


Ilustración 64.- Ventana de configuración del tipo de datos para los conectores de entrada y salida.

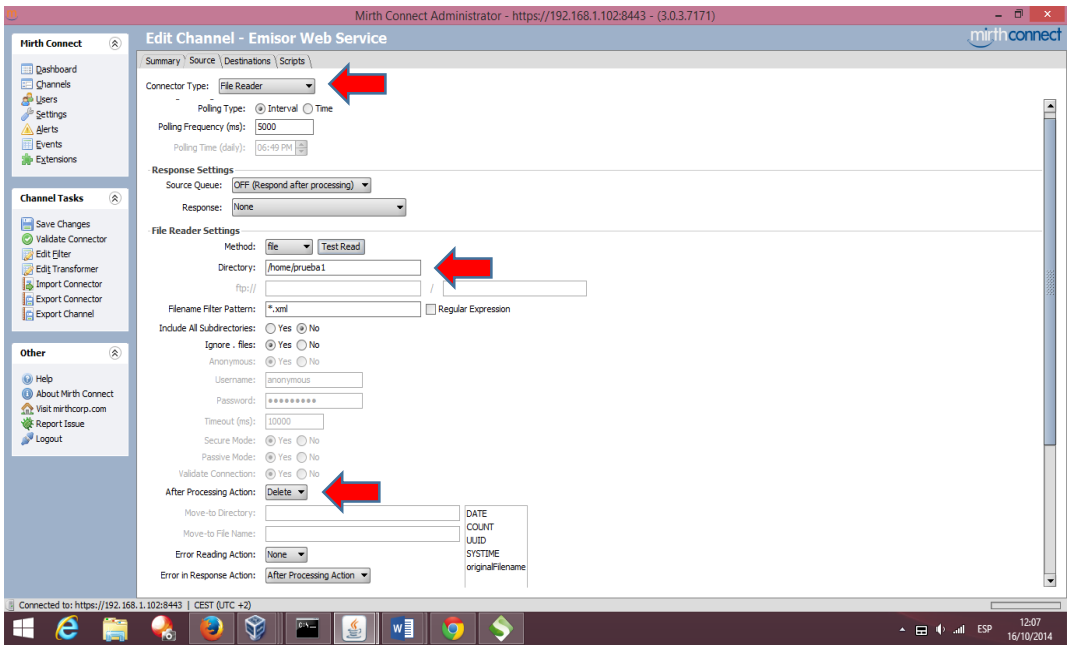


Ilustración 65.- Configuración del conector fuente para el canal Emisor Web Service.

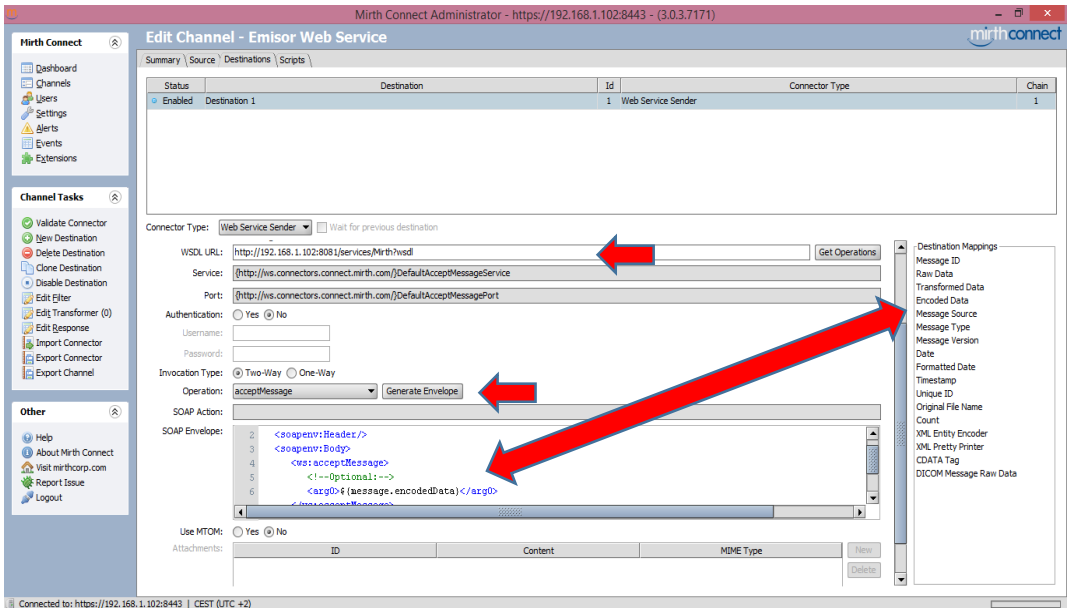


Ilustración 66.- Configuración del conector destino para el Canal Emisor Web Service.

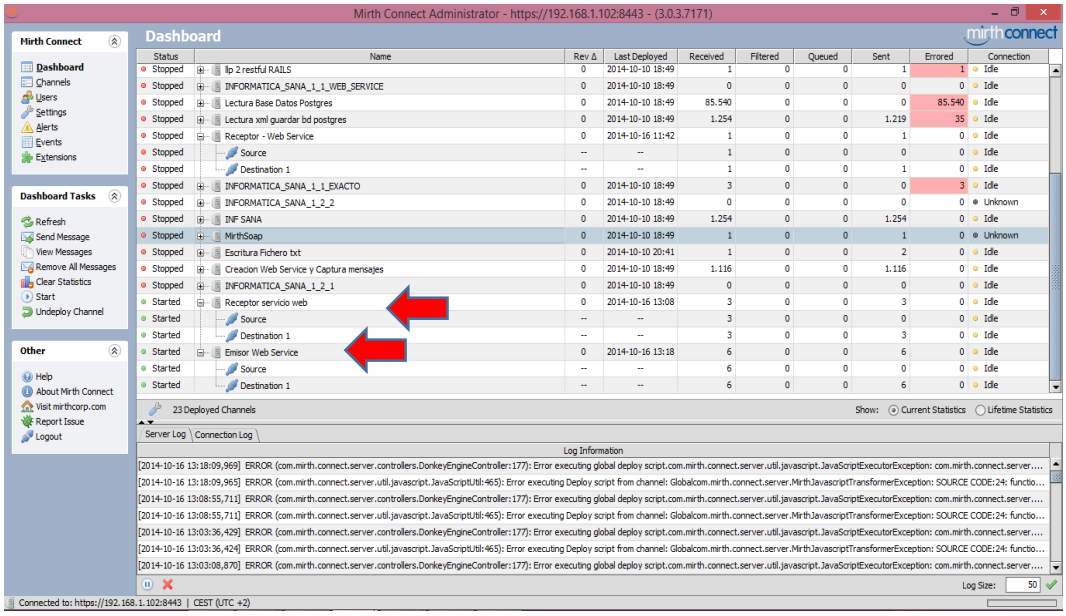


Ilustración 67.- Canales desplegados en el Dashboard.

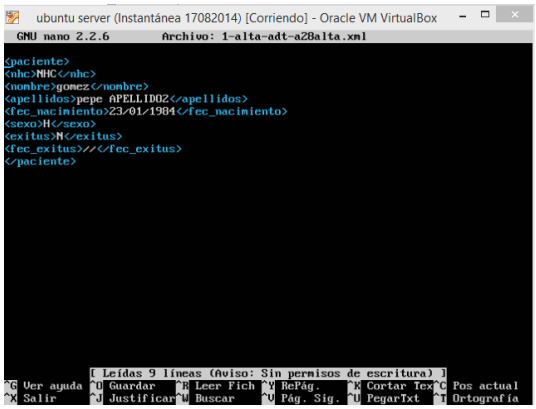


Ilustración 68.- Fichero XML a enviar.

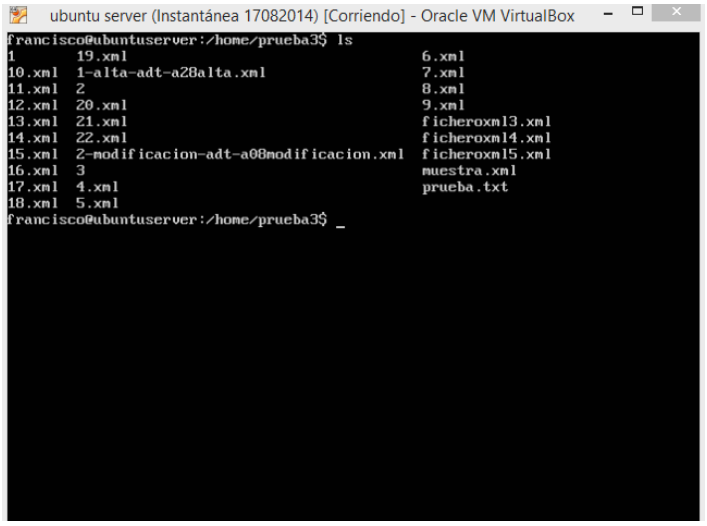


Ilustración 69.- Carpeta destinataria del fichero XML.

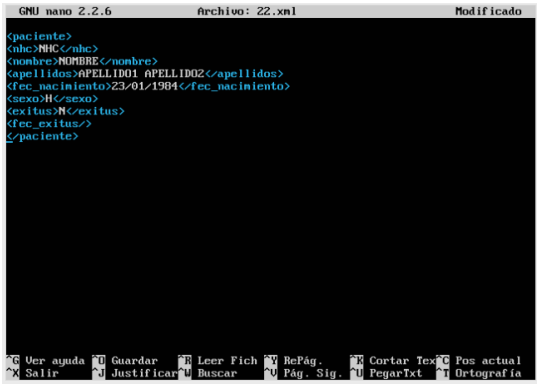


Ilustración 70.- Un posible fichero generado automáticamente.

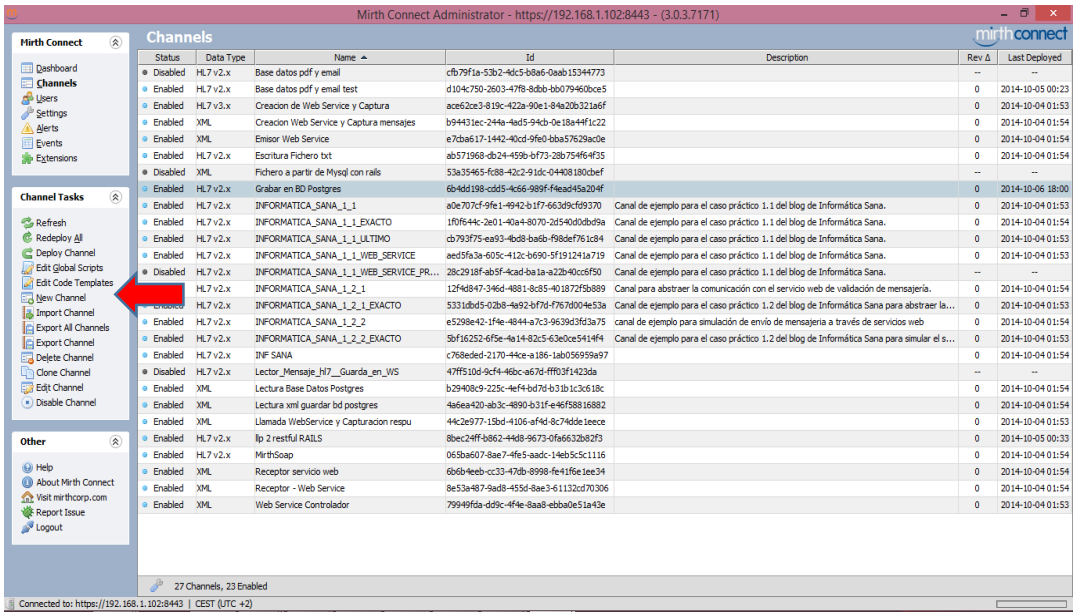


Ilustración 71.- Ventana de configuración de canales.

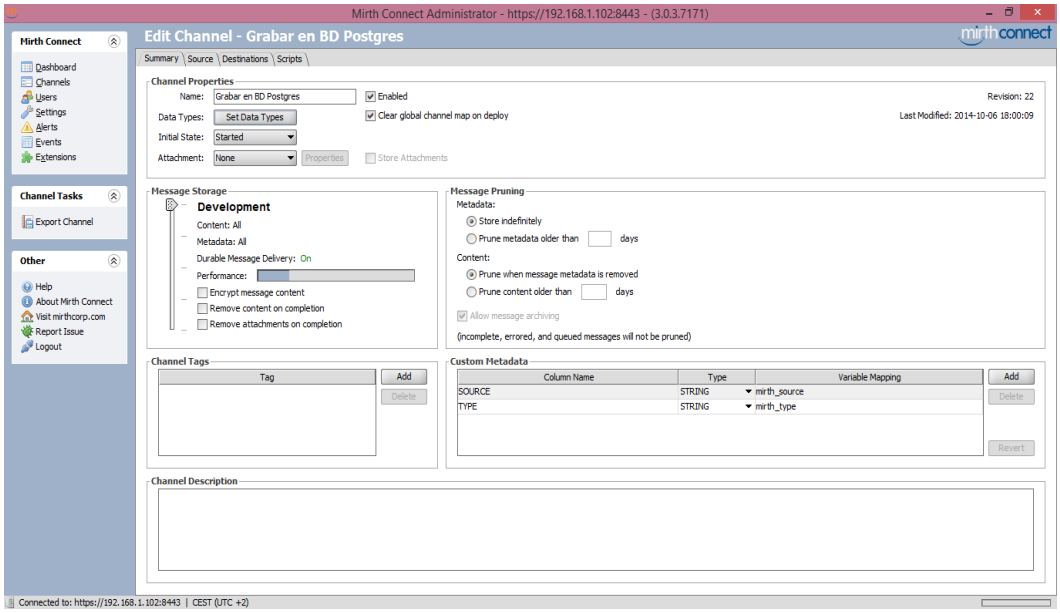


Ilustración 72.- Ventana Summary del canal Grabar en BD Postgres.

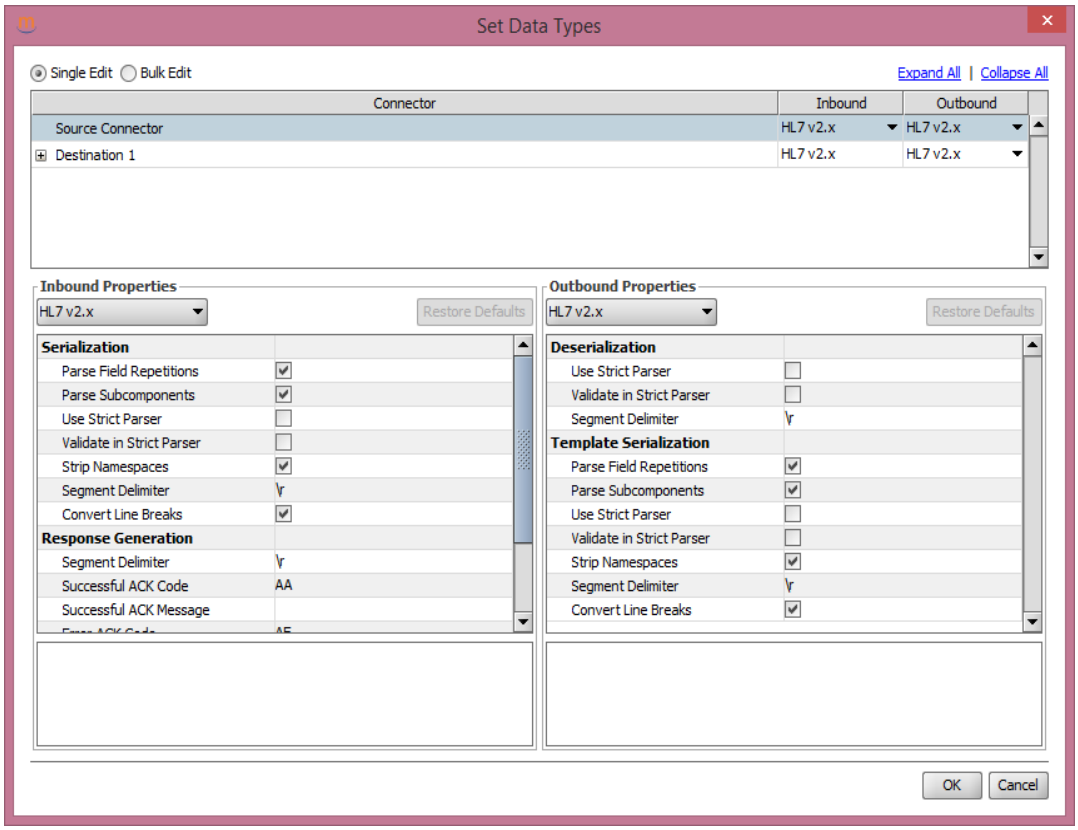


Ilustración 73.- Ventana para la configuración de los datos de entrada y salida.

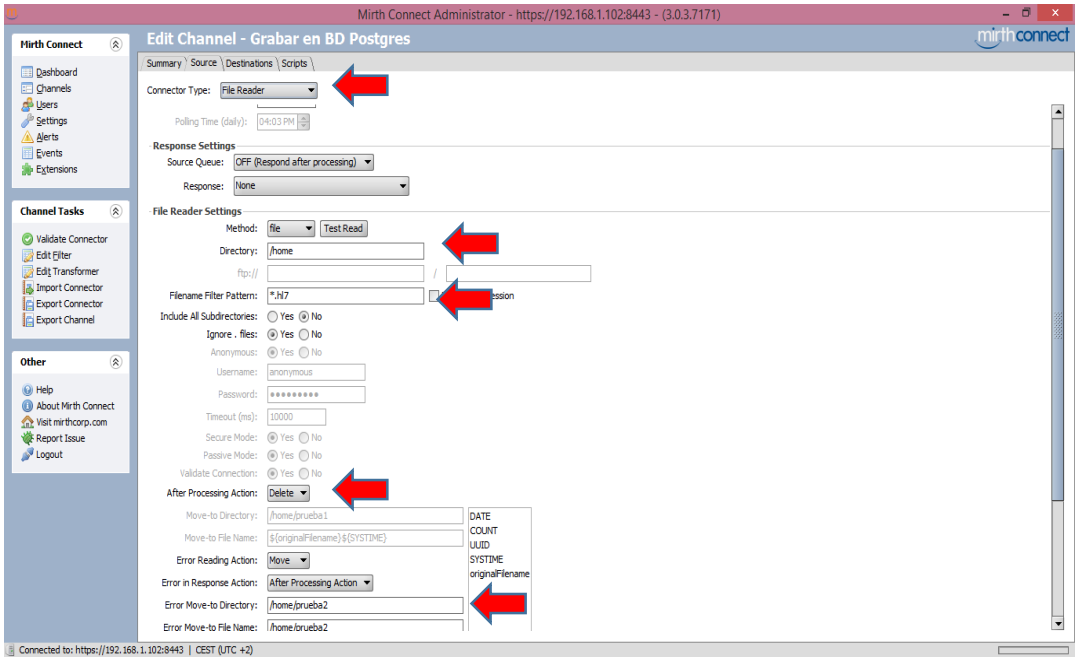


Ilustración 74.- Configuración del conector fuente del canal Grabar en BD PostgreSQL.

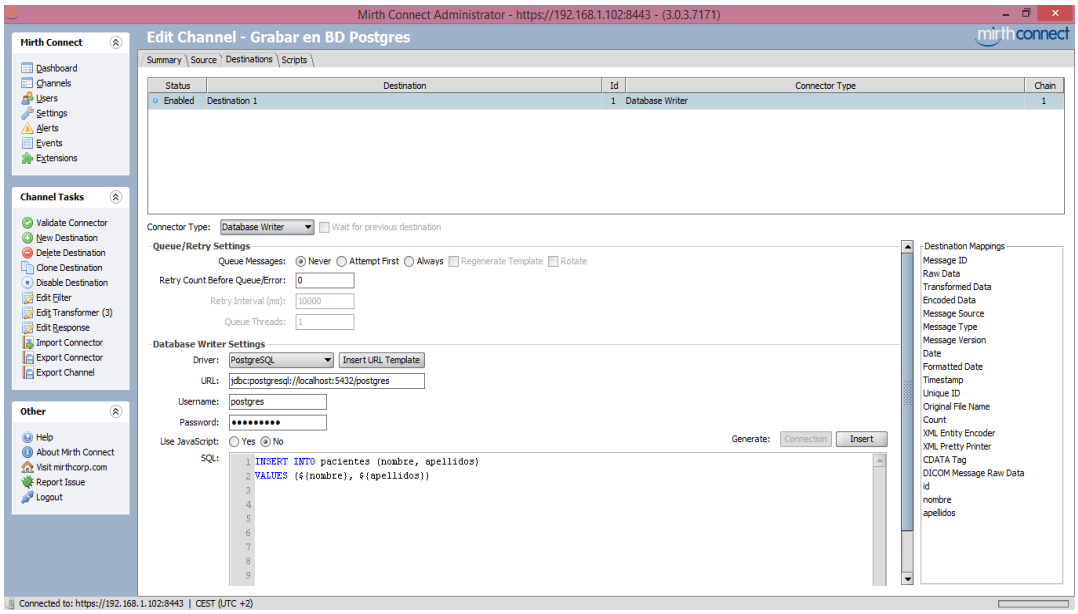


Ilustración 75.- Configuración conector destino del canal Grabar en BD PostgreSQL.

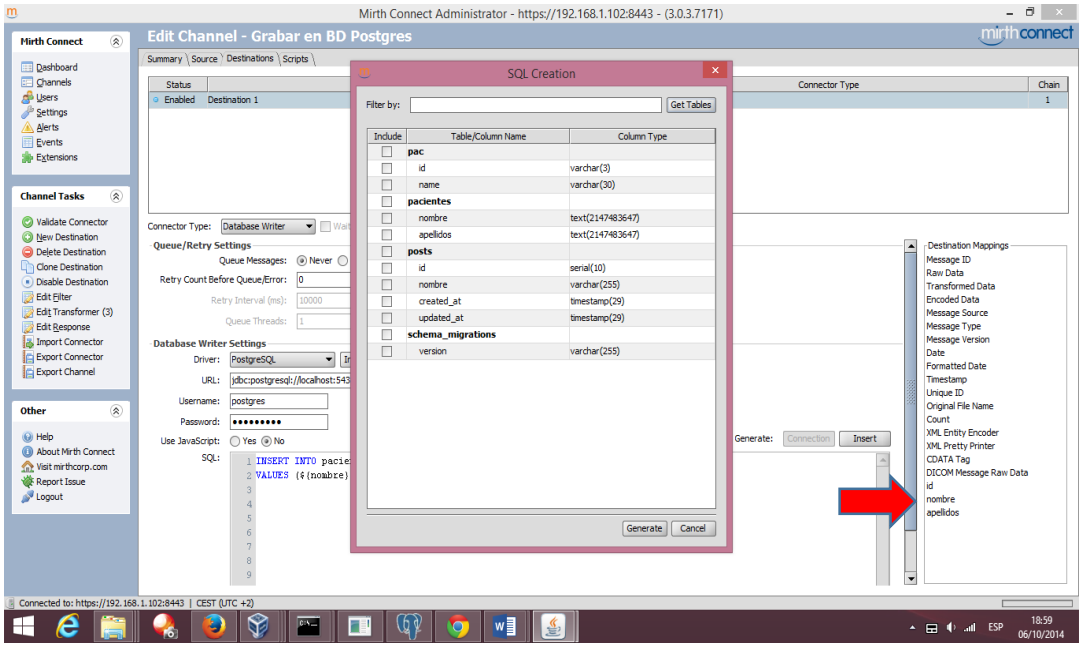


Ilustración 76.- Creación de la instrucción de inserción en la base de datos.

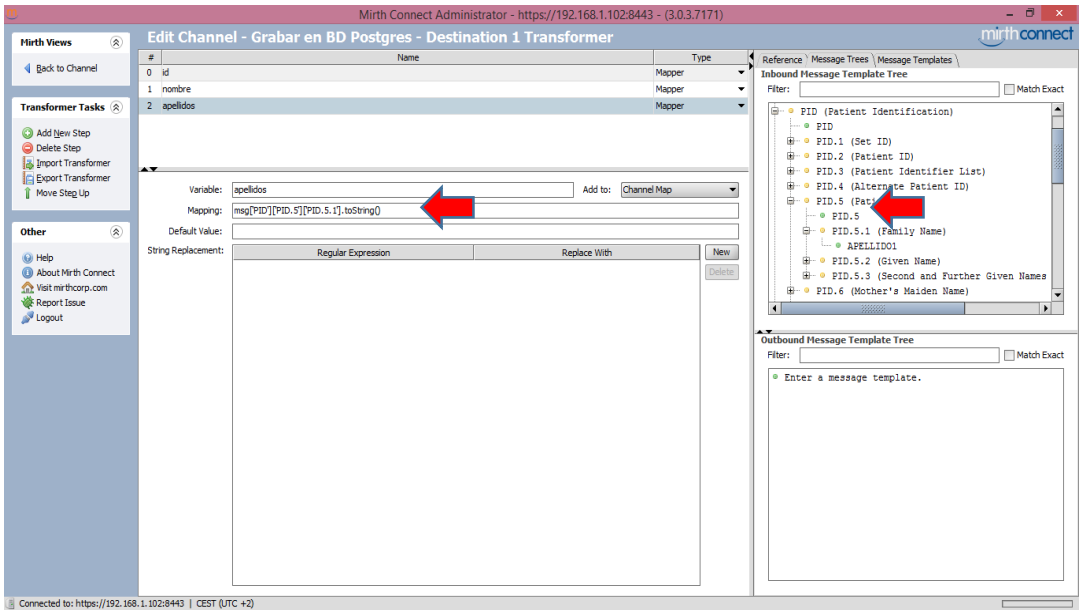


Ilustración 77.- Configuración de 3 variables con las instrucciones Javascript para obtener el valor.

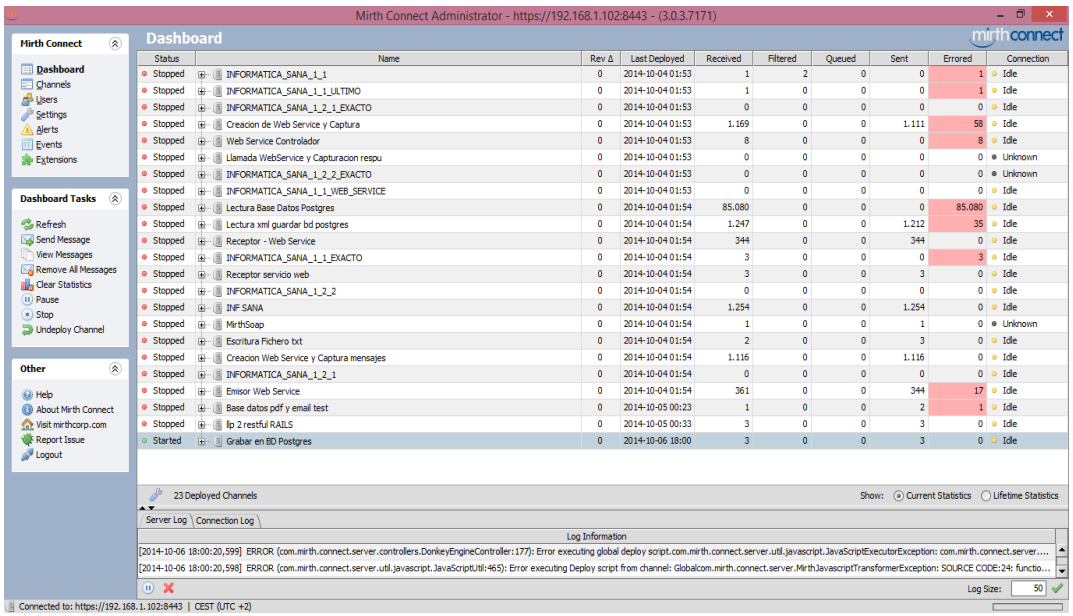


Ilustración 78.- Despliegue del canal Grabar en DB PostgreSQL.

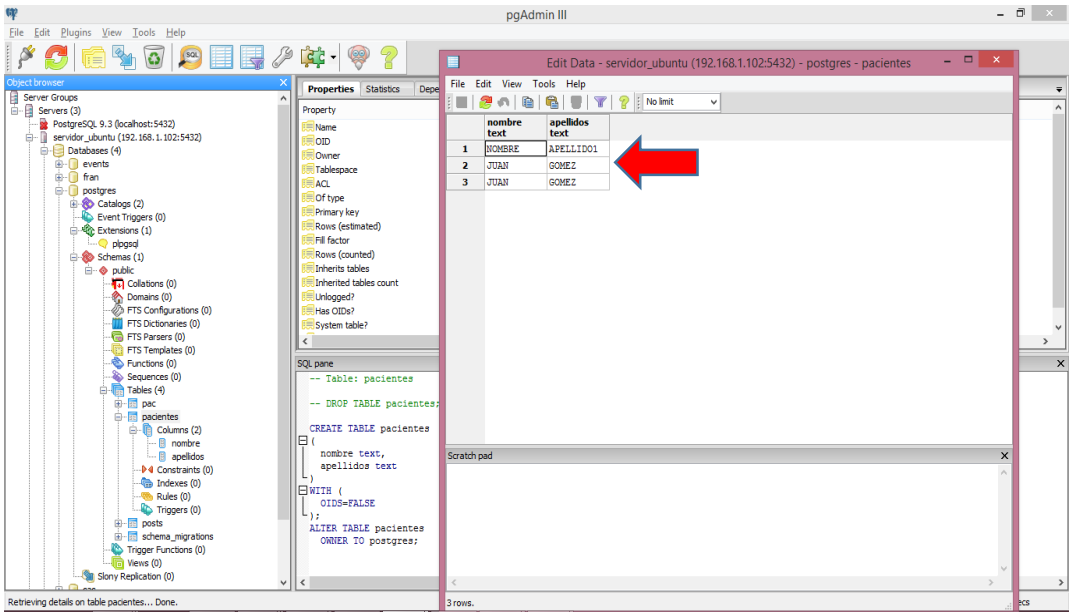


Ilustración 79.- Datos insertados automáticamente con la interacción de Mirth Connect.

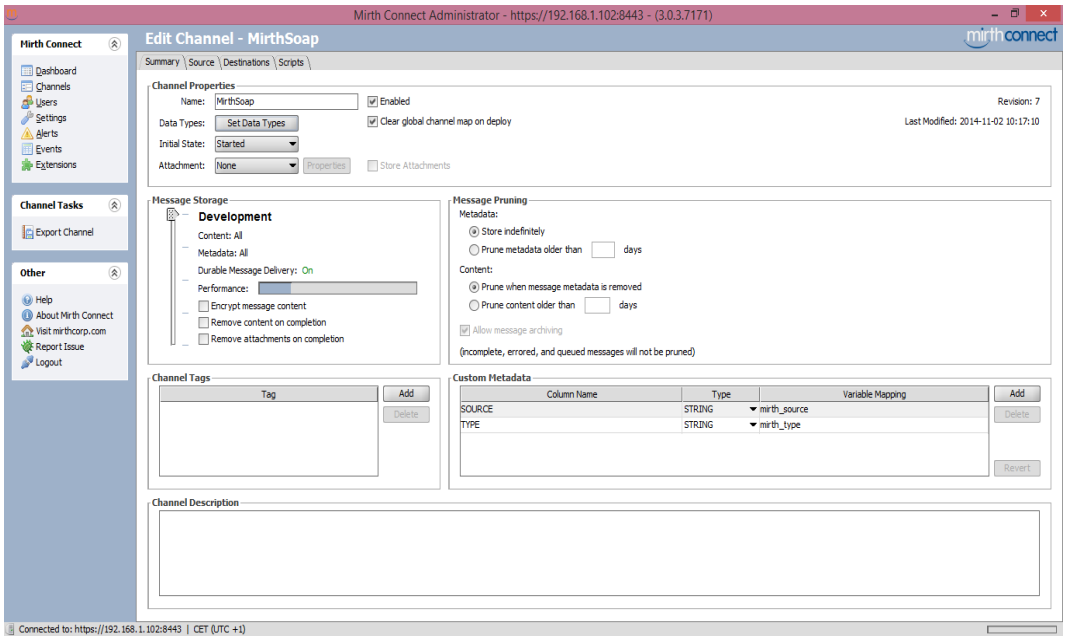


Ilustración 80.- Canal para el envío de mail.

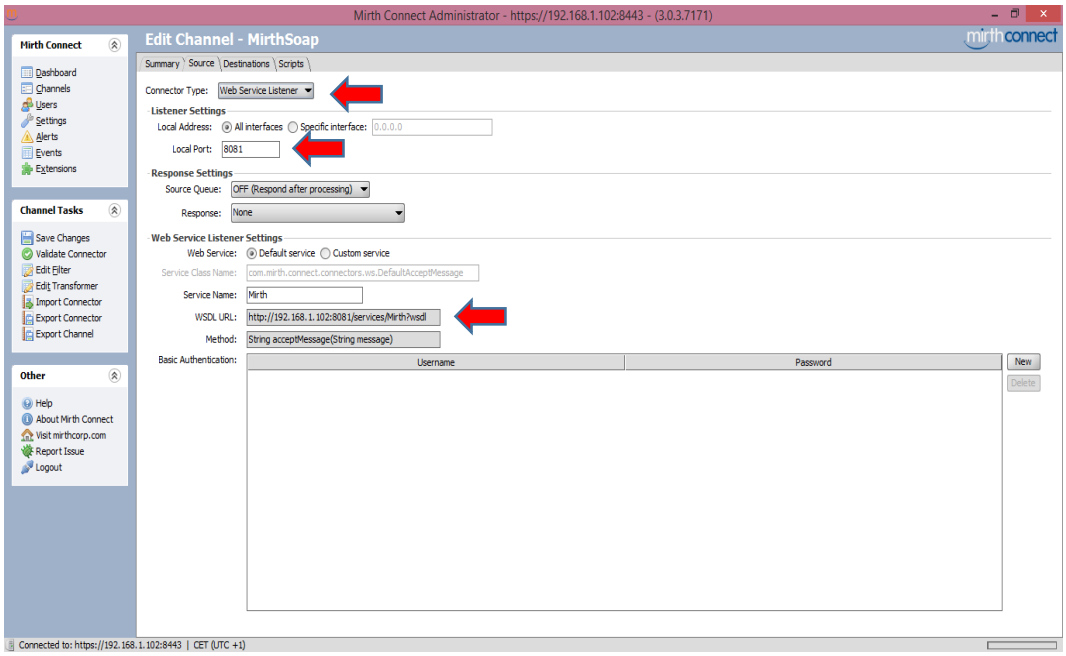


Ilustración 81.- Configuración del conector fuente del canal de envío de mail.

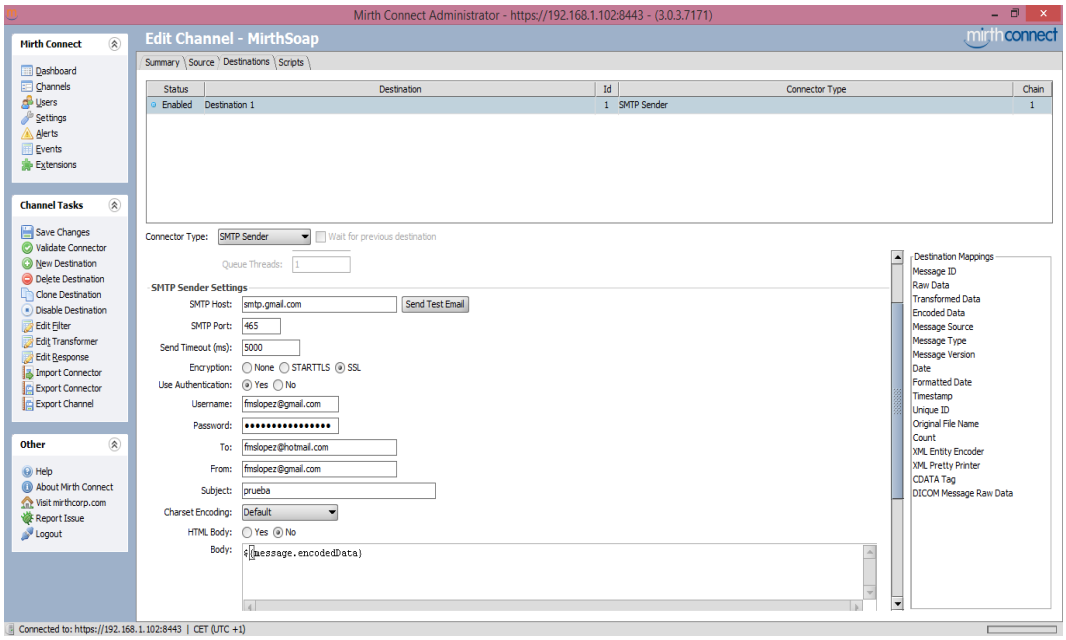


Ilustración 82.- Configuración del conector destino del canal MirthSoap encargado del envío de mail.

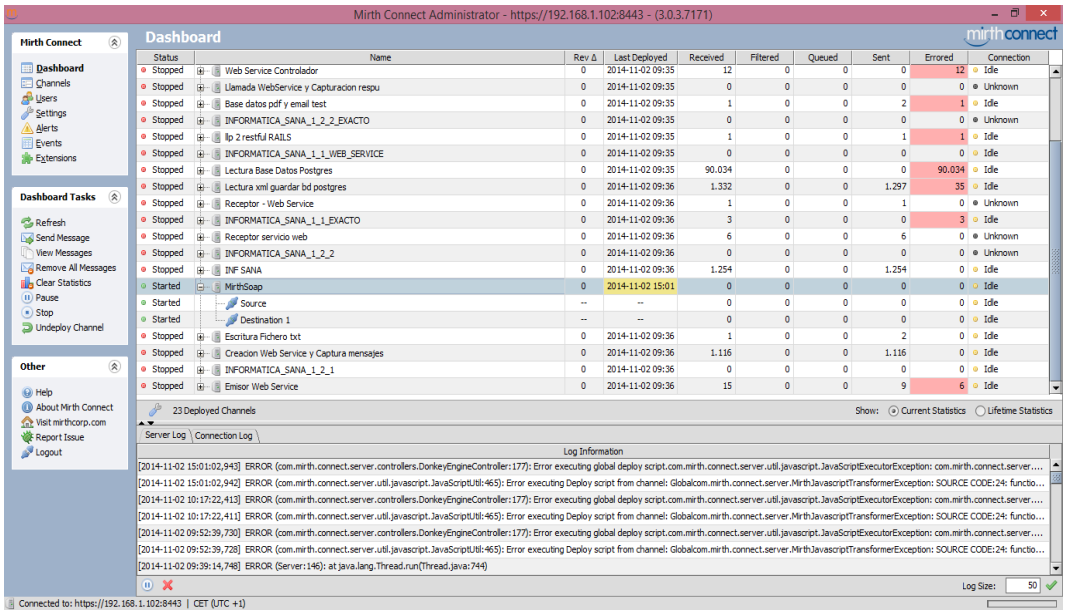


Ilustración 83.-Despliegue del canal MirthSoap para envío de mail y recepción de mensajería Servicio Web.

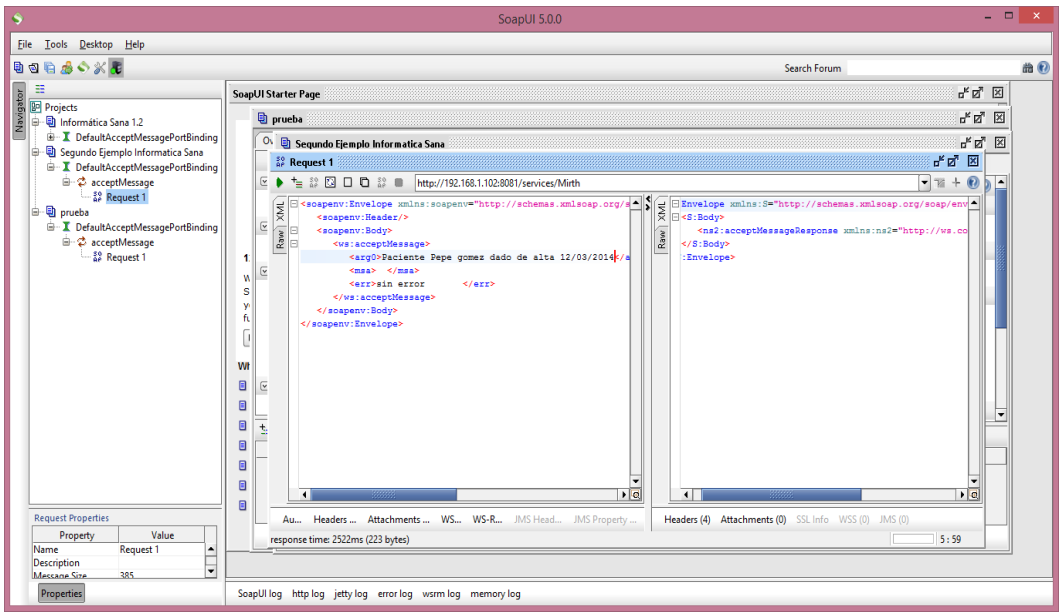


Ilustración 84.- Envío de la mensajería al canal por medio de la herramienta SoapUI.

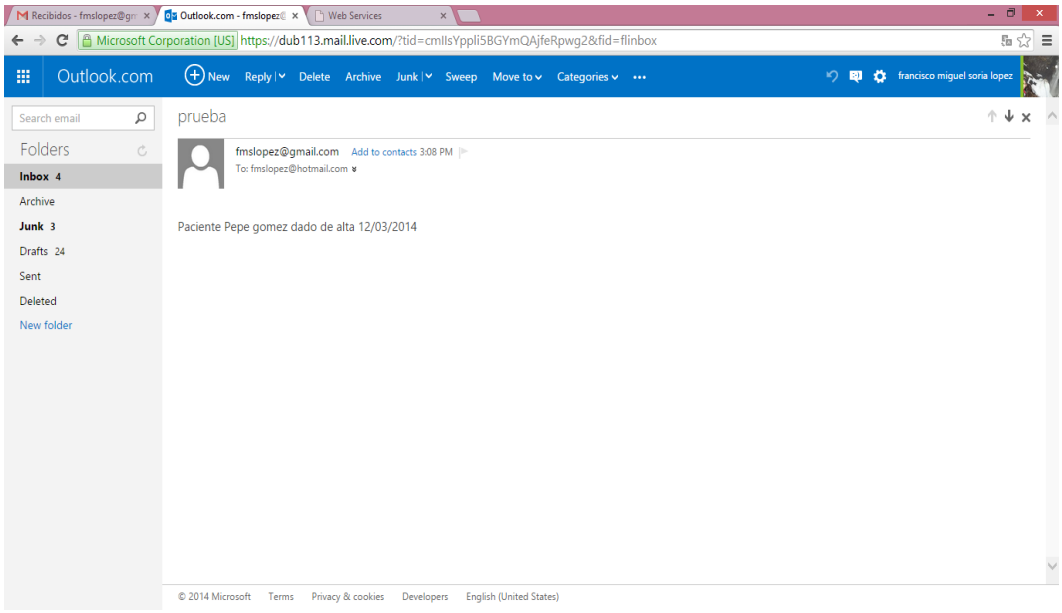


Ilustración 85.- Recepción del mensaje SOAP enviado desde el Web Service al correo.

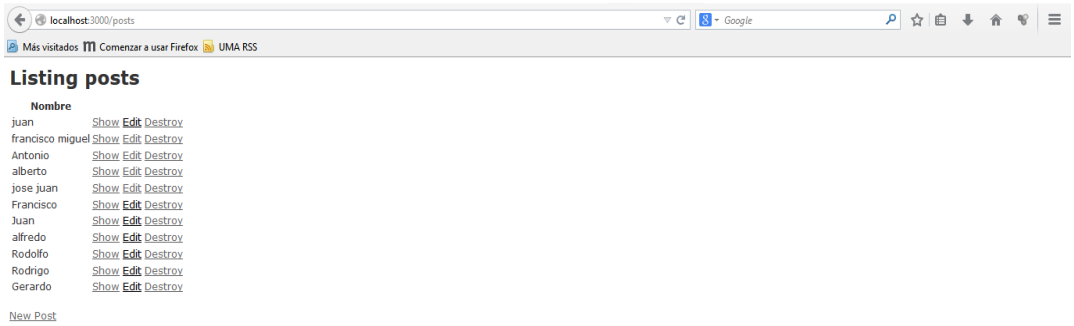


Ilustración 86.- Aplicación ruby on rails para uso de la B.D PostgreSQL.

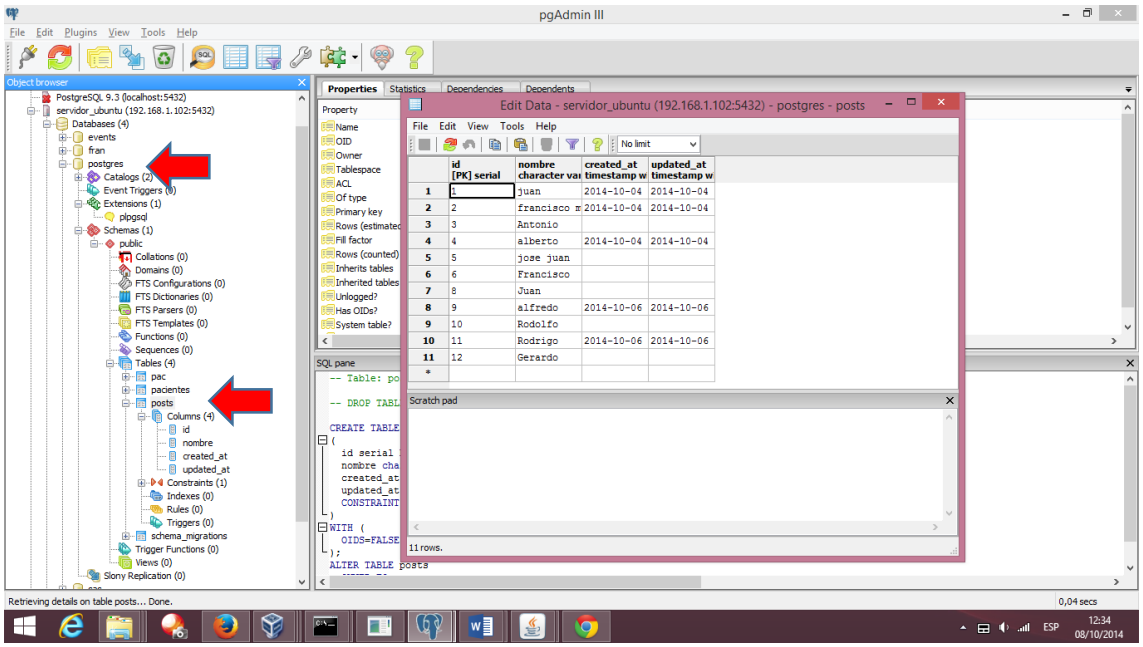


Ilustración 87.- Tabla posts de la base de datos PostgreSQL.

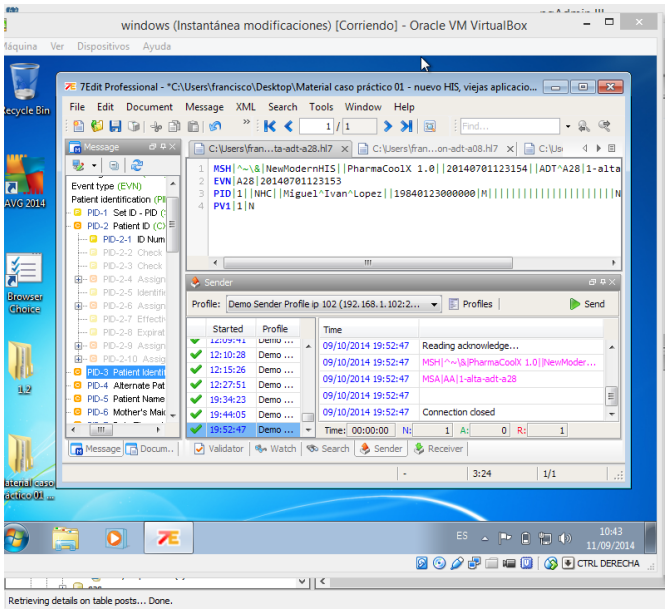


Ilustración 88.- Entorno Windows 7 donde se está ejecutando la herramienta 7Edit para la mensajería HL7.

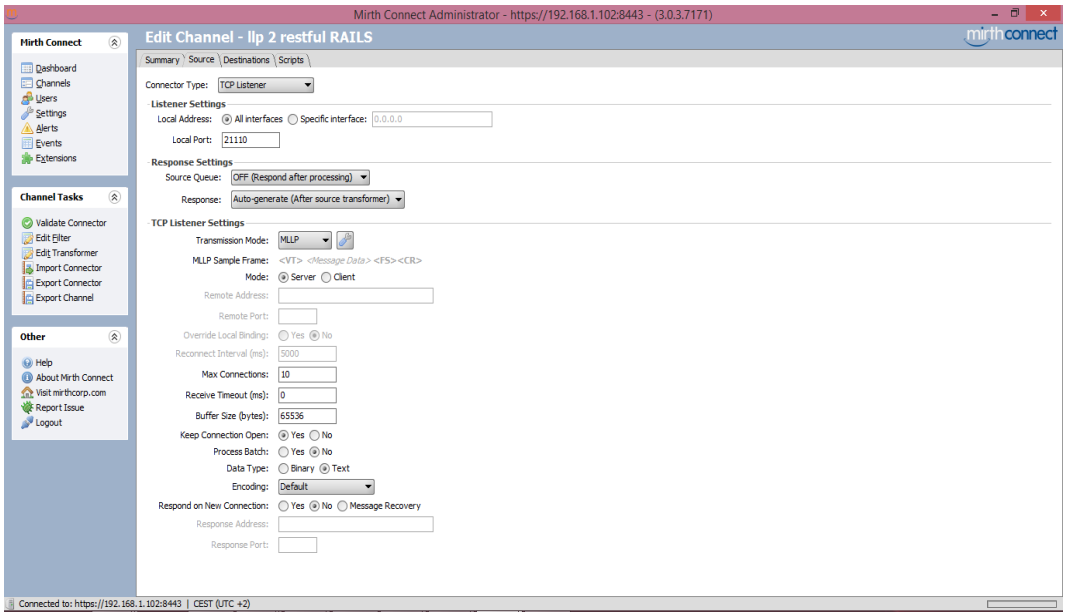


Ilustración 89.- Ventana Source del canal llp 2 restful RAILS.

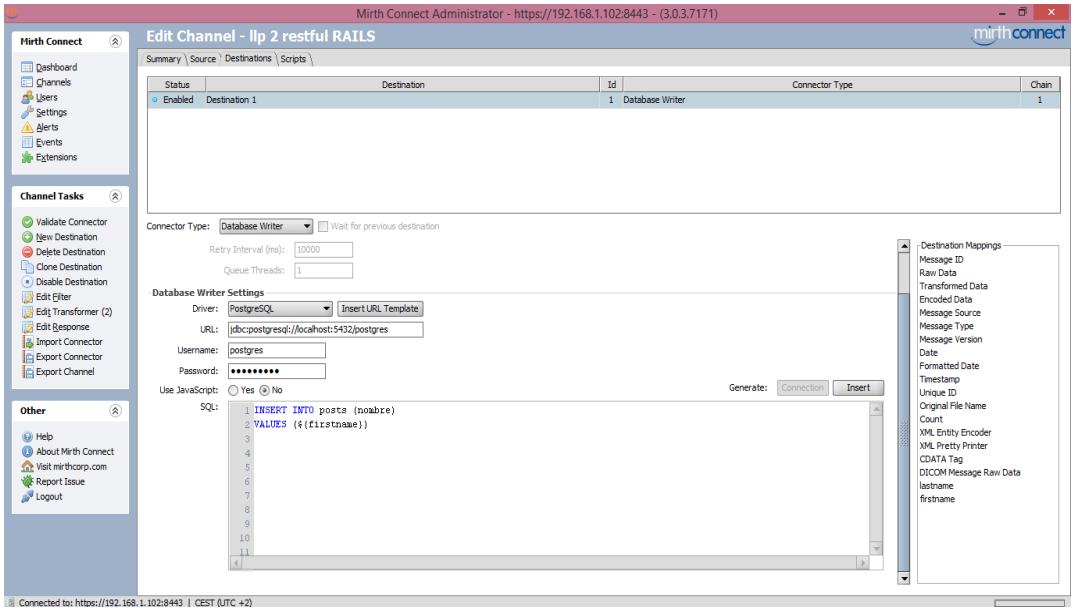


Ilustración 90.- Ventana Destinations del canal llp 2 restful RAILS.

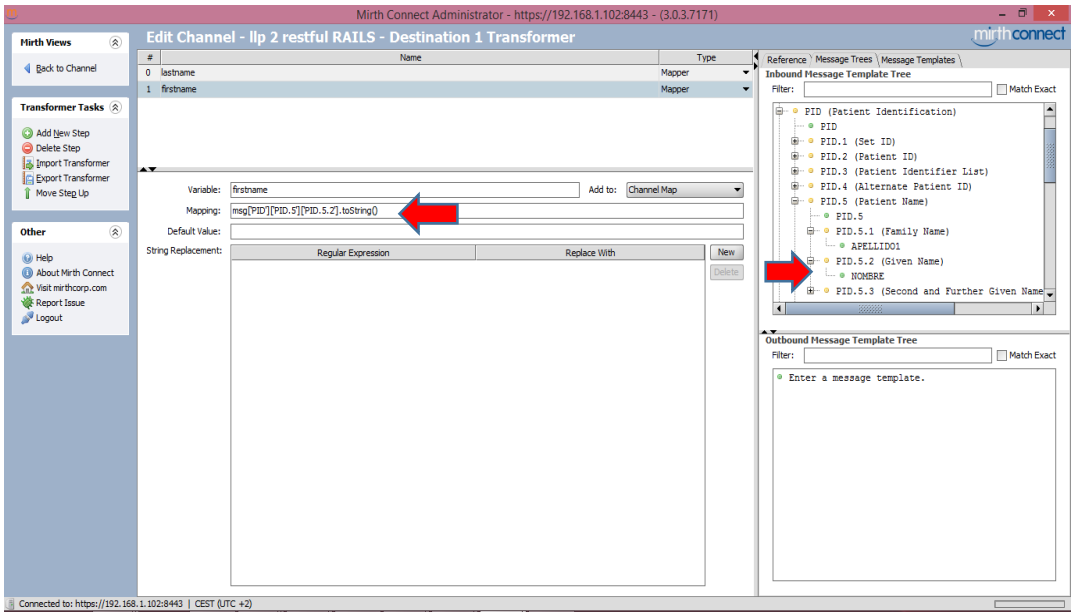


Ilustración 91.- Ventana de los Transformadores del canal llp 2 restful RAILS.