**UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS.**

**FACULTAD 7**


# PROPUESTA DE ARQUITECTURA DE UN SERVIDOR DE INTEGRACION PARA LA COMUNICACIÓN ENTRE SISTEMAS DE INFORMACION EN UN HOSPITAL.

**Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.**

**AUTOR(ES):**

Annia Garcia Ruiz

Yanitza Ramírez Stambor

**TUTOR**

Maykell Sánchez Romero

**ASESOR**

Renier Ricardo Figueredo

**Ciudad de La Habana**

**Junio, 2007**

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 7 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año 2007.

Yanitza Ramírez Stambor

_____

Firma del autor

Annia García Ruiz

_____

Firma del autor

Maykell Sánchez Romero

_____

Firma del tutor

# AGRADECIMIENTOS

**De Yanitza:**

En especial a mi mamá y a mi abuelita por guiarme siempre por el buen camino y confiar siempre en mí.

A mis hermanas, por su cariño y amor, gracias por estar siempre presente.

A mi familia, por haberme apoyado en todo momento.

A mi novio Aloy por estar siempre presente.

A mis amigas, Yuleydis, Indira, Lia, Nina, Karen, Yele, Anet, Alienys, por todos los momentos, buenos y malos, que hemos compartido juntas, por ayudarme y comprenderme siempre.

**De Annia:**

A mi mamá y mi papá, por estar siempre a mi lado, apoyarme, impulsarme a superarme profesionalmente, guiarme por el buen camino y quererme mucho.

A mi abuelita, por estar siempre dispuesta a ayudarme, por haberme cuidado y querido como mi propia madre.

A mi hermana, por haber podido siempre contar con ella.

A Pipa y Apa, por haber sido siempre como mis abluelos, enorgulleciéndose con cada uno de mis logros.

A mi novio Dennys, mis suegros Edda y Reinol, todos mis familiares y amigos, que de alguna manera han sido importantes en mi vida.

**Colectivos:**

A Lourdes, por apoyarnos en la realización de este trabajo.

A nuestro tutor Maykell por guiarnos durante el desarrollo de esta investigación.

A todos los profesores que de una forma u otra han contribuido a nuestra formación.

# DEDICATORIA

**De Yanitza:**

A mi abuelita Nieves, a mi mamá y a mis hermanas que siempre han estado a mi lado.

A Eriberto Samón, quien fue siempre mi padre y en este momento estuviera muy orgulloso de mi.

A toda mi familia por apoyarme siempre.

**De Annia:**

A mi mamá, mi papá, mi abuelita y mi hermana, que siempre me han apoyado y los quiero mucho.

A Pillillo, un gran amigo, que aunque ya no está, se hubiera sentido muy orgulloso.

En general a todas las personas que puedan sentir satisfacción con la realización de este trabajo.

## RESUMEN

En los últimos años, las nuevas Tecnologías de la Información y las Comunicaciones (TIC) han provocado un fuerte movimiento en las comunidades científicas de todo el mundo. Cuba no está exenta a este auge surgido en la nueva “era de la información”, por lo que ha identificado la necesidad de informatizar la sociedad cubana. Dentro del sector sanitario se han desarrollado múltiples aplicaciones encaminadas a mejorar los servicios que brindan las instalaciones hospitalarias de todo el país, pero la heterogénea naturaleza de las mismas no permite su integración.

En el presente trabajo se realiza una propuesta de arquitectura para un servidor de integración que permita comunicar los diferentes sistemas de información que existen en los hospitales mediante el uso del estándar de formato para las comunicaciones clínicas HL7.

Durante el desarrollo del trabajo se analizan las principales tendencias, tecnologías y metodologías actuales, se definen los principales requerimientos del sistema y se obtiene como resultado el diseño de la arquitectura de un servidor qeu permita la comunicacion netre sistemas en en hospital definiendo de esta forma los estilos arquitectónicos, patrones de arquitectura, lenguaje de programacióny gestor de base de datos, que deben ser usados en la futura implementación del servidor de integración.

# TABLA DE CONTENIDOS

# INTRODUCCIÓN

Con el fin de satisfacer las necesidades sociales, lograr más eficiencia en los procesos y aumentar la calidad de vida de los ciudadanos se ha introducido en nuestro país el concepto de "informatización de la sociedad", que constituye "la utilización ordenada y masiva de las tecnologías de la información y las comunicaciones en todas las esferas de la sociedad." [1]

El auge de la informática, surgido en la nueva "era de la información", permitió que los países del primer mundo pudieran tener un rápido acceso a las tecnologías, mientras que el resto de los países apenas daban los primeros pasos, pues no contaban con las tecnologías y personal calificado necesario para trabajar con ellas.

Es evidente que para los países subdesarrollados, lograr la informatización en todas las esferas de la sociedad, resulta un reto, ya que su problemática fundamental está en lograr la supervivencia de sus pueblos.

Una sociedad informatizada podrá ser más eficaz, eficiente y competitiva, por esta razón Cuba ha identificado desde muy temprano la necesidad y conveniencia de introducir en la práctica social las Tecnologías de la Información y las Comunicaciones (TIC). Una de las formas de encaminar a la sociedad hacia el objetivo de un desarrollo sostenible es crear en el hombre nuevo un dominio de la cultura digital, es por ello que el pueblo cubano enfrenta hoy la difícil tarea de informatizar la sociedad.

La salud pública cubana se ha convertido en referencia para muchos países del mundo por el desarrollo y los grandes logros alcanzados durante los 48 años de Revolución, por lo tanto, la informatización del sector sanitario es una de las tareas más importantes y complejas a las que se enfrentan hoy los profesionales de la informática en Cuba.

En el mundo las instituciones de medicina poseen aplicaciones informáticas que se ocupan del registro de los procesos de admisión y egreso de pacientes, del registro y producción de información de laboratorio clínico, de informes de radiología y patología, de la facturación y administración general y otros.

A menudo estas aplicaciones han sido desarrolladas por diferentes proveedores, poseyendo cada producto diferentes formatos de los datos. En la medida en que los hospitales han ido expandiéndose, las operaciones de procesamiento de información han ido aumentando en forma concomitante, y la necesidad de compartir los datos que encierran esa información se torna crítica.

El desarrollo y disponibilidad de sistemas globales de informatización hospitalaria, que solo algunos proveedores muy selectos han desarrollado hasta la fecha, mitigarían la necesidad de estándares para la transmisión externa de datos del tipo del HL7. No obstante, estos programas son todavía escasos y su implementación requiere fuertes inversiones iniciales tanto de hardware como de software, lo cual hace que sigan desarrollándose las aplicaciones específicas de bajo costo.

Por otro lado, las instituciones hospitalarias aún son el objeto de fuertes presiones en el sentido de la adquisición o el desarrollo de aplicaciones departamentales con un criterio modular. Este tipo de presiones, surgen por ciertas necesidades departamentales de procesamiento de información no adecuadamente resueltas por los sistemas "globales".

Otra fuente de presiones, es la necesidad de desarrollar un sistema finalmente global a través de sucesivas etapas, atendiendo a necesidades departamentales según un esquema de prioridades, y no a efectuar una sola adquisición brusca y revolucionaria.

“La necesidad del desarrollo de estándares que sirvan de base a interfases entre sistemas surge a partir de la aparición de la tecnología de redes para la integración de programas de aplicación al área de salud que residen en computadoras funcional y técnicamente diferentes. (…) Lograr su integración suele requerir enorme cantidad de horas de programación específicas al sitio y al ambiente de redes. Esto ocurre a expensas del comprador y/o el vendedor, e impide al personal involucrado dedicarse a iniciativas más productivas como el desarrollo de nuevos módulos o productos. La cantidad de sistemas a interfasear aumenta en forma exponencial la cantidad de tiempo invertido en el desarrollo de interfases, en tanto que el apego a un estándar solo requiere del esfuerzo de su desarrollo por única vez.” [2]

La informatización del Sistema Nacional de Salud Pública (SNS) en Cuba está dada por el “conjunto de métodos, técnicas, procederes y actividades gerenciales dirigidas al manejo de la información en salud, la cual comprende la información sobre el estado de salud de la población, la información sobre el

conocimiento de las ciencias de la salud y la información en general para la toma de decisiones, clínico-epidemiológicas, operativas y estratégicas." [3]

Durante las últimas dos décadas un grupo de instituciones cubanas se han dedicado a desarrollar sistemas encaminados a lograr ciertos niveles de informatización de la salud. Dichos sistemas carecían de integración, además de que no existían los recursos tecnológicos necesarios para su ejecución en el Sistema Nacional de Salud (SNS).

Una de las prioridades del Ministerio de Salud Pública (MINSAP) se ha definido como "informatización". Para ello ha convocado a un grupo de instituciones propias del sector, del Ministerio de la Informática y las Comunicaciones (MIC) y de otros organismos de la administración central del estado, que de conjunto tienen la misión de definir la estrategia a desarrollar para informatizar la salud cubana.

La única forma que permite hablar de informatización de la salud pública es concibiendo y desarrollando los proyectos de forma integrada, nunca como proyectos aislados. Contar con la integración de los datos generados en los distintos niveles de salud donde puede ser atendido un paciente permitirá alcanzar, por etapas, la informatización de la salud pública cubana.

De esta manera será posible perfeccionar la calidad asistencial ofrecida a la sociedad, facilitar las funciones del personal de la salud y colaborar con la gestión administrativa, asistencial, docente y de investigación.

Para el MINSAP se han desarrollado diferentes sistemas informatizados, la heterogénea naturaleza de los mismos y la poca o ninguna observación de estándares permite que no exista integración entre ellos, por lo que la información no fluye de igual manera en los distintos niveles, provocando la existencia de información redundante en los datos del paciente, pérdida de información y desactualización.

Dentro los sistemas mencionados anteriormente se encuentran, por ejemplo, los RIS (Sistemas de Información Radiológica, por sus siglas en inglés), que normalmente, a falta de la existencia de comunicación con otros sistemas, como el Sistema de Información Hospitalario (HIS), se ocupa de realizar la recogida de datos del paciente para la realización del estudio, redundando en información ya existente.

En la misma medida, una vez que se han realizado e informado los estudios, es imposible acceder desde otras áreas del hospital, algo no deseable en todo hospital informatizado, pues los resultados deben ser leídos rápidamente para realizar un diagnóstico temprano.

En estos momentos, un grupo de diferentes empresas como Desoft, Softel, Infomed, la Universidad de las Ciencias Informáticas (UCI), etc, de conjunto con las Direcciones Nacionales del Ministerio de Salud Pública implicadas directamente en los primeros productos, trabajan en el desarrollo e implementación de un grupo de aplicaciones que son básicas para la informatización del sector de la salud.

El **problema** principal radica en ¿cómo lograr la comunicación entre los sistemas de información existentes en un hospital?

La solución a este problema debe tener en cuenta el uso de estándares, pues "si no hay formas fiables y aprobadas para conectar los componentes necesarios, dichos sistemas no pueden funcionar" [4].

El **objeto de la investigación** se enmarca en la integración de sistemas de información sanitarios.

El **campo de acción** está centrado en la integración de los sistemas de información existentes en un hospital.

Para dar solución al problema anterior se propone como **objetivo general:**

Diseñar la arquitectura de un servidor de integración que permita la comunicación entre los sistemas de información existentes en un hospital.

Para cumplir el mismo se han propuesto como **tareas** fundamentales**:**

- Analizar bibliografía referente al tema de la integración de sistemas sanitarios.
- Realizar un estudio sobre IHE como iniciativa de integración de las aplicaciones sanitarias.
- Realizar un estudio sobre HL7 como estándar de formato para las comunicaciones clínicas.
- Evaluar las tecnologías y tendencias actuales para seleccionar las que más se adecue a las necesidades del servidor de integración.

- Fundamentar los patrones de arquitectura a utilizar, así como su aplicación en la solución del problema.

- Diseñar la arquitectura de un servidor de integración que permita la comunicación entre los sistemas de información existentes en un hospital.

El contenido del documento está estructurado en dos capítulos:

El Capítulo 1. Fundamentación teórica: donde se realiza un estudio detallado de las tendencias, tecnologías y metodologías actuales con el fin de proponer las más adecuadas para la solución del problema. Además se da a conocer el estado del arte referente a la integración de sistemas.

El Capítulo 2. Descripción de la arquitectura: en este capítulo se realiza la propuesta de la arquitectura para el servidor de integración, teniendo en cuenta plataforma, lenguaje de programación, patrones de arquitectura, requisitos del sistema, etc.

Finalmente se exponen las conclusiones del trabajo y se hace referencia de la bibliografía utilizada en la investigación.

# CAPITULO 1.Fundamentación teórica

En este capítulo se realizará un estudio detallado de los principales conceptos relacionados con la problemática, así como de las tendencias, tecnologías y metodologías actuales, con el fin de proponer las más adecuadas a la solución del problema. El mismo tiene como propósito dar a conocer el estado del arte correspondiente a la integración de los sistemas de información sanitarios en Cuba y el resto del mundo.

## 1.1 Conceptos Generales

### Sistema de Información

Un sistema de información se puede definir como un conjunto de funciones o componentes interrelacionados que forman un todo, o sea, obtiene, procesa, almacena y distribuye información para apoyar la toma de decisiones y el control en una organización.

### Servidor de Integración

La función principal de un servidor de integración es recoger los mensajes de los diferentes sistemas, validarlos y entregarlos al sistema destino. Para ello debe disponer de diferentes conectores o adaptadores que permitan la integración con las diferentes aplicaciones.

El servidor de integración también debe ser capaz de enrutar los mensajes, es decir, decidir a que sistema va dirigido un mensaje determinado, según la información proporcionada por el emisor.

Los conectores o adaptadores permiten conectar aplicaciones a un servidor de integración. Existen dos tipos de adaptadores; adaptadores de aplicación que permiten integrar aplicaciones de negocio y utilizan API y los adaptadores de tecnologías que permiten la integración de aplicaciones a nivel de plataforma, que se utilizan cuando el destino es un sistema y no una aplicación, o para aplicaciones que no publican una interfaz para su integración.

**Arquitectura de Software**

La arquitectura de software, denominada también arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La arquitectura de software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información.

"La arquitectura software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Toda arquitectura software debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea de computación".[5]

## 1.2 Experiencias anteriores vinculadas al problema

**Ámbito Internacional**

En los últimos años ha existido un especial interés en las soluciones informáticas en el sector de la salud por parte de la comunidad científica a nivel internacional, encaminadas a lograr la integración entre las diferentes aplicaciones. A continuación se muestran algunos ejemplos:

El Hospital Sant Joan de Déu (San Juan de Dios, España) ha integrado las aplicaciones de los departamentos ya existentes como Radiología, Tráfico de Pacientes y Laboratorio con la Historia Clínica Electrónica, gracias a la tecnología de Software AG basada en el estándar XML-HL7.

Especial interés por mejorar la comunicación entre los sistemas de información que se utilizan en la atención al paciente, ha mostrado IHE.

"Integrando las Empresas Sanitarias" es una iniciativa de profesionales de la sanidad, es una organización creada a nivel internacional, que abarca tres grandes áreas: Estados Unidos, Europa y Asia-Pacífico; y desde sus inicios se trazó como objetivos principales:

- Acelerar la adopción de soluciones de integración basadas en estándares.

- Promover la comunicación y cooperación entre la industria y los proveedores de servicios sanitarios.
- Mejorar la disponibilidad de información médica para la toma de decisiones en la atención al paciente.
- Mejorar la eficiencia y eficacia de la práctica clínica.

Son varios los proveedores que ya han contemplado dentro de sus soluciones el uso del estándar HL7. Por ejemplo, Software AG España es una empresa que ofrece las más avanzadas tecnologías y soluciones en arquitectura orientada a servicios (SOA), y se ha consolidado como una de las empresas más innovadoras del mercado español. Dicha empresa trabaja actualmente en la creación de la Historia Clínica Única del Servicio Gallego de Salud (SERGAS).

Software AG promueve el uso de XML/HL7, un estándar basado en XML específico para el sector sanitario, que hace posible la interoperabilidad entre los sistemas informáticos de los distintos servicios de salud. De esta forma, cualquier médico o ciudadano tiene la posibilidad de acceder a su Historia Clínica Única si necesita de atención especializada o requiere de una intervención quirúrgica fuera de su Comunidad.

Una experiencia útil a analizar es el desarrollo de Biz Talk Server: un servidor para la construcción de soluciones de proceso de negocio e integración. Aprovecha las últimas tecnologías de Microsoft y los estándares de la industria para conseguir una potente herramienta que permita automatizar y gestionar procesos de negocio. Actualmente más de 5.000 empresas ya conocen el valor y la agilidad de BizTalk Server.

**Ámbito nacional**

En Cuba también se están haciendo grandes esfuerzos por lograr la integración de las aplicaciones sanitarias en la medida en que se vayan creando.

Existen aplicaciones tales como el “GALEN Hospital”, que está siendo utilizado actualmente en el Hospital “Hermanos Almeijeiras” para la inscripción, registro de los pacientes y movimiento hospitalario, que cuenta además, con módulos como Farmacia, Urgencias, Banco de Sangre, etc. Esta aplicación cuenta con la integración de todos sus módulos, teniendo en cuenta que son parte de una misma aplicación, pero en la actualidad no tiene concebida la integración con otros sistemas.

Recientemente se creó el Grupo de Arquitectura MINSAP-MIC, que tiene como misión revisar los sistemas para tratar de lograr un nivel de integración que permita la llegada de información a niveles superiores en tiempo, que no se maneje información duplicada, etc.

El problema radica en que muchos fabricantes diseñan sus propios formatos y protocolos de comunicación para que sólo sean manejables por los programas que ellos mismos venden con lo que la información de los usuarios "queda prisionera".

Los esfuerzos que se realicen deben tener en cuenta que la integración entre aplicaciones heterogéneas no es posible sin el uso adecuado de estándares.

El uso de estándares abiertos evita los problemas de comunicación entre las organizaciones por la utilización de diferentes formatos, facilita el intercambio de documentos y el trabajo compartido entre sistemas informáticos heterogéneos.

## 1.3 Tendencias, tecnologías y metodologías actuales

### 1.3.1 Metodologías de desarrollo de software

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

No existe una metodología de software universal. Cada equipo de desarrollo escoge la metodología según las características de su proyecto, por lo que es importante determinar el alcance del proyecto antes de escoger la metodología que se va a usar en el desarrollo del mismo.

A continuacion se mencionan algunas metodologías de desarrollo que existen, dando una expilcación sobre las mismas.

**Extreme Programming (XP)**

XP (Extreme Programming, por sus siglas en inglés) es una metodología de desarrollo de software, de las más exitosas en la actualidad utilizadas para proyectos de corto plazo y corto equipo. La metodología consiste en una programación rápida o extrema, donde el usuario final forma parte del equipo.

**Microsoft Solution Framework (MSF)**

MSF (Microsoft Solution Framework, por sus siglas en inglés) es una metodología de desarrollo de software que controla la planificación, el desarrollo y la gestión de proyectos tecnológicos.

MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Puede ser adaptada a proyectos de cualquier dimensión y usada para desarrollar soluciones sobre cualquier tecnología.

**Rational Unified Process (RUP)**

RUP (Rational Unified Process, por sus siglas en inglés) es un proceso de desarrollo de software y junto con UML (Lenguaje de Modelado Unificado, por sus siglas en inglés) constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP divide su ciclo de desarrollo en cuatro fases y ha agrupado sus actividades en 9 flujos de trabajo.

Fases.

- **Inicio**: Esta fase tiene como objetivo determinar la visión del proyecto.
- **Elaboración**: En esta etapa el objetivo es definir la arquitectura del sistema.
- **Construcción**: En esta etapa el objetivo es llegar a obtener un producto listo para su utilización que está documentado y tiene un manual de usuario.
- **Transición**: El objetivo es llegar a obtener el *release* ya listo para su instalación. Puede implicar reparación de errores.

Flujos de Trabajo.

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

- **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.

- **Instalación:** Produce *release* del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.

- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Cada flujo de trabajo tiene como resultado un modelo propuesto por RUP:

- Modelo de Casos de Uso del Negocio.
- Modelo de Objetos del Negocio.
- Modelo de Casos de Uso.
- Modelo de Diseño.
- Modelo de Despliegue.

- Modelo de Datos.
- Modelo de Implementación.
- Modelo de Pruebas.

El modelo de casos de uso, el modelo de diseño, el modelo de despliegue y el modelo de implementación son los modelos más importantes para describir la arquitectura de un sistema teniendo en cuenta que son los que proporcionan el desarrollo de las vistas de arquitectura.

El ciclo de vida de RUP tiene tres características fundamentales:

Guiado por casos de uso.

Los casos de uso reflejan las necesidades de los futuros usuarios, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. Los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura.

La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura (casos de uso arquitectónicamente significativo). El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

Iterativo e incremental.

RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son miniproyectos.

### 1.3.2 Tecnologías

#### Tecnología ASP

ASP (Active Server Page, por sus siglas en inglés) es una tecnología del lado del servidor de Microsoft para páginas Web generadas dinámicamente. La tecnología ASP intenta ser la solución adecuada para un modelo de programación rápida, ya que programar en ASP es como programar en Visual Basic, por supuesto con muchas limitaciones ya que es una plataforma que no se ha desarrollado como lo esperaba Microsoft.

#### Tecnología ASP.NET

ASP.NET es un conjunto de tecnologías de desarrollo de aplicaciones Web comercializado por Microsoft. Es usado por programadores para construir sitios Web domésticos, aplicaciones Web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la tecnología Active Server Pages (ASP).

#### .NET Framework

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones.

“El "framework" o marco de trabajo, constituye la base de la plataforma .NET y denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.” [6]

Bajo el nombre de .NET Framework se encuentran reunidas una serie de normas, por ejemplo: la norma que define las reglas que debe seguir un lenguaje de programación para ser considerado compatible con

el marco de trabajo .NET garantiza que todos los lenguajes desarrollados para la plataforma ofrezcan al programador un conjunto mínimo de funcionalidad, y compatibilidad con todos los demás lenguajes de la plataforma; la norma que define el lenguaje C# pretende reunir en este, las ventajas de lenguajes como C/C++ y Visual Basic.

**Tecnología MONO**

Mono es un proyecto de implementación de .NET de Microsoft utilizando código libre (open source). La iniciativa ofrece a los desarrolladores de Linux y UNÍX la posibilidad de construir y desplegar aplicaciones .NET multiplataforma. La actual patrocinadora del proyecto es la compañía de origen estadounidense Novell.

La plataforma .NET busca como objetivos ofrecer una independencia de lenguaje a los programadores. Así mismo es también más madura, documentada, más amplia en su ámbito de actuación y tiene un diseño consistente. Cualquier API que se escriba utilizando un lenguaje que genere código para el CLR (Common Language Runtime) puede usarse desde cualquier otro lenguaje que genere código para esta plataforma.

Mono pretende implementar las tecnologías desarrolladas por Microsoft, como C# y el CLI (Lenguaje Común de Infraestructura, por sus siglas en inglés) que han sido estandarizadas por la European Computer Manufacturers Association (ECMA).

### 1.3.3 Estilos arquitectónicos

En la ingeniería de software al igual que en la construcción; los estilos arquitectónicos describen categorías, conjunto de componentes, cooperación entre ellos, conectores y modelos. Podemos definir los estilos arquitectónicos como: conceptos descriptivos que definen una forma de articulación u organización arquitectónica.

El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales.

Los estilos arquitectónicos son la clave en la reducción del costo durante el desarrollo del software. Reutilizar soluciones efectivas es una de las prácticas fundamentales en el éxito del proceso de ingeniería, y lo mismo pasa en el contexto de la arquitectura de software. La reutilización a nivel de arquitectura se consigue con la adopción de estilos arquitectónicos.

Los estilos arquitectónicos fueron identificados por Mary Shaw y Paul Clements como "un conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer en sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo." [7]

A continuación se mencionan algunos de los estilos arquitectónicos más utilizados en la actualidad.

**Tuberías y filtros**

Tuberías y filtros es un estilo arquitectónico que siempre se enmarca dentro de las llamadas arquitecturas de flujo de datos.

"Una tubería (pipeline) es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. […] Debido a su simplicidad y su facilidad para captar una funcionalidad, es una arquitectura mascota cada vez que se trata de demostrar ideas sobre la formalización del espacio de diseño arquitectónico." [8]

Cuando un sistema posee arquitectura a modo de tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes.

El estilo arquitectónico tubería-filtros se puede utilizar cuando:

- Se puede especificar la secuencia de un número conocido de pasos.
- No se requiere esperar la respuesta asincrónica de cada paso.

- Se busca que todos los componentes situados corriente abajo sean capaces de inspeccionar y actuar sobre los datos que vienen de corriente arriba (pero no viceversa).

Igualmente se pueden señalar como ventajas del estilo tubería-filtros:

- Es simple de entender e implementar.
- Fuerza un procesamiento secuencial.

**Orientado a Servicios**

La Arquitectura Orientada a Servicios (en inglés Service-oriented architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

"La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP (Protocolo Simple de Acceso a Datos, por sus siglas en inglés) y WSDL (Lenguaje de Descripción de los Servicios Web, por sus siglas en inglés)) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios." [9]

Existen cuatro elementos básicos para la construcción de una Arquitectura Orientada a Servicios:

- Operación: Es la unidad de trabajo o procesamiento en una arquitectura SOA.
- Servicio: Es un contenedor de lógica. Estará compuesto por un conjunto de operaciones, las cuales las ofrecerá a sus usuarios.
- Mensaje: Para poder ejecutar una determinada operación, es necesario un conjunto de datos de entrada. A su vez, una vez ejecutada la operación, esta devolverá un resultado. Los mensajes son los encargados de encapsular esos datos de entrada y de salida.
- Proceso de negocio: Son un conjunto de operaciones ejecutadas en una determinada secuencia (intercambiando mensajes entre ellas) con el objetivo de realizar una determinada tarea.

Por lo tanto, una aplicación SOA estará formada por un conjunto de procesos de negocio, que a su vez estarán compuestos por aquellos servicios que proporcionan las operaciones que se necesitan ejecutar para que el proceso de negocio llegue a buen término. Estas operaciones se ejecutan mediante el envío de los datos necesarios a través de mensajes. El elemento básico de una arquitectura SOA es el servicio.

La Arquitectura Orientada a Servicios propone el uso de los estándares como XML, SOAP, WSDL, etc. Aunque no siempre es necesaria la implementación de todos ellos, es muy recomendable que se utilicen todos.

Estándar XML

El Lenguaje de Etiquetado Extensible (XML, por sus siglas en inglés) es un estándar de comunicación, es un metalenguaje con una importante función en el proceso de intercambio, estructuración y envío de datos en la Web.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y esas partes se componen a su vez de otras partes.

Protocolo SOAP

El protocolo simple de acceso a datos (SOAP), se encuentra en el núcleo de los servicios Web. Este protocolo proporciona un estándar para empaquetar mensajes, y es el primero de su tipo que ha sido aceptado prácticamente por todas las grandes compañías de software del mundo, incluso por aquellas que raramente cooperan entre si, tales como: Microsoft, IBM, SUN, Microsystems, SAP y Ariba.

SOAP fue creado por Microsoft, IBM y otras empresas, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Funciona sobre cualquier protocolo de Internet, generalmente HTTP, que es el único homologado por el W3C.

WSDL

WSDL son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web. WSDL describe la interfaz pública a los servicios Web.

Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

Un programa cliente que se conecta a un servicio Web puede leer el WSDL para determinar las funciones que están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

### 1.3.4 Patrones de Arquitectura

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución.

A continuación se argumentan algunos de los patrones de arquitectura más utilizados en la actualidad.

**Arquitectura en Capas**

"En [GS94] Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior." [10]

En la actualidad muchos sistemas de información están basados en arquitecturas de dos capas, denominadas generalmente nivel de aplicación y nivel de la base de datos. Este estilo tiene algunos inconvenientes como es la recarga del nivel de aplicaciones. Es por ello que existe una fuerte y avanzada tendencia a adoptar arquitectura en tres capa. (Ver Figura.1)

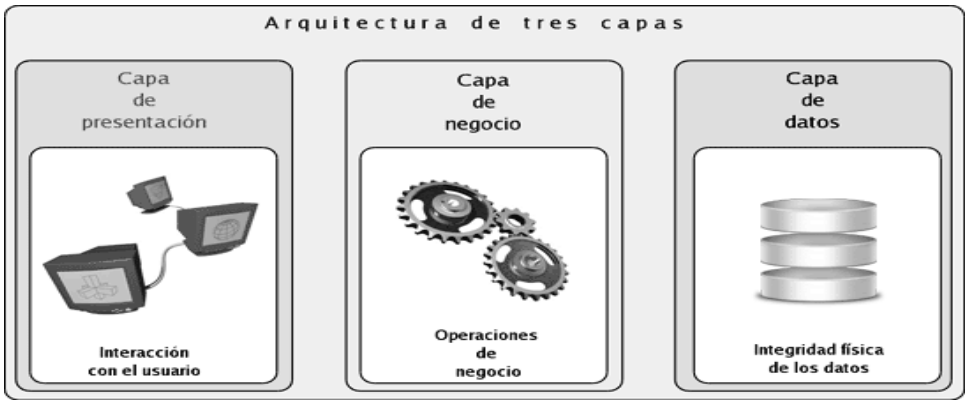



Figura.1 Arquitectura en tres Capas.

Las tres capas que propone esta arquitectura son:

- Lógica de Presentación. Esta es la parte del código que interactúa con un dispositivo como una PC o Terminal de autoservicio. Esta capa se encarga de tareas como la disposición de los elementos gráficos en la pantalla, escribir los datos en pantalla, manejo de ventanas, manejo de los eventos del teclado y Mouse, etc.

- Lógica de Negocio. En esta capa se codifican las reglas del negocio. Por ejemplo, si el sistema es de un banco en esta capa se programan conceptos como plazo fijo, cuenta corriente, cheque, etc; si es una compañía de seguros existirán componentes para asegurados, pólizas, siniestros, etc.

- Lógica del Procesamiento de los Datos. En esta capa se oculta la forma en que se consultan o almacenan los datos persistentes en la base datos.

Este patrón puede ser difícil a la hora de definir qué componentes ubicar en cada una de las capas, sin embargo mejora el soporte del sistema y facilita la localización de errores.

**Arquitectura Cliente – Servidor**

La arquitectura cliente servidor consiste básicamente en que un programa realiza peticiones a otro, y este último le da respuesta a dichas peticiones. Al proceso que inicia el diálogo o solicita los recursos se le denomina cliente, y al proceso que responde a estas solicitudes se le denomina servidor.

### 1.3.5 Lenguajes de Programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.

**Lenguaje C**

C es un lenguaje de programación apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente categorizado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

Ventajas del lenguaje C:

- Lenguaje muy eficiente puesto que es posible utilizar sus características de bajo nivel para realizar implementaciones óptimas.

- A pesar de su bajo nivel es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas conocidos.

- Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.

**Lenguaje C++**

C++ es un lenguaje de programación que está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje.

Anteriormente se había usado el nombre "C con clases", "C++" significa "incremento de C" y se refiere a que C++ es una extensión de C.

Dentro de las principales características de este lenguaje de programación está el soporte para programación orientada a objetos.

**Lenguaje C#**

C# es un lenguaje de programación de propósito general, orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.

C# significa, "do sostenido" (C corresponde a do en la terminología musical anglo-sajona). El símbolo # viene de sobreponer "++" sobre "++" y eliminar las separaciones, indicando así su descendencia de C++.

Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones; mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.

Algunas de las ventajas del Lenguaje C#:

- La facilidad del lenguaje permite crear aplicaciones para Windows en muy poco tiempo. Por tanto, permite un desarrollo eficaz y menor inversión en tiempo que con cualquier otro lenguaje.

- La sintaxis es flexible: Se puede obligar al compilador a ignorar errores o escribir varias instrucciones en una misma línea. El IDE detecta las variables existentes, y en el caso de que al utilizarlas las escribamos de forma diferente -con mayúsculos, por ejemplo- cambia dicha variable en todo el código.

- Permite generar librerías dinámicas (DLL).

- Las versiones más recientes (.NET) incluyen una tecnología denominada control de versiones en paralelo que permite que varias versiones del mismo componente estén instaladas en el mismo equipo con seguridad, de manera que las aplicaciones pueden utilizar una versión determinada de dicho componente.

**Lenguaje PHP**

“PHP es un lenguaje de programación usado frecuentemente para la creación de contenido para sitios Web con los cuales se puede programar las páginas html y los códigos de fuente. PHP es un acrónimo recursivo que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web.” [10]

Es un lenguaje multiplataforma. Es libre por tanto se presenta como alternativa de fácil acceso para todos. Tiene capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, especialmente MySQL.

**Lenguaje PERL**

PERL es un acrónimo de Practical Extracting and Reporting Language (Lenguaje Práctico para la Extracción e Informe). Estructuralmente PERL esta basado en un estilo en bloques como los del C. Es lo que se conoce como un lenguaje ``script''. Fue adoptado por su destreza en el procesamiento de texto y por no tener ninguna de las limitaciones de los otros lenguajes de scripts.

PERL es relativamente rápido para un lenguaje tipo ``script''. Esta disponible en múltiples plataformas y sistemas operativos. Es un buen lenguaje ``pegamento''. Se pueden juntar varios programas de una forma sencilla para alcanzar una meta determinada, los usuarios de Windows agradecerán esta propiedad ya que normalmente adolecen de un buen lenguaje tipo ``script''. Es software libre.

### 1.3.6 Gestores de Bases de Datos

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

**Postgre SQL**

Postgre SQL es un Sistema Gestor de Base de Datos (SGBD), un servidor de base de datos relacional libre. La alta concurrencia se encuentra entre sus principales características: mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

**MySQL**

MySQL es un sistema de gestión de bases de datos relacionales. Una base de datos relacional almacena datos en tablas separadas en lugar de poner todos los datos en un gran almacén, lo que añade velocidad y flexibilidad. La parte SQL de "MySQL" se refiere a "Structured Query Language". SQL es el lenguaje estandarizado más común para acceder a bases de datos.

MySQL es Open Source, lo que quiere decir que es posible para cualquiera usar y modificar el software. Cualquiera puede bajar el software MySQL desde Internet y usarlo sin pagar nada. Si lo desea, puede estudiar el código fuente y cambiarlo para adaptarlo a sus necesidades.

El servidor de base de datos MySQL es muy rápido, fiable y fácil de usar. MySQL Server trabaja en entornos cliente/servidor o incrustados. Además, funciona en diferentes plataformas y fue escrito en C y en C++.

### 1.3.7 Estándar HL7

“HL7 es una organización internacional, iniciada en los Estados Unidos en 1987, que pretende promover el desarrollo y evolución del estándar HL7 (Health Level Seven) para el formato de datos e intercambio de información entre diferentes Sistemas de Información de Salud.”[11] Su misión es definir y publicar especificaciones y protocolos para comunicar sistemas de información sanitarios, dispersos, diferentes y heterogéneos.

HL7 es un estándar que permite que las aplicaciones clínicas se comuniquen entre sí, independientemente de: la arquitectura de los datos, la plataforma tecnológica o el lenguaje de desarrollo.

El mismo debe su nombre a que fue concebido como estándar para la capa 7 (Nivel de aplicaciones) del modelo OSI (Interconexión de Sistemas Abiertos, por sus siglas en ingles), donde la unidad de información es el mensaje. Por ello es relativamente independiente del tipo de conexión física y protocolo de comunicación usado; se ocupa exclusivamente del proceso de dar formato a los datos para convertirlos en mensajes que cualquier aplicación que cumpla la norma puede entender.

**Mensajes HL7 2.X**

Un mensaje HL7 es la unidad atómica de transmisión de datos entre sistemas. Cada mensaje está formado por segmentos en una secuencia definida y cada segmento, a su vez, está formado por campos.

Segmento: Un segmento HL7 es una agrupación de campos. Los segmentos dentro de un mensaje pueden ser requeridos u opcionales, pueden ocurrir una sola vez o permitir repeticiones, se identifican por un código único de tres caracteres denominado “SEGMENT ID”. HL7 permite en cada implementación definir segmentos específicos para intercambiar información no prevista (Segmentos Z (o de usuario)).

Campo: Un campo es una cadena de caracteres definida por un tipo de datos HL7. Cada campo también puede tener partes o componentes separables. Por ejemplo, el nombre del paciente se registra como Apellido, Nombre, Inicial del Segundo Nombre.

Eventos disparadores: El evento disparador es el hecho que genera la transmisión del mensaje. Por ejemplo: la información sobre el ingreso/admisión del paciente, normalmente es ingresada por el sistema de admisión de pacientes e informada al resto de los sistemas que conforman la organización; un evento ADT-A01 es enviado cuando se realiza el ingreso/admisión del paciente, de esta forma, el evento ADT-A01 puede ser usado para notificar al sistema Laboratorio que un paciente ha sido admitido, por lo tanto a dicho paciente se le puede fehacientemente solicitar estudios.

Existen diferentes tipos de mensajes que son utilizados para el intercambio de información del paciente, por ejemplo:

- ADT (Admit): Es el tipo de mensajes relacionados con la admisión o registro de pacientes.

- ACK (Acknowledgement): Se utiliza para confirmar el acuse de recibo de un mensaje. Indica si hubo o no un error al procesar el mensaje.
- ORM (Order Message): Mensaje para órdenes en HL7 V2.x.
- ORR (Order Response): Mensaje para respuesta a órdenes en HL7 V2.x.

Los mensajes HL7 tienen una estructura sumamente sencilla en términos electrónicos (ver Anexos), y es fácilmente interpretable y transportable hacia cualquier sistema o aplicación. La forma en la cual se transmitan los mensajes no esta específicamente indicada en el estándar y deberá ser acordada entre los sistemas emisor y receptor del mensaje.

**Codificación de los mensajes HL7**

Los mensajes HL7 pueden ser codificados mediante XML o String. La codificación String requiere la implementación de un parser capaz de analizar sintácticamente el mensaje, mientras XML cuenta con la implementación de esquemas que validan el formato de los mensajes.

Por ejemplo, XML Schema (XSD) es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML.

Dicho lenguaje de esquema está escrito en XML, basado en la gramática y pensado para proporcionar una mayor potencia expresiva que la DTD (Definición de tipo de Documento).

El principal aporte de XML Schema es el gran número de los tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y strings.

### 1.3.8 Entrada / Salida de los mensajes

Para la entrada y salida de los datos existen diferentes configuraciones posibles. Podrían ser a través de conexiones punto a punto o a través del uso de servicios web. Para ello se puede hacer uso del protocolo TCP/IP o de servicios web.

**Protocolo TCP/IP**

La familia de protocolos de Internet es un conjunto de protocolos de red que implementa la pila de protocolos en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras. En ocasiones se la denomina conjunto de protocolos TCP/IP, en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), que fueron los dos primeros en definirse, y que son los más utilizados de la familia.

TCP, Transmission Control Protocol por sus siglas en inglés, es uno de los protocolos fundamentales en Internet. Es usado para crear conexiones entre los ordenadores de una red para el intercambio de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

TCP es un protocolo de comunicación orientado a conexión y fiable. Se encuentra en el cuarto nivel del modelo OSI (Open System Interconnection), nivel de transporte, encargado de la transferencia libre de errores de los datos entre el emisor y el receptor, capaz de detectar y eliminar paquetes duplicados, etc.

IP, Internet Protocol por sus siglas en ingles, es un protocolo NO orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

Los datos son enviados en bloques conocidos como paquetes o datagramas. Si la información a transmitir ("datagramas") supera el tamaño máximo "negociado" (MTU) en el tramo de red por el que va a circular podrá ser dividida en paquetes más pequeños, y reensamblada luego cuando sea necesario. Estos fragmentos podrán ir cada uno por un camino diferente dependiendo de como estén de congestionadas las rutas en cada momento.

IP no es fiable, no provee ningún mecanismo para determinar si un paquete alcanza o no su destino. Por ejemplo, al no garantizar nada sobre la recepción del paquete, éste podría llegar dañado, en otro orden con respecto a otros paquetes, duplicado o simplemente no llegar. Si se necesita fiabilidad, ésta es proporcionada por los protocolos de la capa de transporte, como TCP.

**Servicio**

Un servicio es una aplicación que cumple las siguientes características:

- Suele iniciarse automáticamente al poner en marcha el sistema, sin necesidad siquiera de que el usuario inicie una sesión interactiva.

- Por regla general los servicios no cuentan con una interfaz de usuario. Se ejecutan de manera silenciosa sin interferir en las tareas habituales del usuario.

- Un servicio se mantiene en ejecución, normalmente, todo el tiempo que el sistema está en funcionamiento. [12]

**Servicios WEB**

En su origen, los Web Services (Servicios Web) fueron creados como un método para compartir recursos en la red. En un entorno donde el aumento constante del número de usuarios demandaba cada vez más un mayor número de recursos en la red, surgió la necesidad de facilitar la distribución entre empresas de dichos recursos para satisfacer las necesidades de sus clientes. El resultado fue el desarrollo de una tecnología de muy fácil implantación y que era capaz de solucionar los aspectos de disponibilidad e inmediatez que se requerían.

Los Web Services son pequeños programas formados por varios componentes que permiten ser publicados en directorios e invocados para su ejecución por otros programas vía HTTP (HyperText Transfer Protocol), generando una respuesta en XML.

Los Servicios Web pueden realizar operaciones por si mismos, e incorporar otros Servicios Web (composición de servicios). Esto permite completar y automatizar transacciones complejas y de alto nivel.

Quizás la ventaja principal de los Web Services es que se trata de un estándar aceptado y que, a diferencia de otras tecnologías de integración, posibilitan la compartición de funcionalidades entre sistemas heterogéneos de forma transparente, mediante el intercambio de datos vía XML. Para este intercambio el único requisito es establecer conexiones TCP/IP posibilitando la comunicación HTTP entre los sistemas.

A pesar de que existen otras tecnologías para computación distribuida, que permiten la comunicación de sistemas en la red, tales como CORBA (Common Object Request Broker Architecture), en comparación con los Servicios Web ellas resultan limitadas, más caras y menos flexibles. Algunos de los problemas de tales tecnologías son:

- Su alta complejidad, y alta inversión necesaria en arquitectura.

- Su funcionalidad está limitada a ciertos tipos de plataforma, lo que dificulta la interoperabilidad entre sistemas heterogéneos.

- Utilizan protocolos cerrados, distintos formatos para mensajes y representación de datos.

Es por ello que el uso de la Web, con protocolos abiertos altamente extendidos como HTTP y TCP/IP, formatos de mensajes y estructuras de datos basados en el estándar XML, hacen a los Servicios Web una excelente alternativa para alcanzar comunicación e interoperabilidad de sistemas heterogéneos.

Los Web Services al estar basados en protocolos ampliamente aceptados, son multiplataforma, facilitando la implantación de los mismos independientemente de la plataforma donde se implante la aplicación.

Esto no implica que los Servicios Web sean el reemplazo obligado de otras tecnologías de computación distribuida, sino que es un valioso componente de más alto nivel, que permite interoperabilidad entre ellas, y facilita la automatización.

Los Servicios Web están definidos, esencialmente por tres tecnologías, construidas sobre los protocolos y estándares de la Web: Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) y Web Services Description Language (WSDL).

Las tecnologías XML, SOAP y WSDL, resultan ser la base mínima necesaria para la descripción e interacción entre agentes en la Web.

Los ahorros de coste no es la única ventaja que aportan los Web Services, permiten además, la distribución de los procesos en la red. La extensión del uso de los Web Services, facilita la reutilización de

recursos aligerando el peso de las aplicaciones, ya que la lógica de negocio puede repartirse entre distintos sistemas.

### 1.3.9 Herramientas de desarrollo

**Rational Rose**

Rational Rose es la herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

El navegador UML de Rational Rose nos permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de casos de uso, vista lógica, vista de componentes y vista de despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

**Visual Studio.NET**

Visual Studio .NET es un IDE (Entorno de Desarrollo Integrado, por sus siglas en ingles) desarrollado por Microsoft**.** La característica más notable del IDE es su soporte de los nuevos lenguajes .NET. como C#, Visual Basic, C++, ASP, etc.

Visual Studio .NET es la herramienta de desarrollo multilenguaje más completa para construir e integrar rápidamente aplicaciones y servicios Web XML. Aumenta de un modo extraordinario la productividad de los desarrolladores y crea nuevas oportunidades de negocio.

En su diseño se han integrado a fondo los estándares y protocolos de Internet, como XML y SOAP, por lo que Visual Studio .NET simplifica considerablemente el ciclo de vida del desarrollo de aplicaciones.

Características.

- Nueva ayuda dinámica: Proporciona acceso instantáneo a ayuda relevante para la tarea de desarrollo actual.

- Nueva interoperabilidad sin problemas entre lenguajes: Los desarrolladores pueden heredar fácilmente código y formularios de otros lenguajes .NET, interactuar sin problemas con otros lenguajes y depurar aplicaciones multilenguaje.

**SharpDevelop**

SharpDevelop es un Entorno Integrado de Desarrollo (IDE) libre para los lenguages de programación C#, Visual Basic .NET y Boo. Es usado típicamente por aquellos programadores de los citados lenguajes, que no desean o no pueden usar el entorno de desarrollo de Microsoft, el Visual Studio.

SharpDevelop está completamente escrito en C# y dentro de sus características más importantes está que cuenta con completado de código, conversor bidireccional entre C# y Visual Basic .NET, y unidireccional hacia Boo, previsualización de documentación XML, además permite importar los proyectos creados con Microsoft Visual Studio .NET.

## 1.4 Topología de red

La topología de la red es la disposición física en la que se conectan los nodos de una red de ordenadores o servidores mediante la combinación de estándares y protocolos. Entre las topologías físicas más frecuentes en el mundo de las redes se encuentran bus, malla, anillo, estrella, etc.; éstas se pueden combinar obteniendo una variedad de topologías híbridas más complejas.

El diseño de una red puede incluir aspectos tales como fiabilidad, camino más económico, factores de coste, etc. El diseñador de la red debe analizar los objetivos que persigue antes de plantear una determinada topología.

La topología totalmente conexa permite que cada nodo de una red esté conectado con los restantes nodos (ver Figura. 2). Esta topología dificulta el proceso de configuración de los nodos cuando uno de ellos deba ser eliminado o agregado a la red, debido a que cada aplicación debe saber con quien debe o con quien no debe comunicarse.

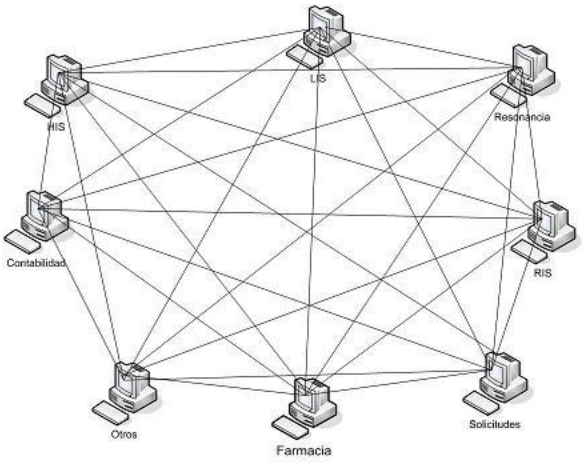



Figura. 2 Topología de red totalmente conexa.

Utilizando una red con topología de estrella se pueden conectar todos los nodos a un nodo central (ver Figura. 3). El fallo del nodo central provocaría la caída de la red; sin embargo este tipo de topología es muy fiable, no existen problemas con colisiones de datos ya que cada estación tiene su propio cable al hub central. Además es fácil de implementar y de ampliar, incluso en grandes redes.

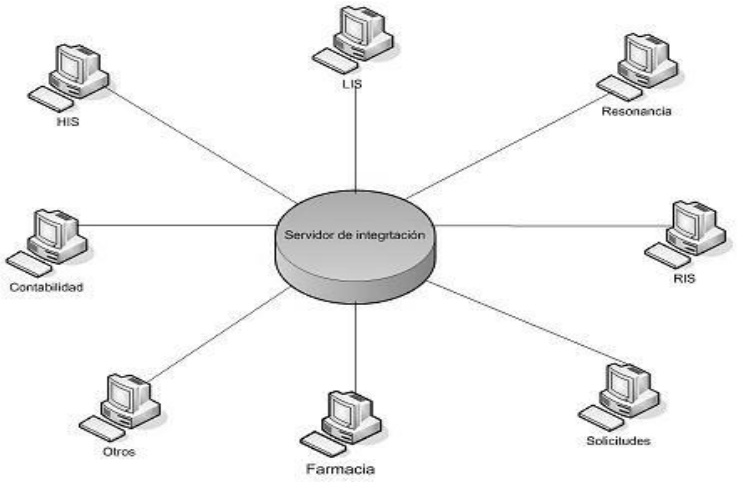



Figura. 3 Topología de estrella.

### 1.4.1 ¿Por qué utilizar un servidor de integración para lograr la comunicación entre sistemas?

Una topología totalmente conexa, por ejemplo, no sería la más adecuada para conectar sistemas desarrollados para el área sanitaria, teniendo en cuenta la gran cantidad de aplicaciones que están surgiendo.

Este tipo de topología tiene como desventaja que para agregar o eliminar un nuevo sistema a la red habría que configurar cada uno de los restantes nodos, de forma que cada uno de ellos sepa cuando se ha incorporado o retirado un sistema.

La propuesta del presente trabajo es logar la integración de los diferentes sistemas existentes en el sector sanitario utilizando una red con topología de estrella, donde todos los nodos estén conectados a un servidor central (servidor de integración), permitiendo que cuando un sistema deba ser eliminado o agregado a la red solo sea informado al servidor central, para que este actualice su configuración.

El servidor de integración propuesto debe funcionar como un servicio. Tiene que ser capaz de mantenerse en ejecución todo el tiempo en que el sistema esté funcionando, sin interferir en el resto de las tareas que desarrolle el mismo y debe ser capaz de iniciarse automáticamente una vez que se ponga en marcha el equipo (máquina o computadora).

**Conclusiones**

Después de haber analizado las tendencias, tecnologías y metodologías se propone RUP como metodología de desarrollo ya que es una de las más utilizadas en el mundo para el análisis, implementación y documentación de sistemas orientados a objetos. Además es un proceso de ingeniería del software que proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Es posible su utilización tanto en pequeños como grandes proyectos.

Se propone utilizar el estilo arquitectónico “tuberías y filtros” debido a sus características ya mencionadas y por la facilidad que nos brinda de dividir la entrega de los mensajes en pasos o tareas para una mejor organización del sistema.

Además de este estilo arquitectónico también se propone implementar una arquitectura de dos capas, para separar la lógica del negocio del acceso de los datos permitiendo a los desarrolladores la fragmentación de un problema complejo en una secuencia de pasos incrementales.

C# es un lenguaje moderno y estándar, orientado a objeto y con una amplia biblioteca de clases que facilitan el desempeño de los desarrolladores. Su naturaleza moderna aporta sintaxis y semántica que enriquecen las aplicaciones incluyendo como parte del lenguaje algunos patrones de diseño. El hecho de haber sido estandarizado permite que terceras partes hagan su implementación libre de compiladores y en general de la plataforma **.**NET, lo cual concuerda con la estrategia de hacer uso de tecnologías libres que lleva a cabo el país, por estas razones se escoge como lenguaje de programación para la implementación de la solución propuesta.

Para almacenar los datos de interés en el servidor de integración es recomendable utilizar MySQL debido a que el tamaño de la base de datos a desarrollar es pequeña y el rendimiento de Postgres disminuye en estos casos.

La necesidad de lograr la integración entre los sistemas en el ámbito de la salud, ha traído consigo que se promueva el uso de estándares que permitan dicha integración. En la actualidad existe la tendencia a utilizar HL7 como estándar para el formato de los mensajes, permitiendo que las aplicaciones se comuniquen entre sí, independientemente de la arquitectura de los datos, la plataforma tecnológica o el lenguaje de desarrollo.

Por lo tanto, se propone el uso de HL7 para estandarizar los mensajes que deben ser enviados entre las diferentes aplicaciones que se comuniquen en un hospital. Los mensajes serán codificados mediante XML, evitando así la implementación de un parser ya que XML cuenta con esquemas que validan el formato de los mensajes.

Uno de los objetivos más importante en la comunicación entre aplicaciones sanitarias es lograr eficiencia en el intercambio de información. Se propone el uso del protocolo TCP/IP para el intercambio de mensajes entre los sistemas porque además de garantizar la comunicación de forma rápida, segura y sin pérdida de información, la versión 2.x de HL7 y en particular la versión 2.3.1 que es la propuesta a utilizar no tiene estandarizados los servicios Web.

Como herramientas de desarrollo se propone Visual Studio .Net y SharpDevelop como IDE libre para posteriormente compilar los desarrollos. Mientras se esté programando la solución, se utilizará el Visual Studio .Net con el .Net framework 2.0. Una vez terminada, se puede compilar con MONO, cargando la solución en el SharpDevelop. Desde el inicio se puede programar directamente en el SharpDevelop pero el mismo no brinda todas las funcionalidades del Visual Studio, lo cual tiene un costo en tiempo.

# CAPITULO 2 .Descripción de la Arquitectura

La arquitectura es el esqueleto o base de una aplicación, en esta se analiza la aplicación desde varios puntos de vista. En la arquitectura aparecen los artefactos más importantes para establecer un esquema de cómo deben ser los próximos artefactos a construir, es un semi-molde al que se deben ajustar sin muchos cambios, los restantes artefactos a construir en la aplicación o solución. De obtenerse un artefacto demasiado diferente a los demás, este formaría parte de la arquitectura.

En este capítulo se especifican los requerimientos funcionales y no funcionales, casos de uso, clases y diagramas que son arquitectónicamente significativos para el sistema.

## 2.1 Descripción del sistema propuesto

La propuesta para solucionar los problemas de integración entre los sistemas que existen dentro de un hospital, es un servidor que permita la comunicación entre ellos mediante el envío de mensajes HL7. Dichos mensaje deben ser codificados mediante XML permitiendo el uso de los esquemas para validarlos lo que evita la implementación de un parser.

El servidor será implementado como un servicio; siempre tendrá que estar disponible para escuchar un mensaje, validarlo y enviarlo a su destinatario, además debe ser capaz de ejecutarse cuando se inicialice el sistema operativo.

Se propone un estilo arquitectónico de tipo tuberías y filtros para el flujo interno de los datos en el servidor y la implementación de una arquitectura SOA para lograr la integración del mismo con el resto de los sistemas de información.

El sistema será implementado en el lenguaje de programación C# de la plataforma .NET. Utilizando HL7 como estándar de formato de los mensajes, TCP/IP como protocolo de comunicación y XML como estándar de comunicación.

Los mensajes enviados y recibidos en el servidor serán almacenados en una base de datos, usando MySQL como gestor.

**Flujo de los datos en el servidor de integración**

El flujo de datos en el servidor de integración propuesto se realizará como muestra la siguiente figura:

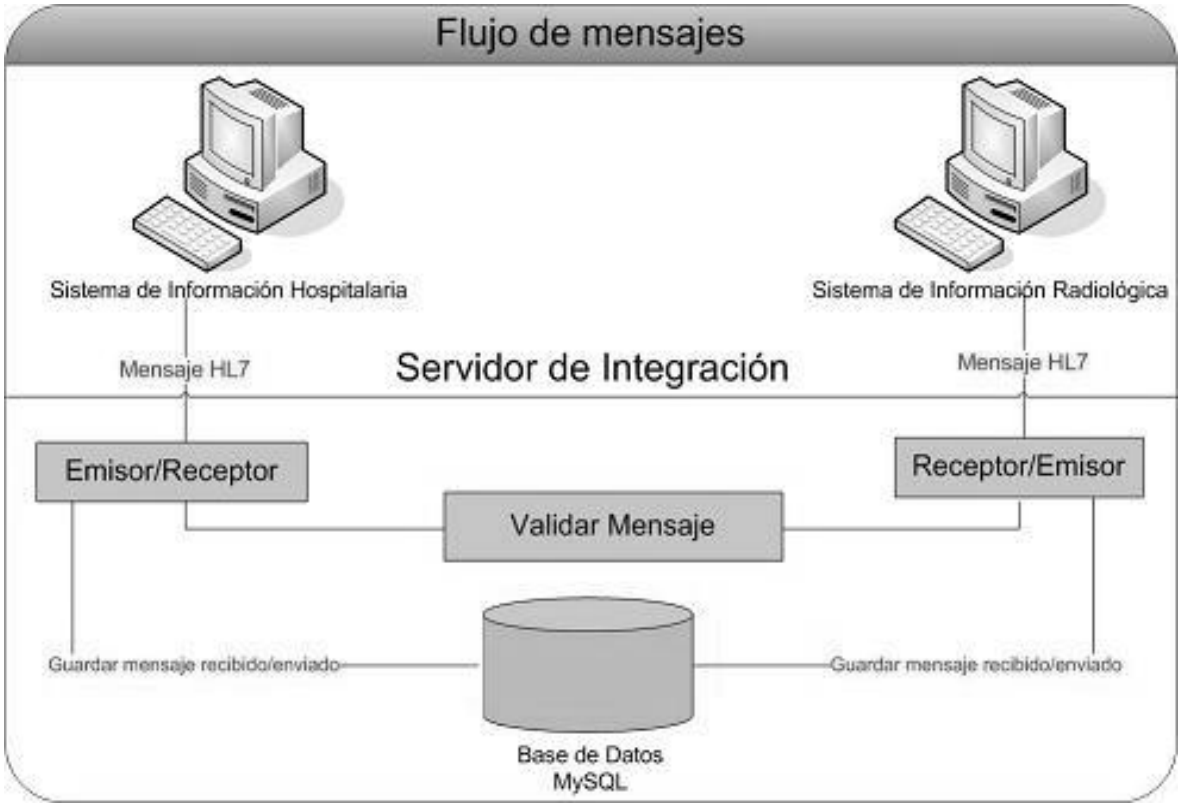



Figura. 4 Flujo de los mensajes en el servidor de integración.

Cuando en uno de los sistemas conectados al servidor de integración se produce un evento determinado (Admisión de Paciente, Alta de Paciente, etc.), se emite un mensaje en formato HL7 a los sistemas que estén involucrados con dicho evento.

El servidor de integración recibe cada uno de los mensajes que se envían de un sistema a otro a través de una interfaz de comunicación. Cada mensaje es validado contra un esquema XML para verificar su

integridad, si dicho mensaje no es correcto entonces se envía un mensaje de error al emisor del mensaje, en caso contrario es enviado a su destinatario.

Los mensajes son almacenados en una base de datos para que quede una traza de los mensajes que recibe y envía el servidor de integración.

### 2.1.1 Requerimientos

La arquitectura de un sistema está guiada, en gran medida, por los requerimientos (funcionales y no-funcionales) que debe cubrir el sistema y normalmente se toma el subconjunto arquitectónicamente más importante de dichos requerimientos para definirla.

Según Booch, Rumbaugh y Jacobson los requerimientos de un sistema se definen como:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

- Define que es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

- Es una característica que un sistema debe tener para cubrir alguna de las necesidades que lo motivan.

Los **requerimientos funcionales** son capacidades o cualidades que el sistema debe cumplir.

Los **requerimientos no funcionales** son propiedades o cualidades que el producto debe tener, los mismos se agrupan en diferentes categorías: Requerimientos de software, Requerimientos de Hardware, Restricciones en el diseño y la implementación, Requerimientos de Seguridad, Requerimientos de Usabilidad y Requerimientos de Soporte.

A continuación se hace mención de los requerimientos funcionales y no funcionales arquitectónicamente más significativos para el desarrollo del sistema en cuestión.

**Requerimientos funcionales del servidor**

RF1: Recibir mensajes.

RF2: Enviar mensajes.

RF3: Validar mensajes.

RF4: Agregar sistema.

RF5: Eliminar sistema.

RF6: Modificar sistema.

**Requerimientos no funcionales del servidor**

Requerimientos de software:

- .NET Framework 2.0 y MONO en una versión compatible con .NET Framework 2.0. Cualquier sistema operativo que soporte .NET Framework 2.0 y MONO.

Requerimientos de Hardware:

- Pentium IV, 2 GHz, 256 de RAM.

Restricciones en el diseño y la implementación:

- Estándar HL7, XML.
- Lenguaje C# de la plataforma .NET.
- Utilizar como herramienta de desarrollo Visual Studio 2005, Sharp Develop (en caso de MONO).
- Bibliotecas de clases de .NET.

Requerimientos de Seguridad:

- La información manejada en el servidor debe estar disponible en cualquier momento.

- A la aplicación solo podrá acceder el personal autorizado.

- El uso de TCP/IP garantiza la entrega de los datos sin errores. Por tanto, el sistema cumple con los requisitos de disponibilidad, confidencialidad e integridad.

Requerimientos de Usabilidad:

- La aplicación será usada por los especialistas. (Pudieran ser los administradores de redes).

Requerimientos de Soporte:

- Compatibilidad con .NET 2.0.

- Configuración: Puerto disponible, haciendo uso de los servicios TCP/IP.

## 2.2 Descripción de la Arquitectura

### 2.2.1 Vistas arquitectónicas

En RUP, la arquitectura se compone de 4+1 vistas fundamentalmente: Vista de Casos de Uso, Vista Lógica, Vista de Procesos, Vista de Implementación y Vista de Despliegue. Estas vistas son la parte esencial de lo que se obtuvo en los flujos de Requerimientos, Análisis y Diseño e Implementación, que son las etapas que más tributan a la arquitectura.

La arquitectura se representa a través de 4+1 vistas (Ver figura. 5), dicho de modo que se entienda que 4 de estas vistas están regidas por una rectora: la Vista de Casos de Uso. Esto responde, además, a una de las características de RUP: Guiado por Casos de Usos.

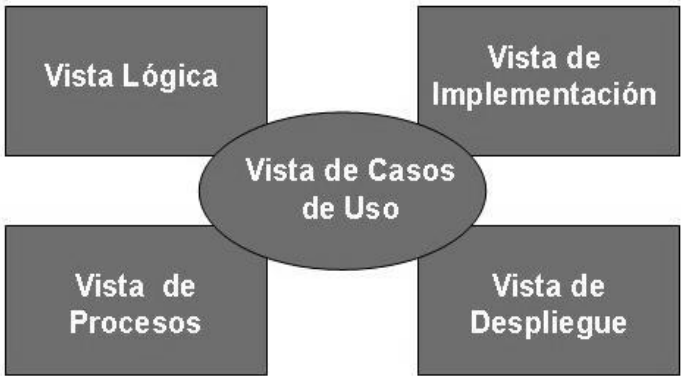



Figura. 5 Vistas de la arquitectura propuesta por RUP.

**2.2.1.1 Vista de Casos de Uso**

La vista de casos de uso representa un subconjunto del artefacto Modelo de Casos de Uso y lista los casos de uso o escenarios del modelo de casos de uso más significativos, con las funcionalidades centrales del sistema.

En esta vista se representa el diagrama de casos de uso con los casos de uso arquitectónicamente significativos para el sistema y una breve descripción de cada uno.

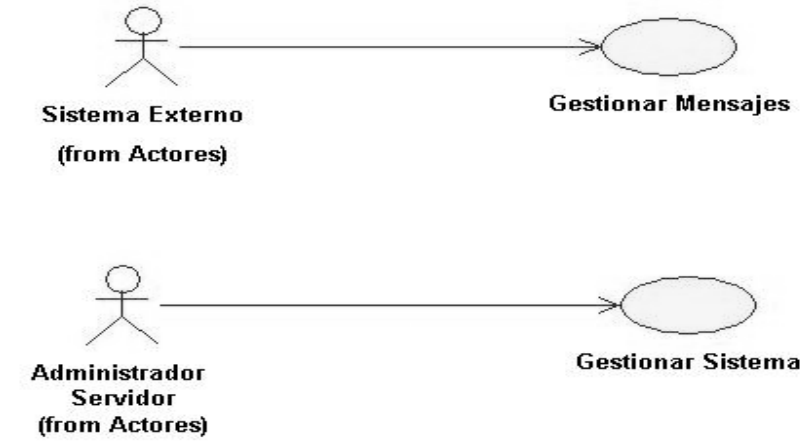



Figura. 6 Modelo de Casos de Uso Arquitectónicamente Significativos.

**Descripción de los casos de uso del sistema arquitectónicamente más significativos**

| **Caso de uso** | | |
|---|---|---|
| **Nombre** | Gestionar sistema. | |
| **Actores** | Administrador Servidor | |
| **Propósito** | Permite agregar un nuevo sistema o eliminar un sistema existente que se encontraba integrado al servidor. | |
| **Resumen:** El caso de uso se inicia cuando el administrador del servidor desea agregar un sistema que deba ser integrado o eliminar otro ya existente que no deba continuar integrado al servidor. | | |
| **Referencias** | RF4, RF5, RF6 | |
| **Acción del actor** | | **Respuesta del sistema** |
| 1. El Administrador agrega un nuevo sistema.<br><br>2. El Administrador elimina un sistema de la red. | | 1.1 El servidor de integración agrega al nuevo sistema.<br><br>2.1 El servidor de integración elimina al sistema. |

Tabla 1 - Descripción textual del caso de uso Gestionar Sistema.

| **Caso de uso** | | |
|---|---|---|
| **Nombre** | Gestionar Mensajes. | |
| **Actores** | Sistema Externo (Cualquier sistema de información existente en un hospital). | |
| **Propósito** | Gestionar los mensajes que se envían entre los sistemas que existen en un hospital para lograr la integración de los mismos. | |
| **Resumen:** El caso de uso se inicia cuando el sistema externo gestiona un mensaje. El mensaje puede ser recibido o enviado y tiene que ser validado. | | |
| **Referencias** | RF1,RF2, RF3 | |
| **Acción del actor** | | **Respuesta del sistema** |
| 1. El sistema externo envía un mensaje a otro sistema externo que este conectado al servidor de integración. | | 1.1 El sistema recibe el mensaje.<br>1.2 El sistema va a la sección “Validar Mensaje”.<br>1.3 El sistema envía el mensaje. |
| **Sección “Validar Mensaje”** | | |
| **Acción del actor** | | **Respuesta del sistema** |
| | | 1.1 El sistema valida el mensaje.<br>1.2 Si el mensaje es correcto entonces se envía un mensaje de aceptación (ACK) al sistema externo emisor y se reenvía el mensaje que se validó al sistema externo receptor.<br>1.3 Si el mensaje no es correcto entonces se envía un mensaje de error (ACK) al sistema externo emisor.<br>1.4 El sistema comprueba si el destinatario y el emisor del mensaje son sistemas que existen en el hospital mediante una verificación a la configuración. |

Tabla 2 - Descripción textual del caso de uso Gestionar Mensaje.

### 2.2.1.2 Vista Lógica

Esta vista representa un subconjunto del artefacto Modelo de Diseño, representando los elementos de diseño más importantes para la arquitectura del sistema. En esta vista se describe las clases más importantes, su organización en paquetes y subsistemas. También describe las realizaciones de casos de uso más importantes como por ejemplo las que describen aspectos dinámicos del sistema.

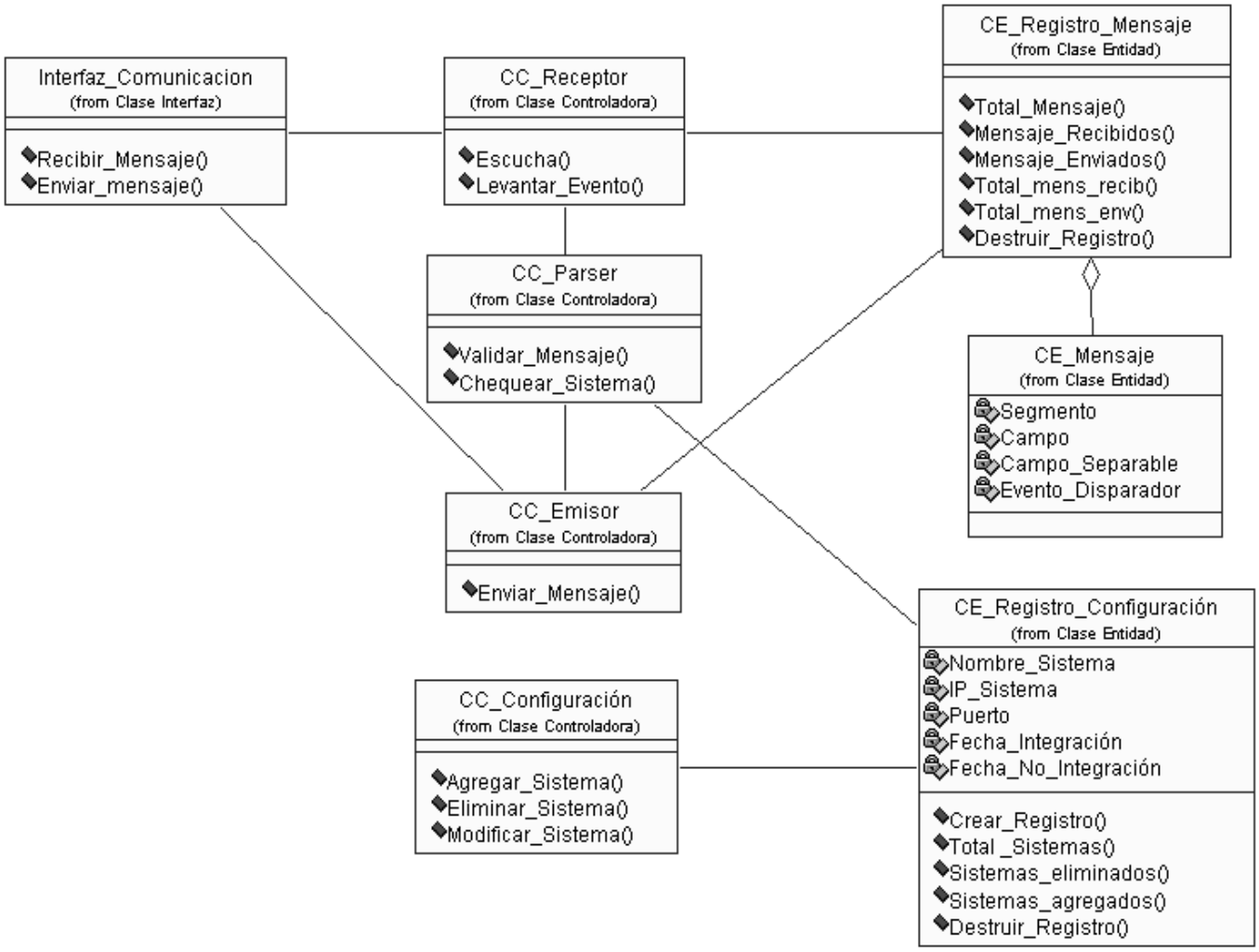



Figura. 7 Modelo de Diseño.

A continuación se muestra una breve descripción de las clases que están presentes en el modelo de diseño de los casos de uso arquitectónicamente más significativos.

**Clase Interfaz de Comunicación**: Es la interfaz que permite la comunicación con los otros sistemas. Mediante la misma el servidor de integración puede recibir y enviar los mensajes.

**CControladora_Receptor**: Permite recibir el mensaje desde la interfaz de comunicación.

**CControladora_Parser**: Permite validar el mensaje para evaluar su integridad. También tiene como función verificar si el remitente y el destinatario del mensaje son sistemas que el servidor de integración reconoce como sistemas que existen en el hospital.

**CControladora_Emisor**: Recibe el mensaje desde la CControladora_Parser y lo envía a la interfaz de comunicación que se encargará de enviarlo a su destinatario.

**CEntidad_Registro_Mensaje**: Registra los mensajes cuando son recibidos y enviados por el servidor de integración.

**CEntidad_Mensaje**: Contiene la información referente al mensaje, los segmentos, campos, etc.

**CControladora_Configuración**: Esta clase permite agregar un nuevo sistema que necesite ser integrado. Además posibilita eliminar un sistema que no necesite integrarse al servidor y modificar algún dato de los sistemas integrados.

**CEntidad_Registro_Configuración**: Almacena los datos de los sistemas que están integrados con el servidor de integración.

**Realización de los Casos de Uso arquitectónicamente más significativos**

Las realizaciones de casos de uso que se muestran a continuación describen el funcionamiento y los aspectos dinámicos del servidor de integración.

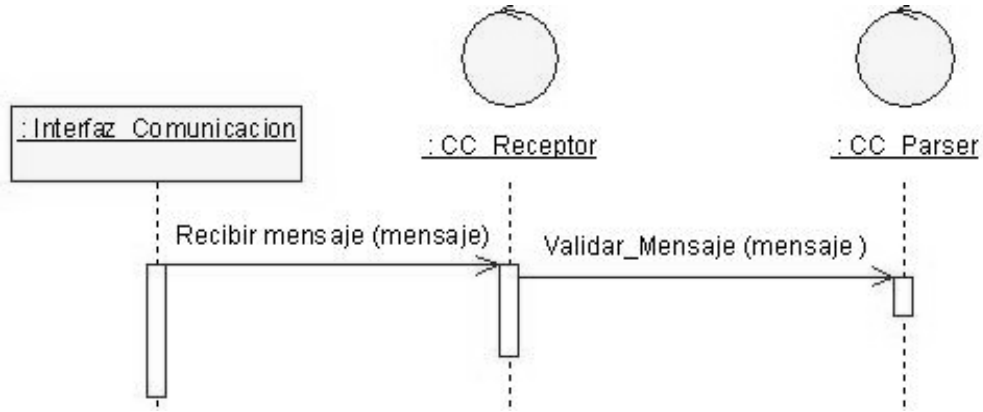



Figura.8 Escenario Validar Mensaje

En el escenario **validar mensaje** se muestra una descripción de flujos de sucesos que permiten demostrar la dinámica durante la validación del mensaje, solo validando la integridad del mensaje. El chequeo de sistema se muestra en el escenario chequear sistema.

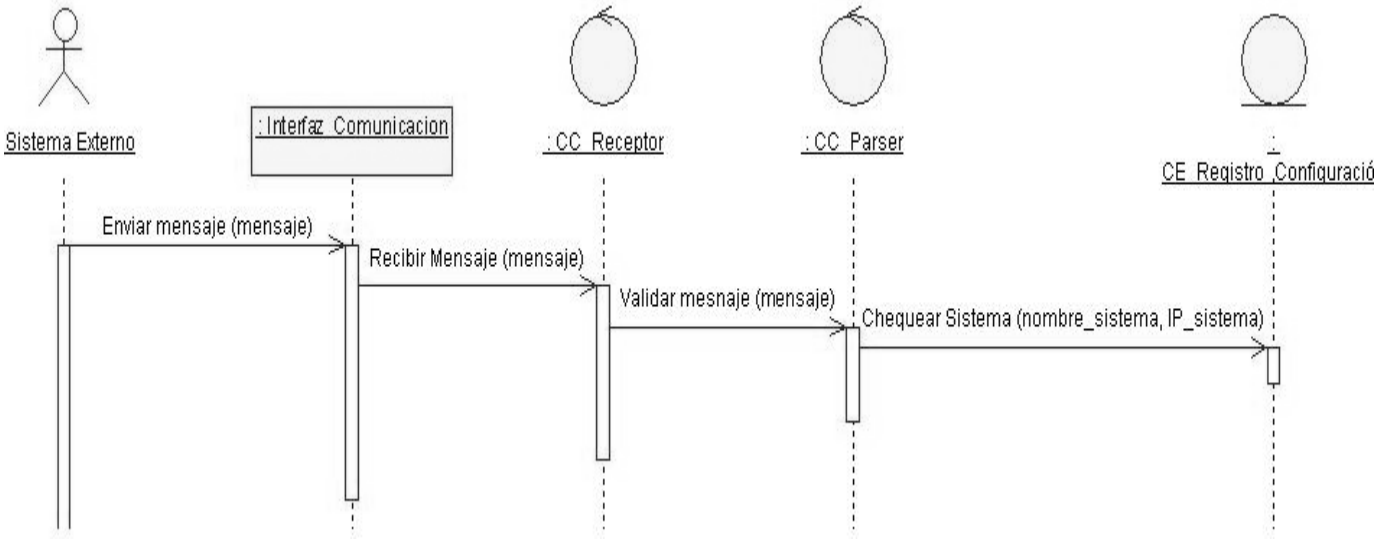



Figura.9 Escenario Chequear Sistema

En el escenario **chequear sistema** se especifican las clases que intervienen el flujo del diseño para permitir verificar si el sistema emisor del mensaje o el destinatario del mismo están configurados en el servidor de integracion como sistemas existentes.

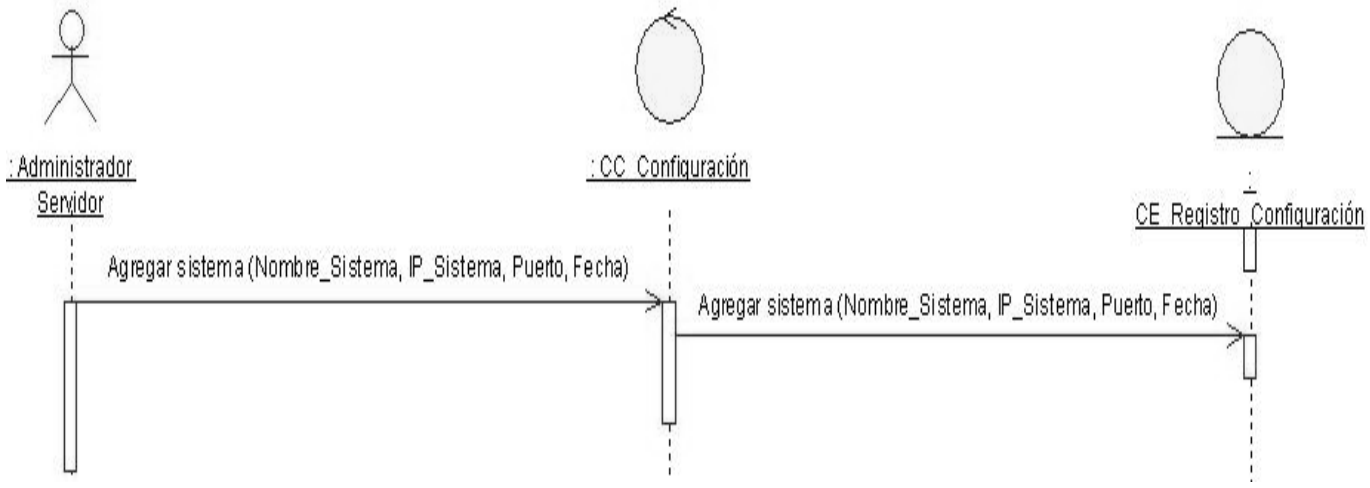



Figura.10 Escenario Agregar Sistema.

El escenario **agregar sistema** permite comprender una de las funcionalidades más importantes del servidor de integración mostrando las clases que intervienen en el flujo de este proceso.

### 2.2.1.3 Vista de Implementación

Esta vista describe la descomposición del software en capas y subsistemas de implementación. También provee una vista de la trazabilidad de los elementos de diseño de la vista lógica, pero ahora para la implementación.

**Descripción Global**

En esta descripción global se muestra de forma general como están distribuidos los paquetes de componentes por capas.

El sistema se divide en dos capas, cada capa va a contener un paquete de componentes. Cada paquete agrupa a su vez un conjunto de componentes que son los que van a dar solución al problema. (Ver Figura. 11)

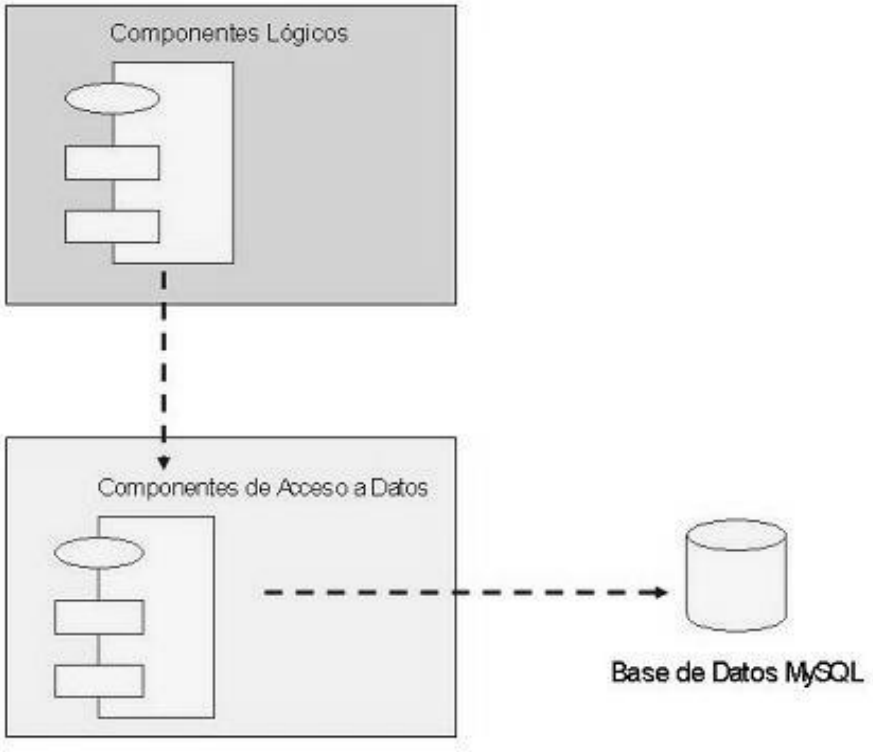



Figura.11 Paquetes de Componentes por capas

**Capas**

En esta apartado se muestra como se implementan los componentes físicos del sistema agrupándolos en capas.

**Capa Lógica.**

Los componentes que intervienen en la lógica de la solución del problema están agrupados en la capa lógica. (Ver Figura. 12)

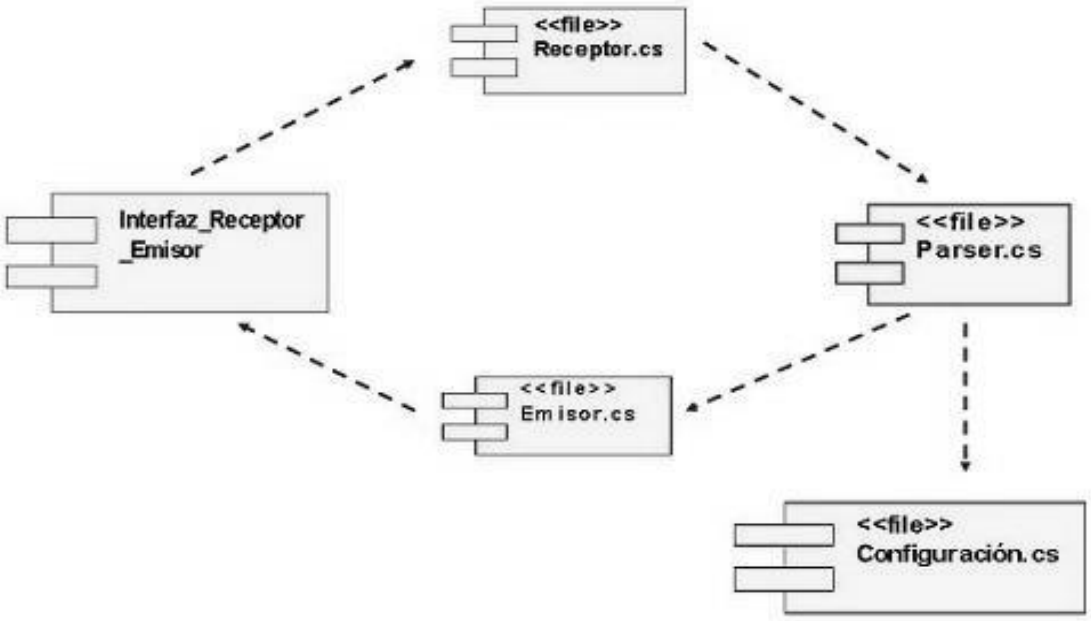



Figura.12 Diagrama de Componentes del paquete Componentes Lógicos.

Componentes que están agrupados en el paquete **Componentes Lógicos**:

Componente **Interfaz_Receptor_Emisor**: Es el componente que permite recibir y enviar los mensajes.

Componente **Emisor.cs**: Es el componente que implementa la clase controladora CC_Emisor.

Componente **Receptor.cs**: Componente que implementa la clase controladora CC_Receptor.

Componente **Parser.cs**: Este componente implementa la clase controladora CC_Parser. Es el componente más importante porque en él se realizan operaciones claves e imprescindibles para el servidor de integración.

Componente **Configuracion.cs**: Componente que implementa las funcionalidades de la clase controladora CC_Configuracion.

**Capa de Acceso a Datos**

Los componentes que interactúan con la base de datos, es decir los que se encargan del acceso de los datos a la misma fueron agrupados en la capa de acceso a datos. (Ver Figura. 13)

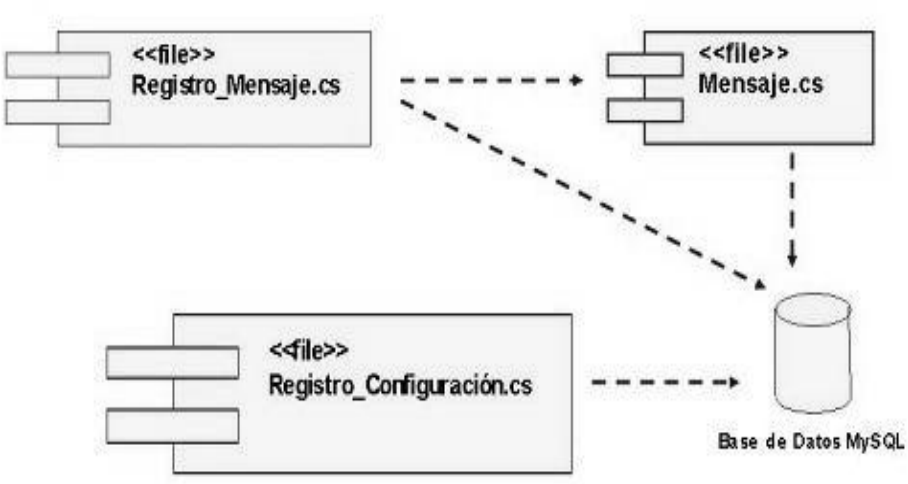



Figura.13 Componentes distribuidos en la capa de acceso a datos.

Componentes que están presentes en la capa de acceso a datos.

Componente **Registro_Mensaje.cs**: Componente que implementa la case entidad CE_Registro_Mensaje.

Componente **Mensaje.cs**: Es el componente encargado de implementar la clase entidad CE_Mensaje.

Componente **Registro_Configuración.cs**: Este componente implementa la clase entidad CE_Registro_Configuración.

**Relación entre las capas**

Los componentes de las capas en las que se han dividido el sistema para la mejor organización y nivelación del trabajo están muy relacionados. Los componentes de la capa de nivel superior (capa lógica) solo pueden usar los componentes de la capa inmediata inferior (capa de acceso a datos), pero no ocurre lo contrario. Los componentes de la capa inferior brindan servicios a los componentes de la capa inmediata superior. (Ver Figura. 14)

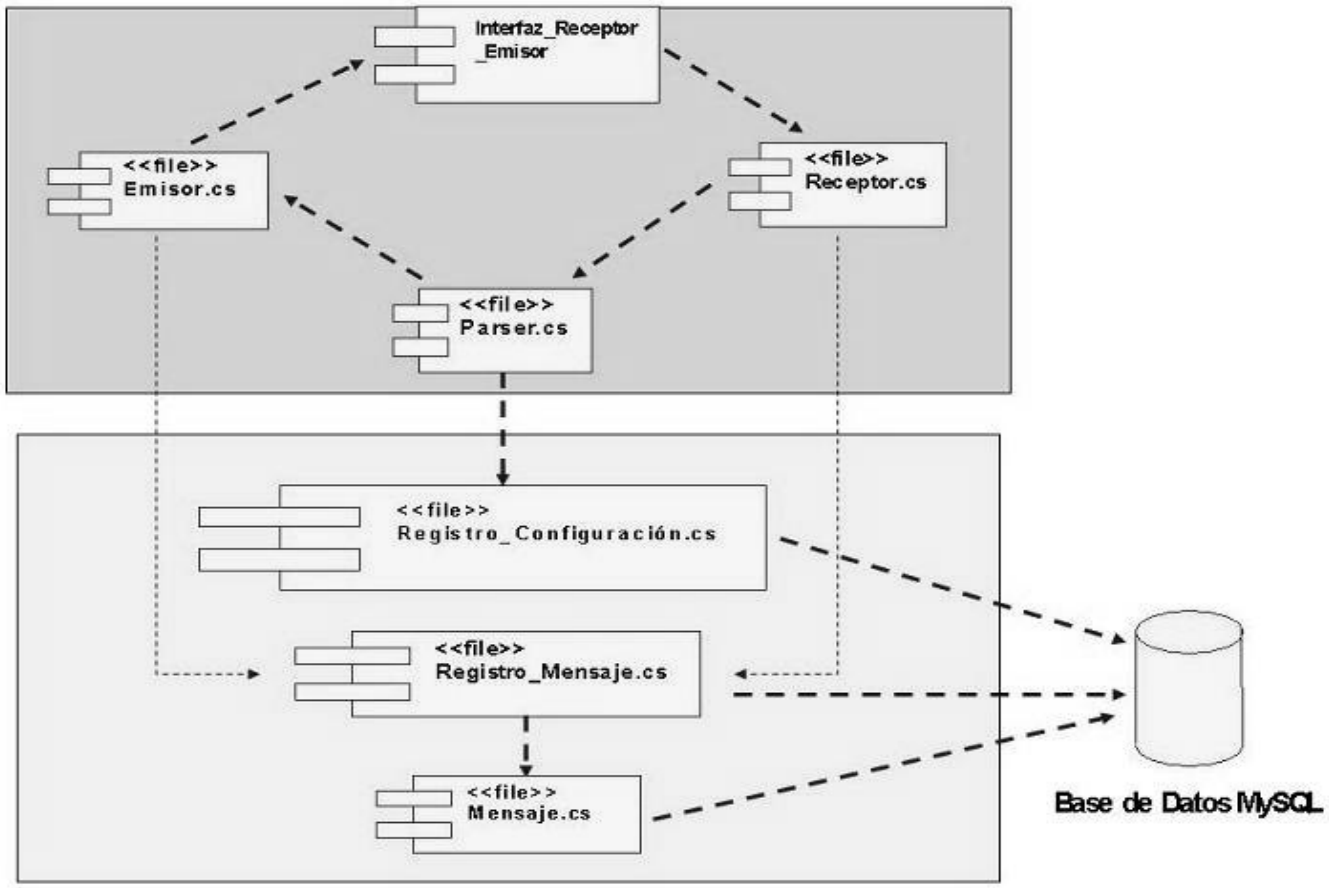



Figura.14 Relación entre las capas.

### 2.2.1.4 Vista de Despliegue

Mediante esta vista se obtiene una base para la comprensión de la distribución física de un sistema a través de nodos. Suele usarse cuando el sistema esta distribuido, y hay una traza directa del modelo de implementación puesto que cada componente físico debe estar almacenado en un nodo, esto incluye también la asignación de tareas provenientes de la vista de procesos en los nodos. (Ver Figura. 15)

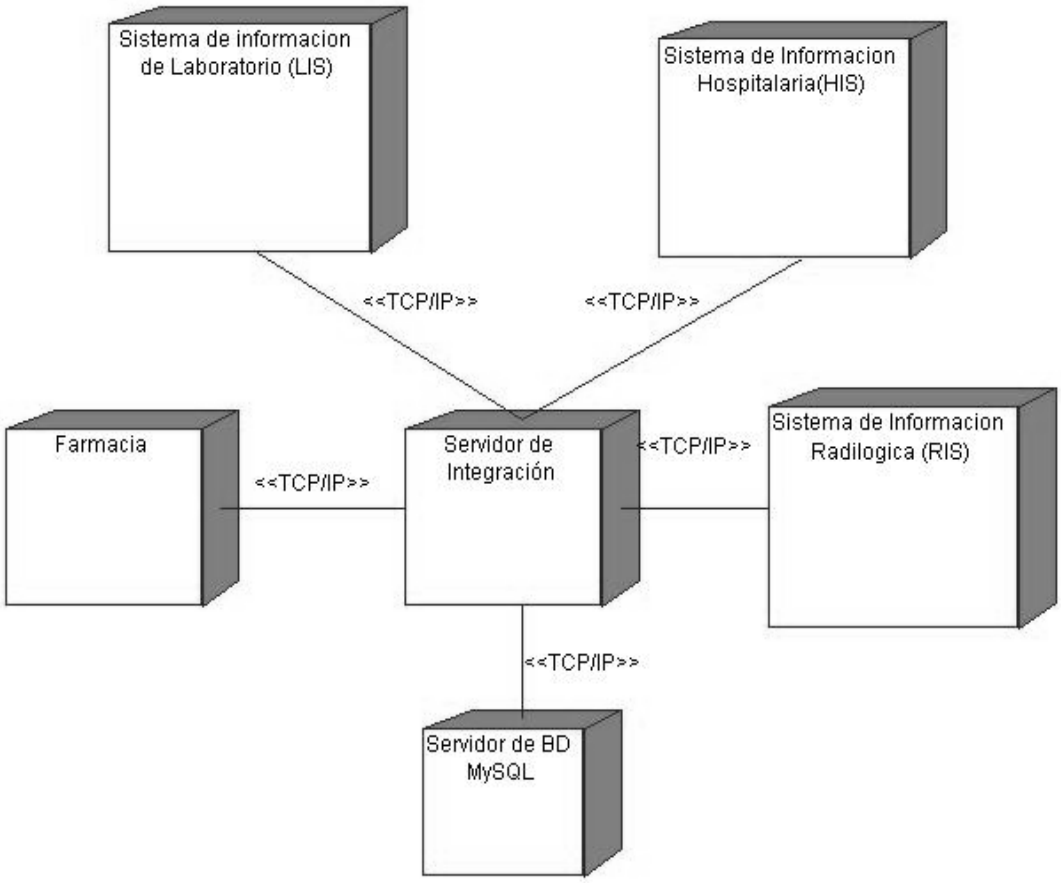



Figura.15 Modelo de Despliegue.

Nodo **Servidor de Integración:** En este nodo se encuentran las funciones principales del servidor de integración.

Nodo **Servidor de BD MySQL:** En el se encuentra la BD que permite almacenar información importante que el servidor de integración debe ser capaz de guardar.

Nodo **Sistema de Información Hospitalaria:** Sistema que se comunica con el resto de los sistemas a través del servidor de integración.

Nodo **Sistema de Información Radiológica:** Sistema que se comunica con el resto de los sistemas a través del servidor de integración.

Nodo **Sistema de Información de Laboratorio:** Sistema que se comunica con el resto de los sistemas a través del servidor de integración.

Nodo **Farmacia:** Sistema que se comunica con el resto de los sistemas a través del servidor de integración.

Estos nodos se van a comunicar mediante el protocolo de comunicación TCP/IP.

**2.2.1.5 Vista de Procesos**

Esta vista suministra una base para la comprensión de la organización de los procesos de un sistema, ilustrados en el mapeo de las clases y subsistemas en procesos e hilos. Solo suele usarse cuando el sistema presenta procesos concurrentes o hilos.

En este caso la solución propuesta no presenta procesos concurrentes por tanto no se hace representacion de esta vista.

### 2.2.2 Patrones de arquitectura y estilos arquitectónicos presentes en la solución del problema

#### 2.2.2.1 Patrón de Arquitectura en Capas

El Patrón de Arquitectura en Capas permite descomponer la aplicación en capas independientes, que son ordenadas con un nivel jerárquico. Cada capa puede usar solo los servicios que brinda la capa inmediata inferior y brindar servicio a la capa inmediata superior.

Es muy importante conocer que un número excesivo de capas o niveles a la hora de diseñar una aplicación puede resultar ineficiente: pero también es importante no descomponer la aplicación en un número muy bajo de capas o niveles porque esto puede hacer de las aplicaciones diseños complejos y pocos eficaces.

En el caso particular del servidor a desarrollar es necesario descomponer la aplicación en dos capas o niveles; la capa lógica y la capa de acceso a datos. (Ver Figura. 16)

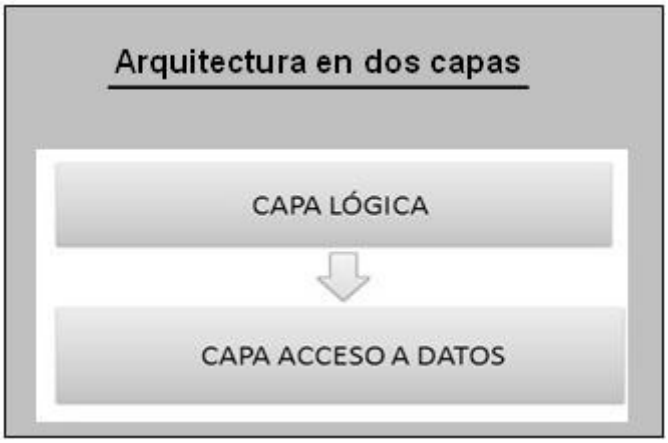



Figura.16 División de Capas.

La distribución de los componentes en cada capa será la siguiente:

**Capa Lógica:** En esta capa se encapsulan los componentes que intervienen en la lógica de la solución al problema. (Ver Figura.12 Componentes distribuidos en la capa lógica).

**Capa de Acceso a Datos:** Esta capa contiene los componentes que interactúan con la base de datos. *Ver* (Fig. 13 Componentes distribuidos en la capa de acceso a datos).

La arquitectura en capas tiene algunas ventajas: permite ganar en organización del sistema, además el patrón soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. Proporciona una amplia reutilización y facilita la corrección de errores.

### 2.2.2.2 Estilo Arquitectónico Tuberías y Filtros

Cuando al proceso de entrega de mensajes van asociadas otras tareas a un lado y otro del canal, una buena solución es estructurar la entrega de los mensajes en varios pasos.

El estilo de arquitectura tuberías y filtros, se basa precisamente, en dividir todo el proceso asociado a la entrega de mensajes en pequeñas tareas que serán implementadas en unos componentes llamados filtros que estarán interconectados entre sí a través de tuberías o canales. Al tener divididas las tareas en componentes independientes vamos a poder construir infinidad de combinaciones de cadena de procesamiento simplemente añadiendo, quitando u ordenando en una nueva secuencia todos estos elementos reutilizando cada una de las funcionalidades en cada uno de los pasos.

Cada filtro recibe los mensajes a través de su canal o tubería de entrada, los procesa y posteriormente los deposita en su tubería de salida. Este canal de salida coincidirá con el canal de entrada del siguiente filtro que continuará a su vez con la cadena de tareas.

En el caso específico del servidor de integración el proceso está dividido en tres tareas fundamentales: recibir el mensaje, validarlo y enviarlo. Estas tareas constituyen los filtros por los cuales va a transitar el mensaje HL7. (Ver Figura. 17)


Figura.17 Estilo Arquitectónico Tuberías y Filtros aplicado al servidor.

#### 2.2.2.3 Arquitectura Orientado a Servicios (SOA)

La primera y más obvia ventaja que acompaña a una estrategia de integración mediante la implementación de una arquitectura SOA sería la comunicación de manera automática, es decir sin intervención humana, de todas y cada una de las aplicaciones con las que contamos y que son necesarias en cada proceso.

Otra gran ventaja que se puede obtener de la arquitectura SOA es la comunicación asíncrona entre las diferentes aplicaciones, lo que significa principalmente, que cuando un sistema solicita a otro un determinado procesamiento, no es necesario que el primero de ellos quede bloqueado a la espera de una respuesta.

Es importante señalar, una preocupación que trae consigo la implementación de SOA, es el hecho de que con toda seguridad cada sistema va a manejar un modelo de información con distinto formato y contenido.

Esta circunstancia obliga a nuestro servidor,  no solo a comunicar unos sistemas con otros sino hacerlo de manera que se entiendan.

### Conclusiones

En este capítulo se desarrollaron los modelos, que según RUP, son más importantes para la descripción de la arquitectura. Además se explicó como se aplica el patrón de arquitectura en capas y el estilo tuberías y filtros para dar solución del problema planteado.

Finalmente quedó establecida la arquitectura del servidor de integración propuesto para lograr la comunicación entre los sistemas sanitarios existentes en un hospital.

# CONCLUSIONES

Durante el desarrollo de la investigación se cumplieron las tareas y objetivos propuestos:

- Se propuso el diseño de la arquitectura para un servidor de integración que permita la comunicación entre los sistemas de información existentes en un hospital, proponiendo, lenguaje, patrones de arquitectura, herramientas de desarrollo, etc.

- Se analizó bibliografía referente a la integración de sistemas sanitarios, lo que permitió conocer el estado del arte asociado al tema.

- Se evaluaron las metodologías, tecnologías y tendencias actuales, lo que permitió la selección de las herramientas que más se adecuan a las necesidades para el desarrollo del servidor de integración.

- Se realizó un estudio sobre IHE como iniciativa de integración de las aplicaciones sanitarias, lo que permitió la elección del uso del estándar HL7 para la comunicación entre sistemas.

- Se realizó un estudio sobre HL7 como estándar de formato de los mensajes para las comunicaciones entre sistemas de información en el sector de la salud.

- Se fundamentaron los patrones de arquitectura que más se ajustan a la solución del problema.

# RECOMENDACIONES

- Se recomienda continuar la investigación con la implementación del servidor de integración propuesto por la importancia que tiene en la informatización del Sistema Nacional de Salud.

- Transformación de mensajes en versión V2.XML a versión 2.x para que la comunicación entre un sistema V2.XML compatible y uno V2.x compatible sea total.

- Para futuras implementaciones del software se recomienda utilizar HL7 en su versión 3.x. Esta versión del estándar permite el uso de los servicios web.

- En caso de incorporar grandes cantidades de información a la base de datos propuesta, se recomienda utilizar PostgreSQL como SGBD.

- El sistema ha sido propuesto con el fin lograr la integración entre la aplicaciones dentro de un hospital, el diseño pudiera ser usado para la comunicación de aplicaciones extra hospitalarias.

## REFERENCIAS BIBLIOGRAFICAS

[1] Pérez, Noel, Arnaiz, Anay y Echevarria, Eduardo. Informatización de la sociedad en el mundo, América y en la región. Disponible en: http://www.monografias.com/trabajos39/informatizacion-sociedad/informatizacion-sociedad2.shtml

[2] A/S Jorge Parrillo, A/S Marcelo Bondarenco. Estándares de intercambio de información médica Implementación de un Broker de HL7. Simposio Argentino de Informática y Salud - SIS 2006.

[3] Delgado Ramos, Ariel y Vidal Ledo, María. Informática en la salud pública cubana. Rev Cubana Salud Pública. Julio-Sept. 2006, vol.32, no.3 [citado: 25 Abril 2007], Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-34662006000300015&lng=en&nrm=iso>. ISSN 0864-3466.

[4] OMS, El establecimiento de sistemas de información en servicios de atención de salud. ISBN 92 75 12266 0.

[5] Arquitectura de Software. Disponible en: http://es.wikipedia.org/wiki/Arquitectura_software

[6] .NET, Disponible en Disponible en: http://es.wikipedia.org/wiki/.NET#.NET_Framework

[7] Reynoso, Carlos y Kiccillof, Nicolas. Estilos y patrones en la Estrategia de Arquitectura de Microsoft. Disponible en: http://www.willydev.net/descargas/prev/Estiloypatron.pdf

[8] Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#6

[9] Arquitectura orientada a servicios. Disponible en: http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

[10] Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#11

[11] PHP. Disponible en: http://es.wikipedia.org/wiki/PHP

[12] Curso de Certificación y Examen HL7 v2.5. 2007. Disponible en: http://www.hl7spain.org/VerPagina.asp?IDPage=0

[13] Charte Ojeda, Francisco. Cómo crear un controlador de servicios Windows con Visual Basic .NET. [Online]. Febrero de 2002. Disponible en: www.micsosoft.com

## BIBLIOGRAFIA

Active Server Pages. Disponible en la World Wide Web: http://es.wikipedia.org/wiki/Active_Server_Pages

Acceso Instantáneo a Imágenes Médicas en cualquier momento y lugar. Dimensión Informática (Grupo Azertia). Vol 2, No 7 (2006). Disponible en: www.RevistaeSalud.com.

Alegre, Luis Y Cabrer, Miguel. El Hospital Son Llàtzer como plataforma para proyectos de Salud. Disponible en: www.RevistaeSalud.com

Arquitectura de Software. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/arquitectura_soft.asp

ASP.NET. Disponible en: http://es.wikipedia.org/wiki/ASP.NET

Barbero Rodríguez, Octavio. IHE. Qué es y por qué es necesario. Presente y futuro de la iniciativa IHE. Foro de Normalización, Valencia. Mayo 2005.

Base de Datos. Disponible en: http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_dato

Booch, G.: Rumbaugh, J. y Jacobson, I. “El Proceso Unificado de Desarrollo de Software”, Volumen I.

Cerritos, Antonio, Fernández, Francisco J. y Gatica Lara, Florina. Sistema de Información Hospitalaria. Universidad Nacional Autónoma de México, Facultad de Medicina, 2003.

C++. Disponible en: http://es.wikipedia.org/wiki/C%2B%2B

Delgado Ramos, Ariel y Vidal Ledo, María. Informática en la salud pública cubana, 2006.

Expertos promueven el uso de estándares abiertos en informática. Rev. La Flecha. 03 mayo 2006. Disponible: www.laflecha.net

Familia de protocolos de Internet. Disponible en:

Gallego, Carlos. HL7 La gestión de un estándar y su adaptación al ámbito nacional.

Georgas, John C., Dashofy, Eric M., y Taylor, Richard N. Desarrollo Centrado en la Arquitectura: Un acercamiento diferente a la Ingeniería de Software. Disponible en: http://www.acm.org/crossroads/espanol/xrds12-4/arqcentric.html (2004)

Historia Clínica Informatizada. Revista el Espejo Sanitario. Julio-2001. , Nro4

HL7. Disponible en http://www.hl7spain.org

Integrating the Healthcare Enterprise, IHE Technical Framework. Integration Profiles. Mayo-2006.Volume I.

Integrating the Healthcare Enterprise, IHE Technical Framework. Transactions. Mayo 2006. Volume II.

Integrating the Healthcare Enterprise, IHE Technical Framework. Transactions. Mayo 2006. Volume III

Lenguaje de programación. Disponible en: http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C

M.E. Stoopen, V. Barois Bolullard, K. K. Fujikami, J. Saavedra Abril, E. Wolpert Barraza. ¿En qué beneficia al Radiólogo y al Clínico un PACS?. Disponible en: http://sintesisrx.com/Revistas/N%BA1-2005/tecnicas.htm

Mandoza Sánchez, María A. Metodologías De Desarrollo De Software. Junio 7 del 2004. Disponible en: www.informatizate.net

Manual de Usuario IHE-Radiología, junio 2005. Disponible en:http://www.ihe-e.org/documentos/Manual%20de%20Usuario%20IHE-Radiologa.pdf

Moreno Sánchez, Luís. Utilización de Patrones para la Construcción de Arquitecturas Orientadas a Servicios. Disponible en: http://www.luismor.com/UtilizaciondePatronesParaLaConstruccionde ArquitecturasOrientadasAServicios.pdf

Panorámica del sistema de gestión de base de datos MySQL. Disponible en: http://dev.mysql.com/doc/refman/5.0/es/what-is.html

Patxi Echarte, Introducción a la plataforma .NET y Mono. Disponible en: http://www.wikilearning.com/proyecto_mono-wkccp-11990-3.htm

Perl. Disponible en: http://es.wikipedia.org/wiki/Perl

Proyecto Mono. Disponible en la World Wide Web: http://es.wikipedia.org/wiki/Proyecto_Mono

Rational Rose: Procedimientos básicos para desarrollar un proyecto con UML. Disponible en: http://www.vico.org/TallerRationalRose.pdf

Robles Viejo, Montserrat. Estandarización de la arquitectura de información para la comunicación de la HCE. Universidad Politécnica de Valencia.

Rodríguez Cesar, Cira. BOLETIN - Instituto de Información Científica y Tecnológica. Enero 2005. No. 2 ISSN 1028-0855.

Servicio Web. Disponible en: http://es.wikipedia.org/wiki/Servicio_Web

Sistema de información. Disponible en: http://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n
Villagrasa, Jesús. HL7 Versión 2.x 'Manos a la Obra'.

SharpDevelop. Disponible en: http://es.wikipedia.org/wiki/SharpDevelop

Villagrasa, Jesús. Detalles generales del estándar de mensajería HL7 V 2.X.

WSLD. Disponible en: http://es.wikipedia.org/wiki/WSDL

XML. Disponible en: http://es.wikipedia.org/wiki/XML#Ventajas_del_XML.

XML Schema. Disponible: http://es.wikipedia.org/wiki/Schema

# ANEXOS

**ANEXO I:** Estructura de los mensajes ADT.

El conjunto de mensajes ADT transmite datos que contienen información demográfica sobre pacientes, así como información sobre los eventos de resignación, admisión, derivación interna y externa, alta y visitas de los mismos. Dado que virtualmente todos los programas de aplicación asimilados a una red hospitalaria requieren información sobre pacientes, esta variedad de transacciones es una de las más frecuentemente observadas.

Generalmente, la información que identifica al paciente y a los motivos de su acceso al sistema es ingresada a través de un sistema específico de registro, y trasladada desde allí a otros sistemas: de historia clínica, de registro de servicios auxiliares, laboratorio, radiología, de facturación, y otros.

Uno de los mensajes más simples, es el de admisión de un paciente (A01), consta de cuatro segmentos obligatorios y 13 segmentos opcionales.

En el caso específico de un mensaje ADT de la variedad A01, los segmentos obligatorios corresponden al encabezado del mensaje (MSH), al codificador de evento (EVN), al segmento que contiene información sobre el paciente (PID) y al que codifica los datos correspondientes al proceso de visita (PV1).

Los campos constitutivos de los segmentos de existencia obligada en un mensaje ADT de la variedad A01. La composición de los segmentos no es específica para cada mensaje, sino que se mantiene a través de todo el estándar. Por lo tanto, la descripción de los segmentos que sigue a continuación es válida para cualquier mensaje en donde el segmento específico aparezca.

**El segmento de encabezamiento del mensaje (MSH - Message Header).**

El segmento de encabezamiento del mensaje define el propósito del mismo, a su remitente y destinatario, y a los detalles de sintaxis de la totalidad del mensaje al igual que la totalidad de los segmentos que componen el HL7.

**Campos del segmento mensaje (MSH - Message Header) y sus Funciones**

| | |
|---|---|
| 1. Separador de campo (caracter codificador) | Identifica al caracter que indicará el final de un campo y comienzo de otro. |
| 2. Caracteres de codificación | Este campo tiene una extensión precisa de 4 espacios, cada uno de los cuales contendrá, respectivamente:<br>Caracter separador de componentes de campo **'^'.**<br>Carácter codificador de segmentos repetitivos **'~'.**<br>Carácter de escape **'\'.**<br>Carácter separador de subcomponentes de campo **'&'**. |
| 3. Aplicación que envía el mensaje. | Cadena de caracteres (generalmente un código o sigla) que identifica al sistema que envía el mensaje. |
| 4. Sitio que envía el mensaje. | Cadena de caracteres (también generalmente un código o sigla) que identifica al sitio de la institución que envía el mensaje. |
| 5. Aplicación a la cual va dirigido el mensaje. | Cadena de caracteres que identifica al programa destinatario del mensaje. |
| 6. Sitio al cual va dirigido el mensaje. | Cadena de caracteres que identifica al sitio destinatario del mensaje. |
| 7. Fecha y hora del mensaje | Fecha y hora de creación del mensaje, conforme al formato especificado por el HL7 (AAAAMMDDHHMMSS). |
| 8. Tipo de mensaje. | Código que identifica al tipo y/o variedad de mensaje. En los casos en que contiene tipo y variedad, estos |

| | |
|---|---|
| | componentes están separados por su separador respectivo. |
| 9. Código de identificación del mensaje. | Número o código que identifica inequívocamente al mensaje. Este campo es utilizado por el sistema destinatario del mensaje para elaborar la respuesta. |
| 10. Identificación de tipo de procesamiento. | Identifica mediante un tipo de procesamiento que deberá aplicarse al mensaje. |
| 11. Versión de HL7. | Identificador de la versión del HL7 que esta siendo utilizado |
| 12. Puntero de continuación. | Identificador de una continuación del mensaje. |
| 13. Tipo de acuse de recibo de aceptación. | Condiciones bajo las cuales los acuses de recibo de aceptación del sistema destinatario son requeridas. |
| 14. Tipo de acuse de recibo de aplicación. | Condiciones para recibir acuses de recibo en respuesta al mensaje. |
| 15. Código de país. | Identifica al país mediante un código de tres letras especificado por la ISO |

**El segmento de Evento del mensaje (EVN - Event).**

El segmento de Evento del mensaje, indicado por la aparición de EVN al comienzo del segmento, comunica información sobre un evento particular en un proceso de atención de la salud referenciado por el mensaje y siempre señala una variedad de alguno de los mensajes HL7.

**Campos del segmento EVNy sus Funciones**

| | |
|---|---|
| 1. Código del tipo de evento. | Identifica al tipo de evento según una tabla predefinida por el HL7 |
| 2. Fecha y hora del evento. | Señala la fecha y hora en que fue generado el evento. |
| 3. Fecha y hora de un evento planeado. | Se recomienda su uso, por ejemplo, para señalar el momento para el cual se planea el alta |
| 4. Codificador de la razón del evento. | Describe la razón por la cual se generó el evento. |
| 5. Identificación del operador. | Contiene un código que identifica a la persona que generó el evento. |

El campo E marca el final del segmento EVN.

**Segmento de Identificación de Paciente del mensaje (PID - Patient Identification).**

El segmento de referencia contiene información demográfica acerca del paciente, y es utilizada virtualmente por todos los sistemas de una red hospitalaria para transmitir la información que permite identificar al mismo.

En este documento no se describen en detalle los campos dado que muchos son autoexplicativos. Solo interesa destacar que algunos contienen varios componentes, delimitados por sus respectivos separadores. También importa señalar que si bien el número de campos es alto, solo son obligatorios en el mensaje los campos 3 y 5 (identificación interna, por ejemplo el número de HC y el nombre del paciente).

**Campos que contienen información demográfica del paciente:**

| | |
|---|---|
| 1 | Identificador de segmentos repetidos. |
| 2 | Identificación externa del paciente. |
| 3 | Identificación interna del paciente. |
| 4 | Identificación alternativa del paciente. |
| 5 | Nombre del paciente. |
| 6 | Apellido de la madre. |
| 7 | Fecha de nacimiento. |
| 8 | Sexo. |
| 9 | Alias del paciente. |
| 10 | Raza. |
| 11 | Domicilio del paciente. |
| 12 | Código de país. |
| 13 | Número de teléfono del hogar del paciente. |
| 14 | Número de teléfono del empleo del paciente. |
| 15 | Lenguaje nativo del paciente. |
| 16 | Estado civil. |
| 17 | Religión. |

| | |
|---|---|
| 18 | Número de cuenta del paciente. |
| 19 | Número de seguridad social del paciente. |
| 20 | Número de licencia de conductor del paciente. |
| 21 | Identificador de la madre. |
| 22 | Grupo étnico. |

**Segmento Visita de Paciente del mensaje (PV1 - Patient Visit).**

El segmento en cuestión contiene la información referente a una o más visitas del paciente a la institución.

**Campos y descripciones del segmento PV1**

| CODIGO | DESCRIPCION |
|---|---|
| 1 | Identificador de segmentos repetidos |
| 2 | Clase de paciente |
| 3 | Localización asignada al paciente |
| 4 | Tipo de admisión |
| 5 | Número de pre-admisión |
| 6 | Localización previa del paciente |
| 7 | Médico de cabecera |

| | |
|---|---|
| 8 | Médico que deriva |
| 9 | Médico consultor |
| 10 | Servicio hospitalario a recibir por el paciente. |
| 11 | Localización temporaria. |
| 12 | Indicador de exámenes de pre-admisión. |
| 13 | Indicador de re-admisión. |
| 14 | Lugar de admisión del paciente. |
| 15 | Estado ambulatorio. |
| 16 | Indicador VIP |
| 17 | Médico que recibe. |
| 18 | Tipo de paciente. |
| 19 | Número de visita. |
| 20 | Clase financiera. |
| 21 | Indicador de escala de precio. |
| 22 | Código de cortesía. |
| 23 | Índice de crédito. |

| | |
|---|---|
| 24 | Código de contrato. |
| 25 | Fecha de comienzo del contrato. |
| 26 | Monto del contrato. |
| 27 | Duración del contrato. |
| 28 | Código de interés. |
| 29 | Indicador de transferencia a cuenta observada. |
| 30 | Fecha de transferencia a cuenta observada. |
| 31 | Código de agencia de cuenta observada. |
| 32 | Suma transferida a cuenta observada. |
| 33 | Suma obtenida del garante de cuenta observada. |
| 34 | Indicador de cuenta discontinuada. |
| 35 | Fecha de cuenta discontinuada. |
| 36 | Estado al alta. |
| 37 | Lugar de destino del paciente al alta. |
| 38 | Tipo de dieta. |
| 39 | Sitio que provee el servicio. |

| | |
|---|---|
| 40 | Estado de la cama. |
| 41 | Estado de la cuenta. |
| 42 | Localización en espera. |
| 43 | Localización temporaria previa. |
| 44 | Fecha y hora de admisión. |
| 45 | Fecha y hora de alta. |
| 46 | Estado actual de la cuenta. |
| 47 | Cargo total. |
| 48 | Ajustes totales. |
| 49 | Pagos totales. |
| 50 | Número de identificación de visita alternativo. |

En el caso de este segmento, solo son obligatorios los campos 2 y 3, que indican la clase de paciente (de emergencia, externo, interno, obstétrico, etc.) y el sitio asignado (habitación/cama, consultorio, etc.).

Como habrá podido observarse, el mensaje tiene una estructura sumamente sencilla en términos electrónicos, y es fácilmente transportable hacia e interpretable virtualmente por cualquier sistema o aplicación. La forma en la cual se transmita no está específicamente indicada en el estándar y deberá ser acordada entre los sitios emisor y receptor del mensaje.

**ANEXO II*:*** Eventos de los mensajes ADT.

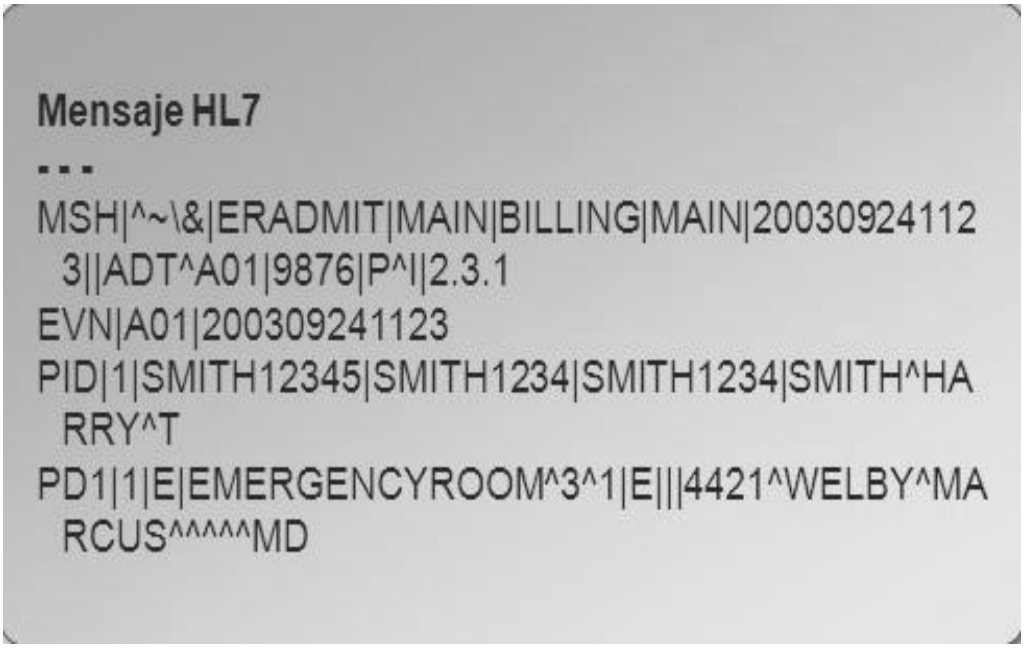
Según el estándar HL7, el conjunto de mensajes ADT codifica 37 eventos diferentes.

| **Evento** | **Función** |
|---|---|
| A01 | Admisión de paciente. |
| A02 | Transferencia de paciente. |
| A03 | Alta de paciente. |
| A04 | Registración de paciente. |
| A05 | Pre- admisión de paciente. |
| A06 | Transferencia de internado a ambulatorio. |
| A07 | Transferencia de ambulatorio a internado. |
| A08 | Actualización de información de paciente. |
| A09 | Paciente saliendo. |
| A10 | Paciente arribando. |
| A11 | Cancelación de admisión de paciente. |
| A12 | Cancelación de transferencia de paciente. |
| A13 | Cancelación de alta de paciente. |
| A14 | Admisión de paciente pendiente. |
| A15 | Transferencia de paciente pendiente |

| | |
|---|---|
| A16 | Alta de paciente pendiente. |
| A17 | Intercambio de pacientes. |
| A18 | Unión de información de paciente. |
| A19 | Consulta sobre paciente. |
| A20 | Actualización de estado de cama. |
| A21 | Paciente sale en "salida con permiso". |
| A22 | Paciente retorna de "salida con permiso". |
| A23 | Borrar registro de paciente. |
| A24 | Enlace de información sobre paciente. |
| A25 | Cancelación de alta de paciente pendiente. |
| A26 | Cancelación de transferencia de paciente pendiente. |
| A27 | Cancelación de admisión de paciente pendiente. |
| A28 | Agregar información sobre persona. |
| A29 | Borrar información sobre persona. |
| A30 | Unir información sobre persona. |
| A31 | Actualizar información sobre persona. |
| A32 | Cancelar paciente arribando. |
| A33 | Cancelar paciente saliendo. |

| | |
|---|---|
| A34 | Enlace de información sobre paciente - solo para identificación (ID). |
| A35 | Enlace de información sobre paciente - solo para número de cuenta. |
| A36 | Enlace de información sobre paciente - solo para ID y número de cuenta. |
| A37 | Anular enlace de información sobre paciente. |

**Ejemplo de Mensaje HL7.**



# GLOSARIO DE TÉRMINOS

**API's:** Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

**Common Object Request Broker Architecture (CORBA):** Arquitectura que posibilita el intercambio de objetos entre aplicaciones. Un objeto CORBA puede estar desarrollado en cualquier lenguaje y correr en cualquier sistema operativo.

**Hyper Text Transfer Protocol (HTTP):** en español, protocolo de transferencia de hipertexto, es el protocolo usado en cada transacción de la Web. El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido. También sirve el protocolo para enviar información adicional en ambos sentidos, como formularios con campos de texto.

**Lenguaje de Modelado Unificado (UML):** es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

**Metodología:** En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

**Ministerio de Salud Pública (MINSAP):** Es el Organismo rector del Sistema Nacional de Salud, encargado de dirigir, ejecutar y controlar la aplicación de la política del Estado y del Gobierno en cuanto a la Salud Pública, el desarrollo de las Ciencias Médicas y la Industria Médico Farmacéutica.

**Parser:** Un parser es un analizador sintáctico, que en informática y lingüística es un proceso que analiza secuencias de tokens para determinar su estructura gramatical respecto a una gramática formal dada. Un parser es así mismo un programa, capaz de reconocer si una o varias cadenas de caracteres forman parte de un determinado lenguaje, es utilizado, por ejemplo en compiladores.

**Proceso de desarrollo de software:** es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software.

**Protocolo de red:** También conocido como Protocolo de Comunicación es el conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre las entidades que forman parte de una red.

**Rational Unified Process (RUP):** Es un proceso de desarrollo de software.

**Servidor Web:** Un servidor Web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. Cabe destacar el hecho de que la palabra servidor identifica tanto al programa como a la máquina en la que dicho programa se ejecuta. Un servidor Web se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP que solemos conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita.

**Servicios Web:** Un servicio Web (en inglés Web service) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, a través de la Web.

Los servicios Web pueden ser utilizados por distintas aplicaciones de software, desarrolladas en diferentes lenguajes de programación y ejecutados sobre cualquier plataforma, para intercambiar datos en redes de ordenadores como Internet.

**Sistema de Información Hospitalaria (HIS):** un Sistema de Información Hospitalaria es un sistema de información orientado a satisfacer las necesidades de generación de información, para almacenar, procesar y reinterpretar datos médico-administrativos de la cualquier institución hospitalaria. Permitiendo la optimización de los recursos humanos y materiales, además de minimizar los inconvenientes burocráticos que enfrentan los pacientes.

**Sistema de Información Radiológica (RIS):** Es el sistema informático del servicio de radiodiagnóstico que recoge, controla y explota todos los datos que se obtienen en un servicio de radiodiagnóstico. Esto incluye la cita del paciente, todos los pasos que se realizan para llevar a cabo la prueba dentro del servicio de radiodiagnóstico, e incluye tanto la realización como la distribución del informe e imagen al médico

solicitante. Es la herramienta informática que nos permite realizar los procesos de gestión de un departamento de radiología.

Todo sistema de información hospitalaria genera reportes e informes dependiendo el área o servicio para el cual se requiera, dando lugar a la retroalimentación de la calidad de la atención de los servicios de salud.

**W3C:** es la abreviatura de World Wide Web Consortium, es un consorcio internacional que produce estándares para la World Wide Web. Está dirigida por Tim Berners-Lee, el creador original de URL (Uniform Resource Locator, Localizador Uniforme de Recursos), HTTP (Hyper Text Transfer Protocol) y HTML (Lenguaje de Marcado de Hiper Texto) que son las principales tecnologías sobre las que se basa la Web.