UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

CENTRO DE INVESTIGACIÓN DE SISTEMAS DE INFORMACIÓN

PROTOTIPO DE SISTEMA AUTOMATIZADO

DE TRIAJE PARA EMERGENCIA HOSPITALARIA

Trabajo Especial de Grado presentado ante la ilustre

Universidad Central de Venezuela por el

Br. Jose Schmidt

Para optar el título de Licenciado en Computación

Tutor: Prof. Wuilfredo Rangel

Diciembre, 2010

ACTA

Quienes suscriben, miembros del Jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el bachiller Jose Gustavo Francisco Carlos Schmidt Pérez CI 16.691.912, con el título **PROTOTIPO DE SISTEMA AUTOMATIZADO DE TRIAJE PARA EMERGENCIA HOSPITALARIA** a los fines de optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 06 de diciembre del 2010 a las 11:00 am, para que sus autores lo defendieran en forma pública, lo que se hizo en el salón de post grado de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondió las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas a los 06 días del mes de diciembre del año 2010.

_____

Prof. Wuilfredo Rangel

Tutor

_____

Prof. Haydemar Nuñez

Jurado

_____

Prof. Esmeralda Ramos

Jurado

AGRADECIMIENTOS

Esta tesis de pregrado, si bien ha requerido de mucho esfuerzo por parte del autor y de su tutor, no hubiese sido su finalización sin la cooperación de todas y cada una de las personas que a continuación citaré.

Primero, dar gracias a Dios, por estar conmigo siempre y por haber puesto en mi camino que han sido mi soporte y compañía durante todo mi periodo de estudio.

A mi familia por creer siempre en mí, apoyándome en todo momento brindándome lo que necesitase a lo largo y ancho de mi carrera. A mis padres Edgar Schmidt e Isabel Belén, a mi hermano Edgar, a mi hermana Lised, a mis primos Bodo y Jorge a quienes les debo muchos momentos, y en general a todos muchísimas gracias.

A mi amada esposa Cristina, quien estuvo conmigo prácticamente toda la carrera siempre dándome ánimos para continuar y lograr dar esta primera vuelta del ciclo que hoy apenas comienza.

A mi nueva familia, Sr. Pedro, Sra. Conchita, Sra. Mercedes, Pedro José, brindándome su apoyo en esta última etapa de mi primera vuelta, al Sr. Félix quien abrió las puertas de su casa permitiéndome culminar con holgura.

A la Sra. Laila, a Flavio, a Aarón quienes me han dado tanto durante todos estos años; no estuviera aquí de no ser por Uds.

A mi tutor, Prof. Wuilfredo Rangel, quien además de acompañarme en esta jornada ha sido un gran amigo.

A mi centro de investigación, Profa. Mercy, Prof. Antonio, Profa. Tina, Profa. Mayerlin, Pasqual, Keyla, Villasana, y demás compañeros mis más sinceros agradecimientos.

A todas las personas, mis compañeros y amigos, sin Uds. no habría vivido toda esta experiencia, muchísimas gracias a todas y todos.

DEDICATORIA

Le dedico esta tesis a:
A Dios, mi familia y mi esposa, por estar siempre allí cuando los necesito.

Universidad central de Venezuela

Facultad de ciencias

Escuela de computación

Área: Sistemas de Información

Prototipo de Sistema Automatizado de Triaje para Emergencia Hospitalaria

Autor: Br. Jose Schmidt
Tutor: Wuilfredo Rangel
Fecha: Noviembre, 2010

RESUMEN

El triaje de emergencia persigue desarrollar el proceso de valorización clínica preliminar para ordenar los pacientes según el nivel de urgencia o gravedad, antes de la atención médica, de forma que, en una condición de saturación del servicio o disminución de recursos, los pacientes más graves sean tratados con prioridad. Un proceso de triaje presenta un alto grado de imprecisión o incertidumbre al momento de expresar el estado de salud de un paciente, en consecuencia puede haber dificultad y riesgo en la clasificación de las urgencias. En los servicios de emergencias de los hospitales públicos venezolanos, en especial el servicio de emergencia del Hospital Universitario de Caracas (HUC), se ha evidenciado un incremento en el número de casos que se atienden, siendo una constante la saturación del servicio, lo cual imposibilita que los pacientes con prioridad de atención, sean tratados más rápidamente de forma oportuna y eficaz. En este Trabajo Especial de Grado se construyo un prototipo Web de triaje sustentado en el modelo conceptual de datos difuso de triaje para el sector salud venezolano, basado en el proceso de triaje hospitalario de la Sociedad Venezolana de Medicina de Emergencia y Desastres (SVMED), ya que éste daría un empuje a la implementación de un Sistema Clínico de soporte a la toma de decisiones para triaje a nivel nacional tomando como eje el HUC. El sistema consta de cuatro (04) módulos: el primero es la implementación del esquema relacional difuso, el segundo es el módulo Cliente FSQL, el tercero ejecuta las consultas tanto difusas como no difusas y por último, se tiene un módulo que implementa la captura de los datos de entrada y emite el resultado de la clasificación de triaje obtenido mediante el módulo Cliente FSQL.

Palabras Claves: Triaje, Base de Datos Difusas, Sistema Clínico de soporte a la toma de decisiones.

INDICE

Índice de Figuras

Índice de Tablas

Índice de Códigos

Introducción

Los servicios médicos de urgencias en la mayoría de los hospitales están sufriendo un fuerte incremento en el número de casos que se presentan. Con el fin de usar los recursos de forma más eficiente, cada vez en más hospitales se hace imprescindible la implantación de un sistema de triaje, o sistema para la catalogación de la urgencia con la que un paciente en la sala de emergencias debe ser atendido. Existen recomendaciones o sistemas estándares de triaje (Gómez, 2003) para hacerlo. De todos ellos, el sistema canadiense CTAS (The Canadian Emergency Department Triage & Acuity Scale) de amplia difusión en el Continente Americano y el MAT (Modelo Andorrano de Triaje) implantado sobre todo en España. El MAT además se basa en categorías sintomáticas y algoritmos clínicos. Ambos utilizan 5 niveles de triaje, que del 1 al 5 son: resucitación, de emergencia, urgente, menos urgente y no urgente.

Junto con la implantación de los sistemas de triaje hospitalario han aparecido diversos sistemas computacionales para el soporte a la decisión que pretenden ayudar a la hora de decidir el nivel de urgencia que se le asigna a un paciente. Sin embargo son muy limitadas las funciones que realizan incluso cuestionándose que no son verdaderos sistemas de apoyo a la toma de decisiones, sino soportes digitales del procedimiento a seguir (Guterman, Mankovich, & Hiller, 2003). Así ocurre por ejemplo con el “Mobile Emergency Triage” (MET), desarrollado por expertos en medicina de urgencias de la Facultad de Medicina de la UCLA, el “Symptons, Advice, Measure” (SAM), desarrollado por un médico generalista y el Ped's Advice (PA), sólo para servicios de pediatría, desarrollado a nivel local por médicos y enfermeras del Hospital Académico de Niños de Upsala (Suecia). En estos dos últimos sistemas, el usuario (generalmente enfermeros) introducen una palabra referente al más notable de los síntomas, por ejemplo, tos, y el sistema proporciona un cuestionario de respuesta si/no paso a paso y finalmente el sistema recomienda un nivel de triaje. Estos dos sistemas han sido criticados por sus usuarios por no ser sistemas reales de apoyo a la decisión, dado que no están adaptados a las situaciones de la práctica diaria (Holmstrom, 2007). Cabe destacar que algunas de las limitaciones más relacionadas con la falta de precisión en el proceso de triaje es que se presenta un alto grado de subjetividad por parte del paciente al momento de expresar su sintomatología ó estado de salud, causando dificultad y riesgo en el proceso de clasificación de la prioridad de atención, debido a que este alto grado de subjetividad genera un alto grado de incertidumbre o imprecisión.

Es aquí donde cobra fuerza el ámbito de lógica difusa (Zadeh, 1965), ya que permite acercar el funcionamiento de los sistemas de información al modo de trabajo de los seres humanos, pues las personas manejan con gran frecuencia conceptos difusos (como lo es la escala de dolor: “Ligero”, “Agudo”, “Moderado”, “Intenso”...) que incluyen cierta imprecisión y que los sistemas de información tradicionales no entienden y, por tanto, no pueden utilizar. Con el fin de gestionar esta

imprecisión en bases de datos, se han propuesto en los últimos años varios trabajos sobre “Bases de Datos Relacionales Difusas” (BDRD ó BDRF), que pretenden aplicar la lógica difusa a la tecnología de las bases de datos. Dos propuestas sobresalientes de estas aplicaciones son FSQL (Fuzzy SQL, SQL Difuso) (Galindo, 1999) y SQLf (SQL fuzzy) (Bosc & Pivert, 1995).

Bajo estas tendencias de trabajo de Bases de Datos Difusas, se motivo la siguiente propuesta para implementar el Modelo de Triaje para el Sector Salud con Soporte en Bases de Datos Difusas planteado por Rangel (2010) sobre el servidor de base de datos difusas *FSQL Server*, que permite representar las Reglas o Base de Conocimiento del Dominio de Triaje y desarrollar inferencias sobre el proceso de catalogación de la urgencia.

El contenido de este trabajo se desglosa de la siguiente forma:

- Capítulo I – Motivación y Objetivos: En este Capítulo se expone la motivación que llevó a la realización de este trabajo y los objetivos a alcanzar.
- Capítulo II – Marco Conceptual: Se describen los aspectos teóricos necesarios para el desarrollo de este trabajo, esto incluye: Triaje, FSQL Server, Modelo difuso de triaje, Tecnologías Web y Metodología de desarrollo AUP.
- Capítulo III – Marco Aplicativo: Se presenta la aplicación de la metodología AUP para la realización del prototipo de sistema automatizado de triaje.

Capítulo I – Motivación y Objetivos

Los hospitales de Venezuela, en particular el Hospital Universitario de Caracas (HUC), no disponen de un proceso de triaje para la clasificación de los pacientes según el nivel de urgencia, y mucho menos sistemas automatizados de triaje. Sin embargo, la Sociedad Venezolana de Medicina de Emergencia y Desastre ha propuesto un proceso de triaje para priorizar a los pacientes dentro de la cola de la emergencia basada en 5 niveles de clasificación. Rangel (2010) propone un modelo difuso de triaje que representa al proceso propuesto por la SVMED.

En torno al contexto encontrado en los elementos expuestos, se crean una serie de interrogantes que son base fundamental para este trabajo:

1.- ¿Es posible implementar el modelo conceptual de datos difusos de triaje para emergencia hospitalaria planteado por Rangel (2010), en el servidor FSQL Server?

2. ¿Es factible desarrollar un sistema de apoyo a la gestión de triaje soportado en la implementación del modelo de triaje planteado por Rangel (2010)?

3.- ¿Es factible desarrollar este sistema utilizando la metodología UP Ágil y tecnologías Web?

En la búsqueda de respuestas a estas interrogantes, se planteo un problema de investigación fundamentado en la hipótesis de que la organización HUC no posee un proceso automatizado que permita apoyar a la toma de decisión en la clasificación de las urgencias hospitalarias.

1.1 Objetivo general

Implementar un prototipo de Sistema de Gestión de triaje, utilizando el Modelo conceptual de datos difusos de triaje para emergencia hospitalaria.

1.2 Objetivos específicos

- Analizar los requerimientos funcionales del proceso de triaje de la SVMED, basados en el modelo conceptual de datos difusos triaje.
- Diseñar los modelos arquitectónicos del sistema: la arquitectura de Bases de Datos, Componentes de Software, Casos de Uso.

- Implementar el modelo relacional de datos difusos para triaje utilizando el servidor Oracle 8i y el Fuzzy SQL Server.
- Desarrollar los componentes de la aplicación utilizando el lenguaje de programación PHP.
- Desarrollar los casos de prueba que den validez al sistema.

Capítulo II - Marco Conceptual

En este capítulo se exponen los aspectos teóricos concernientes a servicios de emergencia, triaje (definición, contextualización, tipos, modelos de triaje, modelo de triaje en Venezuela), Sistema de Apoyo a la Toma de Decisiones, Lógica Difusa (Definición, Funciones de membrecía), aplicaciones de Lógica Difusa en bases de datos y Tecnologías Web (Páginas Web, Arquitecturas de desarrollo para aplicaciones Web) y la metodología de desarrollo AUP.

2.1 Servicios de Emergencia

Según (Rodríguez, 2007) los servicios de emergencia hospitalarios son unidades diseñadas para proporcionar tratamiento médico altamente especializado, con disponibilidad inmediata de recursos especiales a pacientes que requieran (o necesiten) cuidados de emergencia, a cualquier hora del día o de la noche. A pesar de esto, es necesaria la clasificación de los pacientes de acuerdo a la gravedad de sus síntomas, para de esta forma no generar esperas innecesarias, aglomeración o sobrecarga del servicio de emergencia y gastos excesivos e innecesarios en estudios complementarios.

2.2 Triaje

La palabra triaje o en inglés *triage* es un derivado de la palabra en francés *trier* que significa 'para ordenar'. Es un proceso de valoración clínica preliminar que ordena los pacientes antes de la valoración diagnóstica y terapéutica en base a su grado de urgencia, de forma que ante una situación de saturación del servicio o de disminución de recursos, los pacientes más urgentes son tratados primero y el resto controlados y reevaluados continuamente (Gómez, 2009).

El concepto de triaje fue introducido por Baron Dominique-Jean Larrey, Jefe de los cirujanos de la Guardia Imperial de Napoleon Bonaparte alrededor de los años 1810 (Larrey, 1987), el pensó que atender a los heridos durante las batallas sería beneficioso para ganar las guerras, planteando las primeras reglas del triaje, que son en primer lugar atender a los gravemente heridos y, en segundo lugar, los menos graves deben esperar.

Actualmente Iserson & Moskop (2007) expresan que existen varios tipos de triaje que se definen según su aplicabilidad, estos son: Triaje del Departamento de Emergencia, Triaje de Hospitalización, Triaje por Incidentes, Triaje Militar y Triaje para Desastres**.** Este Trabajo Especial de Grado se centra en la implementación de un sistema de triaje para el servicio de emergencia.

2.2.1 Triaje del Departamento de Emergencia

Los sistemas de triaje para los departamentos de emergencia están diseñados específicamente para identificar las necesidades más urgentes (o potencialmente más graves) y así garantizar que los pacientes reciban un tratamiento prioritario, según lo indique la escala de triaje.

Según (Rodríguez, 2007) el objetivo del triaje de emergencia consiste en priorizar la atención del paciente según la gravedad mediante una primera valoración rápida, exhaustiva y rigurosa basada en unos protocolos ya establecidos o en el criterio del profesional encargado del servicio. Entre otros objetivos dentro del proceso de triaje se tiene:

- Mejorar la calidad asistencial del servicio, garantizando la equidad en la asistencia, valorando el nivel de gravedad y el tiempo de espera.
- Diferenciar los casos realmente urgentes de aquellos que no lo son a partir de una serie de preguntas y protocolos establecidos, clasificando a los enfermos según criterios de gravedad y no de orden de llegada.
- Disminuir la ansiedad del paciente y la familia al establecer una comunicación inicial y proveer información sobre el proceso de atención y tiempo de espera probable.
- Aplicar, si procede, ciertas técnicas iníciales y básicas propias del ejercicio profesional.
- Determinar el área más adecuada para tratar a un paciente que se presenta en el servicio de emergencias.

2.2.2 Modelo de triaje de la Sociedad Venezolana de Medicina de Emergencia y Desastre

La práctica actual en el área de emergencia de los hospitales venezolanos no considera un proceso de valorización clínica preliminar que ordene los pacientes según el nivel de urgencia o gravedad (Rodríguez, 2007). En tal sentido, la SVMED, tiene como objetivo diseñar un proceso de triaje que se convierta en un instrumento valioso de ayuda a la gestión de la asistencia del servicio de emergencia, colaborando en la eficiencia del servicio y aportando un orden justo en la asistencia, basado en la urgencia/gravedad de los pacientes. El proceso de triaje debe cumplir con los siguientes propósitos:

- Identificar a los pacientes en situación de riesgo, mediante un sistema estandarizado de clasificación de cinco niveles, basados en el MAT (Gómez, 2003).
- Señalar la prioridad en función del nivel de clasificación, acorde con la condición clínica del paciente.

- Determinar el área más adecuada para tratar al paciente que se presenta en el servicio de emergencia.
- Informar sobre las necesidades de exploraciones diagnósticas preliminares.
- Indicar a los pacientes y sus familias sobre el tipo de servicio que necesita el paciente y el tiempo de espera probable.

En la **Figura 1** se ilustra el proceso de triaje de emergencia hospitalario definido por SVMED, el cual se inicia con una evaluación rápida del paciente en el momento de su llegada al servicio de emergencia, mediante la aplicación del método básico denominado ABC de la reanimación, en el cual se obtiene la primera impresión del estado general de salud, tomando como referencia su síntoma guía y desarrollando un despistaje en base a la clasificación de los grupos A, B, C y D para determinar el nivel de prioridad de atención del paciente, el cual va desde la Prioridad 1 = Crítico , hasta la Prioridad 5 = No urgente**.** Se puede apreciar que existen atributos dentro del triaje que no son precisos como pulso débil o pulso muy fuerte, piel fría o muy caliente. La lógica clásica no es capaz de representar estos atributos, es por esto que se recurre a la lógica difusa en donde se podrán tratar estos atributos en su totalidad.

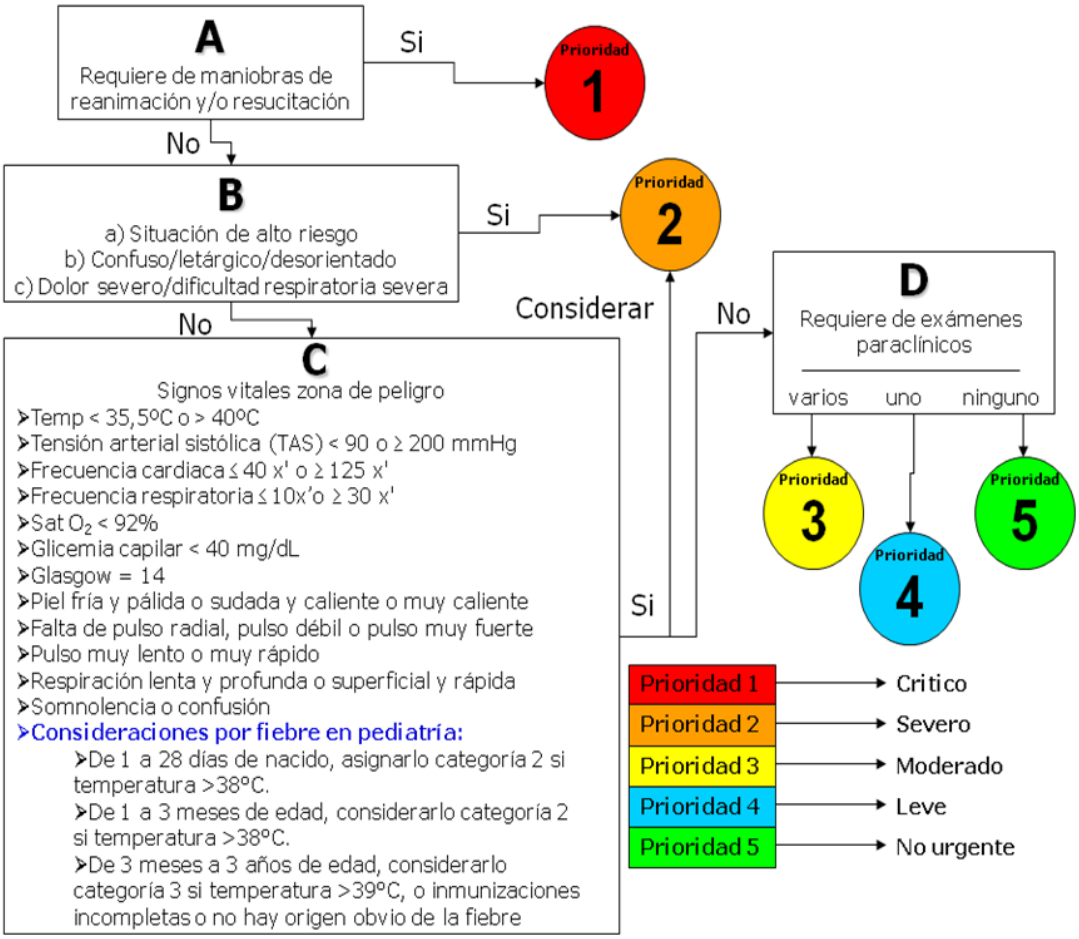

Figura 1. Modelo de Triaje propuesto por la SVMED (Rodríguez, 2007)

Las reglas o sintomatologías que pertenecen a cada grupo son:

Grupo A: dado el nivel de criticidad, estas reglas son de evaluación más rápida o directa, dado que estos pacientes requieren maniobras de reanimación. Para diagnosticar cuales pacientes entran en esta categoría se tienen las siguientes preguntas:

- ¿Tiene una vía aérea permeable?
- ¿Está respirando?
- ¿Tiene pulso?
- ¿Fue intubado en el área prehospitalaria?
- ¿Oxigenación tisular suficiente?
- Si posee alguna de las condiciones siguientes:
- Apnea
- Paro cardiorespiratorio
- Distress respiratorio severo
- SatO2<90
- Paciente no responde a estímulos verbales o dolorosos

Los pacientes dentro del grupo A deben ser atendidos de manera inmediata.

Grupo B: estas reglas requieren un poco más de evaluación y observación. Los antecedentes y el interrogatorio por parte del personal asistencial le dan valor agregado a las evaluaciones dentro de este grupo. Las reglas del grupo B son:

- *Interrogatorio*: en el interrogatorio se pregunta la sintomatología que presenta el paciente. Varios expertos apuntan que se puede encontrar síntomas comunes que se podrán utilizar como pivote, estos son:
    - Fiebre: Se requiere preguntar cuándo inicio; cuál es el grado o temperatura; si tomó algún medicamento; si aún se encuentra presente; cuántas veces le da en un día; duración.
    - Dolor: Se necesita saber la localización; si es alguno de los tipos cólico, urente, punzante, opresivo, sordo; intensidad; duración; cuantas veces le da en el día; cuándo comenzó.
    - Diarrea: Si es liquida o pastosa; cantidad; si posee moco o sangre; qué tan seguido va al baño en un día; cuándo comenzó.
    - Tos: Si es seca o húmeda; si es productiva, en caso positivo preguntar si el moco es verde o claro; cuándo comenzó.

- *Determinación del nivel de conciencia*: Clasificados en 4 niveles, Alerta (A) que significa que el paciente responde al llamado y está ubicado en tiempo, espacio y persona, respuesta verbal (V) en esta el paciente responde al estímulo verbal pero no está ubicado en tiempo, espacio y persona, respuesta al dolor (D) el paciente no responde al estímulo verbal pero responde al dolor, por último sin reacción (R) el paciente no responde a estímulos verbales ni a los dolorosos. Se considera una situación de riesgo cualquier nivel superior a A.
- *Escala de dolor*: La escala del dolor es una herramienta que se usa comúnmente para describir la intensidad del dolor, o qué tanto dolor está sintiendo el paciente, dentro de adultos las escalas de dolor más utilizadas son las numéricas donde se utilizan números del 1 al 10 para identificar desde ausencia de dolor (1) al peor dolor inimaginable (10) y la escala análoga visual la cual trata de una línea que va desde nada de dolor hasta muchísimo dolor y se pide al paciente que coloque el dedo en la posición de la línea tal que identifique su dolor.

Los pacientes dentro del grupo B deben ser atendidos a lo sumo diez minutos después de su llegada al servicio de emergencia, además es necesario realizar un electrocardiograma a personas mayores de 30 años.

Grupo C: Si el paciente no es clasificado dentro del grupo B es necesario hacer un conjunto de evaluaciones médicas más objetivas las cuales están categorizadas dentro del grupo C, donde se miden ciertos signos vitales y se comparan contra un conjunto de límites de tal forma que todo paciente que cumpla con por lo menos dos (02) debe ser atendido con la misma urgencia que el grupo anterior:

- Temperatura: se debe medir la temperatura del paciente, las barreras de temperatura son menor a 35,5 ºC o mayores a 40 ºC.
- Tensión arterial sistólica (TAS): cualquier tensión arterial del paciente que esté por debajo de 90mmHg o por encima de 200mgHg serán considerados de situación crítica.
- Frecuencia cardíaca: los pacientes con frecuencia cardíaca inferior a 40 ó superior a 125 se encuentran en situación de riesgo.
- Frecuencia respiratoria: la frecuencia respiratoria se considera dentro de los valores normales mientras ésta se encuentre dentro de 10 ó 30 respiraciones por minuto.
- SatO2 < 92%
- Glicemia capilar < 40 mg/dL
- Glasgow = 14
- Piel fría y pálida, o, sudada y caliente, o muy caliente
- Falta de pulso radial, pulso débil o pulso muy fuerte

- Pulso muy lento o muy rápido
- Respiración lenta y profunda, o superficial y rápida.

Grupo D: Si el paciente requiere de exámenes paraclinicos, o sencillamente no posee algún signo vital que coloque su vida en riesgo entonces cae dentro de este grupo.

Para modelar y apoyar los sistemas de triaje, desde la ciencia de la computación surgen los sistemas de información de apoyo a la toma de decisiones clínicas o en inglés *Clinical Decision Support System* (CDSS), como se describe en el próximo punto.

2.3 Sistema de Apoyo a la Toma de Decisiones y CDSS

Los sistemas de apoyo a la toma de decisiones, son sistemas casi siempre interactivos, que están diseñados para dar soporte a la toma de decisiones poco estructuradas. Estos sistemas (también conocidos como DSS, del inglés, *Decision Support Systems*) incorporan datos y modelos para ayudar a resolver un problema que no está totalmente estructurado.

En el marco de esta investigación se estudió un tipo de DSS llamado CDSS por sus siglas en ingles *Clinical Decision Support System* o Sistemas para Apoyo a Toma de Decisiones Clínicas. En (Garg, Adhikari, & McDonald, 2005) definen a los CDSS como sistemas de información diseñados para mejorar la calidad de las tomas de decisiones a nivel clínico, en general un CDSS tiene una base de conocimiento de pacientes y una serie de algoritmos que generan recomendaciones especificas para cada paciente. Los CDSS pueden ser clasificados en:

Sistemas para el manejo de información: Son aquellos que proveen datos y conocimiento para el personal de la salud, dando un poco más de especificidad estos mismos se pueden clasificar en:

Sistemas de atención: Estos sistemas funcionan como alarmas que recuerdan al personal médico asistencial tareas que requieren atención.

Sistemas para proveer recomendaciones específicas para pacientes: Se encargan de dar consejos basados en el historial clínico del paciente. Estos sistemas usan técnicas para la toma de decisiones, según sean las reglas con las que fue diseñado el sistema, como por ejemplo los distintos modelos de triaje.

2.4 Lógica Difusa

En (Zadeh, 1965) se introduce la teoría de conjuntos difusos como una generalización de la teórica clásica de conjuntos buscando abarcar los conceptos de vaguedad e imprecisión. Para expresar o

identificar conjuntos difusos, se puede utilizar un **predicado** o **etiqueta lingüística**, de esta forma una función de membresía expresa el grado en que un valor x pertenece al conjunto A (etiqueta).

En (Galindo, 2006) ejemplifican el concepto de etiquetas lingüísticas utilizando el universo del discurso *edad* en el que se expresan conceptos cualitativos como: Joven, Adulto y Viejo. Si colocamos este concepto, en un eje ordenado, rápida e intuitivamente se puede ubicar a joven antes que adulto y adulto antes que viejo, en la **Figura 2** se observa una representación de esta función.

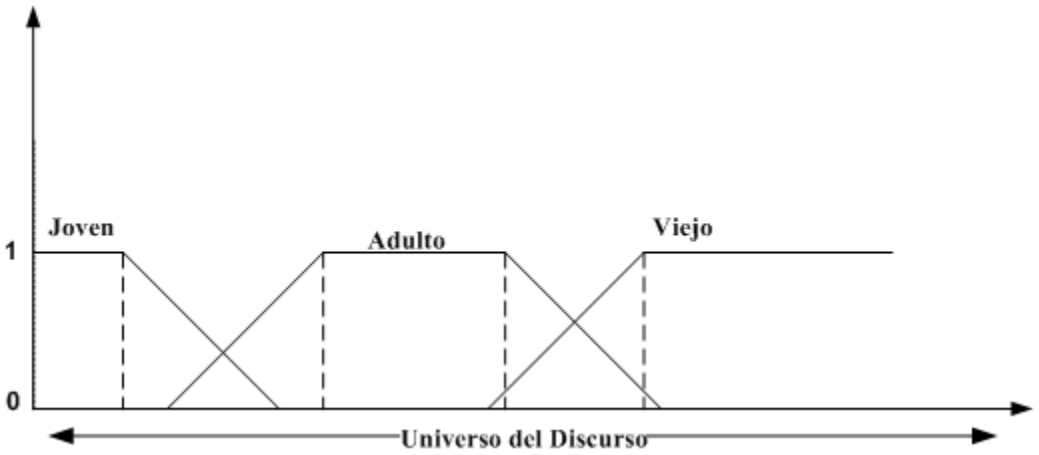

Figura 2. Representación del universo del discurso para edad

En esta figura se puede observar que la función de membresía con la que se definieron las etiquetas forma cada una de ellas un trapecio. A continuación se describen algunas funciones de membresía.

2.4.1 Funciones de Membresía

En 1965 Zadeh propuso un conjunto de funciones de pertenencia que se clasifican en 2 grupos: las formadas por líneas rectas denominadas "lineales", y las contrarias denominadas "curvas" o formas Gausianas. Galindo (2006) afirma que dependiendo de la función de membresía que se utilice se obtendrán diferentes tipos de conjuntos difusos.

A continuación se describirán las funciones de membresía utilizadas en este trabajo.

Función Triangular

La función triangular, como se muestra en la **Figura 3,** está definida bajo 3 parámetros constantes y 1 variable, como e muestra en la **Tabla 1**:

Tabla 1. Variables de la función triangular

Parámetro	Descripción
a	Límite inferior
b	Límite superior
m	Moda
x	Elemento

$$A(x) = \begin{cases} 0 & \text{si } x \le a \\ (x-a)/(m-a) & \text{si } x \in (a, m] \\ (b-x)/(b-m) & \text{si } x \in (m, b) \\ 1 & \text{si } x \ge b \end{cases}$$

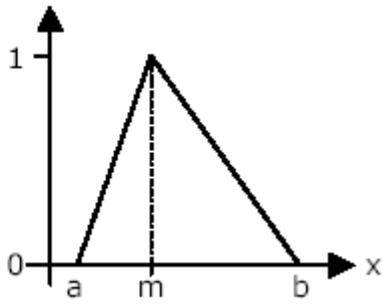

Figura 3. Función triangular

Función Trapezoidal

Definida por sus límites inferior *a* y superior *d*, y los límites de su núcleo, *b* y *c*, inferior y superior respectivamente, como se muestra en la **Figura 4**.

$$T(x) = \begin{cases} 0 & \text{si } (x \le a) \text{ o } (x \ge d) \\ (x-a)/(b-a) & \text{si } x \in (a, b] \\ 1 & \text{si } x \in (b, c) \\ (d-x)/(d-c) & \text{si } x \in (b, d) \end{cases}$$

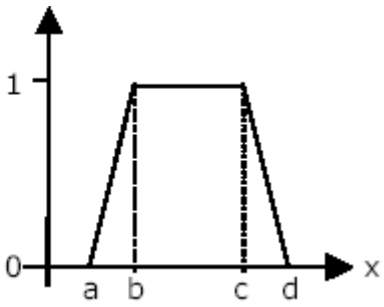

Figura 4. Función Trapezoidal

2.5 Fuzzy SQL (FSQL)

FSQL es una extensión del Lenguaje Estructurado de Consulta SQL (Structures Query Language), siendo éste, el lenguaje aceptado como estándar para los SGBDR.

Una típica consulta de lenguaje SQL permite encontrar algo como "Dame todas las notas de Jose Schmidt en el semestre 2-2010 " o "Cuales son los vendedores que vendieron *más de* 15000 Bs en el 2009", sin embargo este lenguaje no nos permitiría evaluar una consulta como "Dame todas las

personas ***Altas*** que sean ***Jóvenes***" dado que Altas y Jóvenes son dos términos lingüísticos que no están asociados a un valor especifico sino a un rango de valores, lo que lleva a tratar data imprecisa o difusa.

Para lograr manejar los valores difusos dentro de los lenguajes de consulta, Medina (1994) propone la extensión de las estructuras DML y DDL del lenguaje SQL denominando esta extensión FSQL de tal forma que SQL se vuelve un subconjunto del lenguaje FSQL.

2.5.1 Definición de Atributos difusos

FSQL se basa en GEFRED (Medina, 1994; Galindo, 1999), donde existen cuatro tipos de atributos que pueden tener tratamiento impreciso. Los atributos toman valores en un dominio específico, de manera que, el conjunto difuso de posibles valores que puede tomar un atributo se denominará dominio difuso. Los atributos difusos fueron analizados por Urrutia (2003), y clasificados en cuatro tipos: Tipo 1, para datos precisos que admiten tratamiento difuso, Tipo 2, para datos imprecisos en dominio de referencial ordenado, Tipo 3, para datos imprecisos con dominio de referencial no ordenado asociados a una relación de similitud y Tipo 4, que son semejante al Tipo 3, pero no asocian una relación de similitud. Sus correspondientes características se resumen en la **Tabla 2.**

Tabla 2. Características de los atributos difusos

Atributos Difusos	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Tipos de datos que representa	Datos Precisos (crips) Datos Imprecisos	Datos Imprecisos	Datos Imprecisos	Datos Imprecisos
Representación	Escalares simples Etiquetas Lingüísticas Valores aproximados	Distribución de Posibilidad trapezoidal Etiquetas Linguisticas Valores Aproximados Intervalos de Posibilidad UNKNOW UNDEFINED NULL	Escalares Simples Distribución de Posibilidad sobre escalares UNKNOW UNDEFINED NULL	Escalares Simples Distribución de Posibilidad sobre escalares
Dominio Subyacente	Referencial Ordenado	Referencial Ordenado	Referencial no Ordenado	Referencial no Ordenado
Función aplicada	Función de Pertenencia $\mu_{A(u)}$	Función de Pertenencia $\mu_{A(u)}$	Función de Similitud $\mu_{A(u,v)}$	No aplica

2.5.2 Predicados (Etiquetas lingüísticas)

Un aporte interesante de Fuzzy SQL es el uso de predicados, dado que se puede tratar atributos bajo lógica difusa usando etiquetas lingüísticas. Estas etiquetas se colocarán dentro de la sintaxis con el prefijo $. Existen 2 tipos de etiquetas lingüísticas:

- Etiquetas para atributos ordenados: están asociadas a atributos con valores que se encuentran naturalmente ordenados como edades, tamaños, etc. Estas etiquetas son utilizadas en atributos fuzzy tipo 1 y 2.

Etiquetas para atributos no ordenados: se utilizan cuando la relación entre las valores de los atributos no se encuentra naturalmente ordenada, sino que en vez se utiliza una relación de similitud, un caso clásico son los colores. Los tipos de atributo fuzzy en los que aplican son 3 y 4.

Las etiquetas lingüísticas son utilizadas dentro de las clausulas *where* o *having*.

2.5.3 Comparadores Difusos

Además de los comparadores lógicos que son absorbidos de SQL como son <,>,=, etc, FSQL añade una nueva serie de comparadores que se denominan operadores de posibilidad y necesidad. La idea fundamental de estos comparadores es vencer la restrictividad asociada a los comparadores clásicos de tal forma que podamos obtener más resultados que pueden ser de interés en una consulta.

En esencia existen dos familias de operadores difusos: los operadores de posibilidad y los operadores de necesidad. Los comparadores de posibilidad son más generales que los de necesidad lo que traduce que uno retoma más registros que el otro. El comparador de posibilidad será denotado utilizando FEQ (Fuzzy Equal) y el de necesidad será denotado como NFEQ (Neccesity Fuzzy Equal).

2.5.4 Grado de cumplimiento y Calificadores

Para cualquier condición simple se puede especificar un grado de cumplimiento (por defecto es 1) de la siguiente forma:

<condición> THOLD t

lo cual significa que la condición debe ser satisfecha por un mínimo grado de cumplimiento t, donde t $\in$ [0,1], y a su vez t puede tomar el valor de un calificador, por ejemplo si definimos $alto

en el sistema como 0.8 podemos hacer una consulta como <condición> THOLD $alto. FSQL también permite el uso de THOLD en comparaciones compuestas (con varias condiciones).

2.5.5 Función CDEG() y operadores lógicos

Esta función permite calcular el grado de compatibilidad o cumplimiento de un argumento, para calcular el grado de compatibilidad de una condición compuesta es posible llevar la función CDEG() (*Compatibility Degree*) a su elemento más interno, es decir si se tiene la condición <condición1> AND <condición2> y aplicamos el CDEG la función queda de la siguiente manera

min(CDEG(<condición1>),CDEG(<condición2>))

Para mostrar cual es el valor de grado de cumplimiento de los registros dentro de la consulta FSQL se puede utilizar CDEG(*) dentro del select, o bien si se quiere todas las columnas en forma semejante al * de SQL y además el CDEG entonces podemos utilizar el símbolo %

2.6 Fuzzy SQL Server

En (Medina, 1994) y (Galindo, 1999) se expuso un módulo para permitir extender la capacidad de un Sistema Manejador de Base de Datos Relacional (SMBDR) clásico que pueda representar y manipular información “imprecisa”. Este módulo, llamado FIRST (Fuzzy Interface for Relational SysTems, Interface Difuso para Sistemas Relacionales), utiliza GEFRED (Medina, 1994) como modelo teórico y los recursos del modelo relacional clásico para poder representar este tipo de información.

El objetivo de esta arquitectura es poder almacenar y tratar información imprecisa, difusa y para ello se establece un método formal de hacerlo. Cuenta con un conjunto de elementos básicos con los que está construida y cómo se relacionan unos con otros. Esos elementos son de distinta naturaleza: estructuras de datos, tablas, vistas, gramáticas para la definición de DML (Data Manipulation Language) y DDL (Data Definition Language), programas en distintos lenguajes, datos, entre otros. Los componentes principales de esta arquitectura son:

SMBDR: Todas las operaciones que se conciben para la extensión difusa que representa esta implementación, se traducen a peticiones al SMBDR anfitrión (en general en SQL). Las peticiones al SMBDR se realizan empleando el lenguaje SQL o FSQL, dependiendo si la consulta involucra o no relaciones y/o condiciones difusas. Las sentencias FSQL serán procesadas por el Servidor FSQL.

BD (Base de Datos): Almacena, en formato relacional toda la información que sea de interés, igual que cualquier otra base de datos. La única diferencia es que esta base de datos permitirá el

almacenamiento de información difusa en sus tablas. La forma en que se representan los datos en dichas tablas dependerá de la naturaleza de los mismos.

FMB (Fuzzy Metaknowledge Base, Base de Metaconocimiento Difuso): El diccionario o catálogo del sistema de un SMBDR representa aquella parte del sistema que almacena información sobre los datos recogidos en la base de datos, así como otro tipo de informaciones: usuarios, permisos, accesos, datos de control. Dentro de este catálogo se ha incluido la FMB, que extiende esta parte del sistema a fin de que pueda recoger aquella información necesaria relacionada con la naturaleza imprecisa de la nueva colección de datos a procesar.

Servidor FSQL: Su objetivo es captar las sentencias en lenguaje FSQL y traducirlas a lenguaje SQL que entiende el SMBDR. Para efectuar esta traducción utiliza la información almacenada en la FMB. Este Servidor está íntegramente programado en el lenguaje PL/SQL de Oracle y se encuentra implantado como una colección de módulos almacenados en el Servidor Oracle.

Cliente FSQL: Se trata de un programa modo cliente/servidor, que hace de interfaz entre el usuario (u otro programa) y el Servidor FSQL. Este programa puede ser muy simple, pues el trabajo principal de una operación FSQL será efectuado por el Servidor FSQL. La **Figura 5** muestra el esquema general de Bases de Datos Relacional Difusa (BDRD).

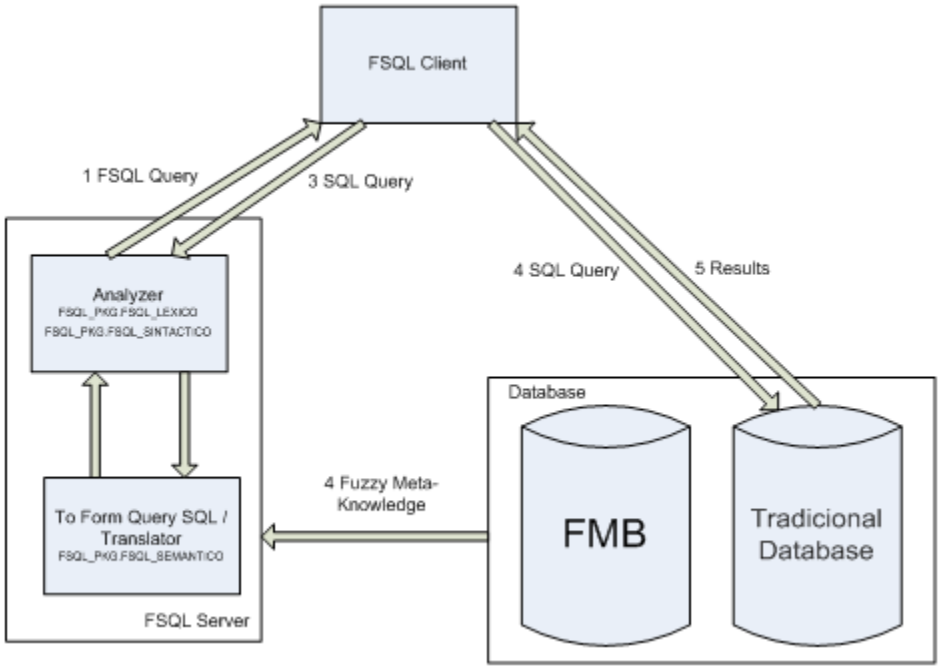

Figura 5. Arquitectura de la implementación FSQL (Galindo, 1999)

Cabe destacar que para cada tipo de atributo difuso, es necesario especificar dos aspectos:

¿Cómo se representan los valores (que puede almacenar el atributo)?.
¿Qué información debe ser almacenada en la (FMB) para procesarla, y cómo esta información debe ser organizada?

La FMB será la responsable de organizar toda la información relacionada con la naturaleza imprecisa o el contexto de estos atributos. La FMB se contempla como una ampliación del catálogo del sistema (Data Dictionary), la cual organiza la información mediante el uso de tablas o relaciones.

2.6.1 Representación de atributos difusos

Cada tipo de datos, descrito en los apartados anteriores, tiene su respectiva representación en el SMBD (Galindo, 1999):

- **Atributos Difusos Tipo 1**: Son atributos con datos precisos, clásicos o crisp, que pueden tener etiquetas lingüísticas definidas sobre ellos. Este tipo de atributos se representa igual que los datos precisos; sin embargo pueden tener información en la FMB, donde se almacenan las etiquetas lingüísticas; por lo tanto son atributos clásicos que admiten el tratamiento difuso, sobre los cuales se pueden efectuar consultas (flexibles), aunque no permitan almacenar valores difusos.

- **Atributos Difusos Tipo 2**: Son atributos que pueden recoger “datos imprecisos sobre referencial ordenado”. Estos atributos recogen el tipo de datos cuya representación ha sido descrita en el apartado anterior de representación de datos sobre referencial ordenado, así como también permiten la representación de información de tipo UNKNOWN, UNDEFINED, NULL, como se muestra en la **Tabla 3**.

Tabla 3 Representación interna para Atributos Difusos Tipo 2.

Tipos de Valores	**4 Atributos de la BD para cada tipo 2**				
	FT	**F1**	**F2**	**F3**	**F4**
Unknown	0	Null	Null	Null	Null
Undefined	1	Null	Null	Null	Null
Null	2	Null	Null	Null	Null
Crisp	3	D	Null	Null	Null
Etiqueta	4	Fuzzy_ID	Null	Null	Null
Intervalo [n, m]	5	N	Null	Null	M
Aproximadamente(d)	6	D	d - margen	d + margen	Margen
Trapecio [α, β, γ, δ]	7	A	β - α	γ – δ	Δ

- **Atributos Difusos Tipo 3**: Son atributos sobre "dominio discreto no ordenado con analogía". Estos atributos recogen datos escalares simples (SIMPLE) o distribuciones de posibilidad (DISTRIBUCIÓN POSIBILIDAD) sobre dominios escalares, representados en la forma descrita en el apartado anterior. También aceptan datos de tipo UNKNOWN, UNDEFINED y NULL.

Igual que en atributos difusos Tipo 2, para atributos de este tipo se tiene que almacenar en la base de datos el tipo del valor almacenado y los datos de este valor. La FMB contabilizará cada atributo de este tipo que aparezca en la base de datos. También almacenará las relaciones de semejanza definidas sobre el dominio subyacente.

En la Tabla 4 se muestra la representación para atributos difusos Tipo 3. Así, un atributo difuso Tipo 3, llamado por ejemplo F, está compuesto, de hecho, por un número variable de atributos clásicos:

Tabla 4. Representación interna para Atributos Difusos Tipo 3.

Tipo de Valores	**5 Atributos de la BD para cada Tipo 3**					
	FT	**FP1**	**F1**	...	**FPn**	**Fn**
Unkown	0	Null	Null	...	Null	Null
Undefined	1	Null	Null	...	Null	Null
Null	2	Null	Null	...	Null	Null
Simple	3	p	d	...	Null	Null
Dist. Pos	4	p_1	d_1	...	p_n	d_n

En la **Tabla 4**, se observa el valor n como el máximo número de pares (grado posibilidad, valor) que puede representar el atributo instanciado. Este valor, que tiene que ser establecido en el momento en que se declara el atributo, acota su capacidad para representar distribuciones de posibilidad y está almacenado en la FMB. En un valor de tipo DISTRIBUCIÓN de POSIBILIDAD, se podrán almacenar hasta n parejas, donde en cada una de ellas el valor de posibilidad estará en el intervalo [0,1]. Se pueden usar menos de n parejas dejando el resto de campos a NULL. En la FMB se almacenan las etiquetas, su relación de semejanza y el valor de n.

2.6.2 FMB (Fuzzy Metaknowledge Base, Base de Metaconocimiento Difuso): Definición de Tablas

Como se ha visto en el apartado anterior, existe cierto tipo de información sobre los atributos difusos descritos, que precisa ser almacenada de una forma accesible por el sistema. Esta información se almacena en formato relacional en la llamada FMB (Fuzzy Metaknowledge Base).

En FIRST se contempla FMB como una extensión del catálogo del sistema, por ello, organiza la información mediante el uso de tablas o relaciones. Los elementos del tratamiento impreciso que se almacena en la FMB son los siguientes:

- Atributos de la base de datos que reciben tratamiento impreciso.
- Clase de información imprecisa que recogen
- Tipo de atributo difuso (Tipo 1, 2 ó 3) y longitud máxima de las distribuciones de posibilidad para los atributos Tipo 3.
- Objetos definidos en el ámbito de la base de datos, como por ejemplo cuantificadores difusos de consultas.
- Objetos difusos sobre cada atributo:
    - Etiquetas lingüísticas (para atributos difusos Tipo 1, 2 y 3).
    - El margen para valores apropiados y la distancia mínima para que dos valores sean considerados como “muy” separados (para atributos Tipo 1 y 2).
    - Relaciones de semejanza (para atributos difusos Tipo 3).
    - La descripción.

A continuación se detalla como FIRST implementa la FMB, que es el núcleo básico de FIRST. En la **Tabla 5** se expone el conjunto de las tablas y vistas de la FMB y el nombre del sinónimo público creado para cada una.

Tabla 5. Tablas del FMB de FIRST-2

Objeto Tabla/Vista	**Sinónimo**
T. FUZZY_COL_LIST	FCL
T. FUZZY_OBJECT_LIST	FOL
T. FUZZY_LABEL_DEF	FLD
T. FUZZY_APPROX_MUCH	FAM
T. FUZZY_NEARNESS_DEF	FND
T. FUZZY_COMPATIBLE_COL	FCC
T. FUZZY_QUALIFIERS_DEF	FQD
V. FUZZY_FOR_OBJCOL	LFOC
V. FUZZY_OBJCOL_T3	LOCT3
V. ALL_COMPATIBLES_T3	ACT3

2.7 Modelo difuso de triaje SVMED

La **Figura 6** muestra el modelo EER para la representación del Proceso de Triaje de la SVMED propuesto por Rangel (2010). En este modelo se distinguen entidades tales como: TRIAJE y PACIENTE que son requeridos en el servicio de emergencia hospitalaria, DOCTOR que realiza la actividad de triaje, MOTIVOURGENCIA que conlleva al paciente al servicio de emergencia,

SIGNOVITAL y CONSTANTE que observa el doctor en el paciente, SINTOMA que presenta el paciente al momento de la llegada al servicio, PROTOCOLO sintomático que agrupa los principales

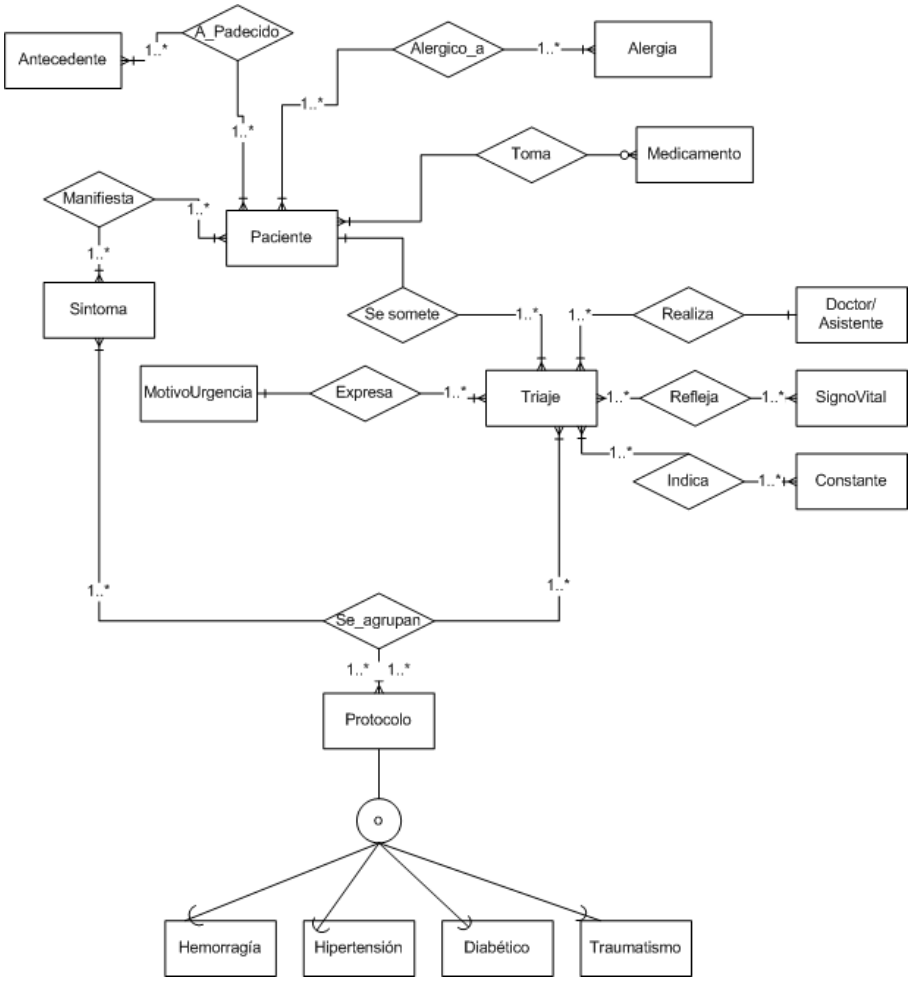

Figura 6. Modelo EER para triaje basado en el proceso de la SVMED (Rangel, 2010)

motivos clínicos de consulta. Además, se pueden encontrar algunas subclases (o subtipos) de PROTOCOLOS sintomáticos, tales como: HEMORRAGIA, HIPERTENSIÓN, DIABÉTICO, TRAUMATISMO, etc., que en algún momento pueden ser entidades disjuntas o solapadas. Por ejemplo, un paciente puede presentar síntomas de Hemorragia y Traumatismo en una entrevista de triaje.

Las entidades que no presentan atributos con dominios son: PACIENTE, DOCTOR, TRIAJE, ANTECEDENTE, MEDICAMENTO y ALERGIA. La descripción de estas entidades y sus atributos son las siguientes:

DOCTOR: En esta entidad se almacena la información del médico que se encuentra realizando el proceso de triaje. Los atributos son {código **integer**, nombre **string**, apellido **string**, especialidad **string**}.

ANTECEDENTE: Esta entidad se encarga de almacenar los posibles patologías que sufren o han sufrido los pacientes antes de su llegada a la emergencia como por ejemplo diabetes, tensión alta, asma, etc. Los atributos de esta entidad son {código **integer**, descripción **string**}.

MEDICAMENTO: Almacena de ser necesario, los medicamentos a los que el paciente se está sometiendo actualmente, además estos pueden ser clasificados como antihipertensivos, analgésicos, etc. Los atributos de esta entidad son {código **integer**, categoría **string**, descripción **string**}.

PACIENTE: Esta tabla almacena los datos personales del paciente. Sus atributos son {código **integer**, nombre **string**, apellido **string**, cedula **integer**, genero **char**}.

TRIAJE: Tabla que almacena un episodio de triaje, esta tabla se encuentra relacionada con todas las demás tablas. Los atributos de esta entidad son {código **integer**, doctor_código **integer**, motivo_urgencia_código **integer**, paciente_código **integer**, fecha_hora_minuto **datetime**}.

ALERGIA: Esta tabla almacena las distintas alergias que pueden padecer los pacientes. Los atributos son {código **integer**, descripción **string**, tipo **string**}.

A continuación se discuten algunos atributos y sus correspondientes dominios, tanto precisos como imprecisos, tomando como referencia las entidades SINTOMA, CONSTANTE, MOTIVOURGENCIA, SIGNOSVITALES y CATEGORIA. Se ha considerado la representación de las entidades por separado, para un mejor entendimiento de cada atributo y su dominio.

2.8 Modelo Conceptual EER Difuso de Triaje de Emergencia Hospitalaria

En reuniones sostenidas con los *stakeholders* se analizaron las distintas entidades difusas que propone el modelo. De estas reuniones y con el análisis del modelo de triaje de la SVMED se obtuvo un primer conjunto de valores que fueron utilizados dentro de las etiquetas difusas, sin embargo quedo claramente establecido que estos valores deben ser sometidos a fuertes pruebas posteriores que permitan mejorar la calidad de los mismos.

SÍNTOMA. En esta entidad se establecen los atributos {Código **integer**, Inicio **T2**, Descripción **string**, Duración **T2**, Nivel **T3**, Localización **T2**, Repetición **T2**, Pregunta1 **boolean**, Pregunta2

boolean}. Los atributos clásicos que se definen son: Código, Descripción, Tipo, Pregunta1 y Pregunta2. Los atributos difusos y sus características se especifican a continuación:

T2 :Duración: Las etiquetas definidas para este atributo miden cuánto dura en minutos el síntoma, como se muestra en la **Tabla 6**.

Tabla 6. Representación del atributo difuso T2: Duración

Nombre	**A**	**B**	**Γ**	**Δ**
Poco	0	1	5	10
mas_o_menos	8	15	25	28
Mucho	25	30	10000	10001

T2 :Localización: En caso de ser posible, corresponde a la parte del cuerpo donde se manifiesta el síntoma. Las partes en las que se divide el cuerpo son: Cabeza, Cuello, Torax Izquierdo, Torax Derecho, Abdomen, Genitales, Miembro Superior Izquierdo, Miembro Superior Derecho, Miembro Inferior Izquierdo, Miembro Inferior Derecho. La razón por la que propone Tipo 2, es por la necesidad de poder definir este atributo como UNDEFINED, en cualquier otro caso este atributo tomará valores crisp.

T2: Repetición: Las etiquetas definidas para este atributo se observan en la **Tabla 7**.

Tabla 7. Representación del atributo difuso T2: Repetición

Nombre	**A**	**B**	**Γ**	**Δ**
Poco	0	1	4	6
mas_o_menos	3	5	10	15
Mucho	9	12	10000	10001

T2: Inicio: Las etiquetas definidas para este atributo están expresadas en horas. En la **Tabla 8** se muestra la representación de estas etiquetas.

Tabla 8: Representación del atributo difuso T2: Inicio

Nombre	**A**	**B**	**Γ**	**Δ**
hace_poco	0	8	18	24
mas_o_menos	20	48	72	96
hace_mucho	84	96	10000	10001

T3: Nivel: La **Tabla 9** muestra los grados de semejanza para este atributo difuso.

Tabla 9. Representación del atributo difuso T3: Nivel

	Ligero	**Moderado**	**Intenso**
Ligero	1	0.5	0
Moderado	0.5	1	0.5
Intenso	0	0.5	1

MOTIVOURGENCIA. Esta entidad refleja el motivo por el cual el paciente llega hasta el servicio de emergencia, los atributos de esta entidad son {Código **integer**, Descripción **string**, Duración **T2**, Inicio **T2**}. Los atributos clásicos de esta entidad son Código y Descripción. Los atributos difusos son Duración e Inicio y sus características se especifican a continuación:

T2: Duración: En la **Tabla 10** se muestras la representación de las etiquetas definidas para este atributo.

Tabla 10. Representación del atributo difuso T2: Duración

Nombre	**A**	**B**	**Γ**	**Δ**
Poco	0	1	5	10
mas_o_menos	8	15	25	28
Mucho	25	30	10000	10001

T2 :Inicio: En la **Tabla 11** se muestra la representación de las etiquetas definidas para este atributo

Tabla 11: Representación del atributo difuso T2: Inicio

Nombre	**A**	**B**	**Γ**	**Δ**
hace_poco	0	8	18	24
mas_o_menos	20	48	72	96
hace_mucho	84	96	10000	10001

CONSTANTE. En esta entidad se establecen los atributos {Código **integer**, Temp **T2**, TAS **T2**, FC **T2**, FR **T2**, SatO2 **integer**, GlicemiaCapilar **T2**, NIHSS **T2**}. Código es un atributo clásico, el resto de estos atributos son considerados como difusos y sus características se especifican a continuación:

T2: Temp: En la **Tabla 12** se representan las etiquetas definidas para este atributo.

Tabla 12. Representación del atributo difuso T2: Temp

Nombre	**A**	**β**	**Γ**	**Δ**

Baja	0	1	36	37
Normal	37	38	39	40
Fiebre	39	40	41	42
fiebre_muy_alta	41	42	10000	10001

T2: TAS: En la **Tabla 13** se representan las etiquetas definidas para este atributo.

Tabla 13. Representación del atributo difuso T2: TAS

Nombre	**A**	**B**	**γ**	**Δ**
muy_baja	0	1	90	95
normal	90	95	180	200
muy_alta	190	200	10000	10001

T2: FC: En la **Tabla 14** se representan las etiquetas definidas para este atributo.

Tabla 14. Representación del atributo difuso T2: FC

Nombre	**A**	**B**	**γ**	**Δ**
muy_baja	0	1	40	45
normal	40	45	120	126
muy_alta	120	125	10000	10001

T2: FR: En la **Tabla 15** se representan las etiquetas definidas para este atributo.

Tabla 15. Representación del atributo difuso T2: FR

Nombre	**A**	**β**	**γ**	**Δ**
muy_baja	0	1	10	13
normal	10	11	29	31
muy_alta	29	30	10000	10001

T1: SatO2: Las etiquetas definidas para este atributo son:

Tabla 16. Representación del atributo difuso T1: SatO2

Nombre	**A**	**β**	**γ**	**Δ**
muy_baja	0	1	92	93
normal	92	93	10000	10001

T2: GlicemiaCapilar: En la **Tabla 17** se representan las etiquetas definidas para este atributo.

Tabla 17. Representación del atributo difuso T2: GlicemiaCapilar

Nombre	A	β	Γ	Δ
Hipoglucemia	0	1	69	73
Normal	68	70	99	105
glucosa_alterada_en_ayuno	98	100	124	126
Hiperglucemia	124	126	10000	10001

T2: NIHSS: La **Tabla 18** muestra las etiquetas definidas para este atributo, las cuales tienen como única intensión dar una connotación expresada bajo términos lingüísticos.

Tabla 18. Representación del atributo difuso T2: NIHSS

Nombre	A	β	Γ	Δ
Alerta	0	1	2	3
somnolencia	3	4	5	6
Estupor	6	7	8	9
coma_profundo	9	10	11	12

SIGNOVITAL. En esta entidad se establecen los atributos {código **integer**, tipo_piel **T3**, pulso_radial **T2**, pulso **T2**, respiracion **T2**, glasgow **integer**}. Los atributos clásicos son código y glasgow. Los atributos difusos son:

T3 :tipo_piel: La **Tabla 19** muestra la representación de las etiquetas definidas para este atributo.

Tabla 19. Representación del atributo difuso T3: tipo_piel

	Fría y pálida	Sudada y Caliente	Muy Caliente
Fría y pálida	1	0.4	0
Sudada y Caliente	0.4	1	0.7
Muy Caliente	0	0.7	1

T2 :pulso_radial: En la **Tabla 20** se muestran las etiquetas definidas para este atributo.

Tabla 20. Representación del atributo difuso T2: pulso_radial

Nombre	α	β	γ	Δ
falta	0	1	2	3
débil	3	4	5	6
normal	6	7	8	9
fuerte	9	10	11	12

T2 :pulso: La **Tabla 21** muestra las etiquetas definidas para este atributo.

Tabla 21. Representación del atributo difuso T2: pulso

Nombre	**A**	**β**	**γ**	**Δ**
bradicardia	0	1	50	51
Normal	50	51	119	122
taquicardia	119	120	10000	10001

T2 :respiración: En la **Tabla 22** se representan las etiquetas definidas para este atributo.

Tabla 22. Representación del atributo difuso T2: respiración

Nombre	**A**	**β**	**γ**	**Δ**
bradipnea	0	1	12	13
Normal	12	13	18	20
taquipnea	19	20	10000	10001

CATEGORIA. La **Tabla 23**, muestra algunas subclases que pueden ser similares, identificadas por el atributo difuso T3:Tipo_categoría.

Tabla 23. Representación del atributo difuso T3: Tipo_categoría

	Hemorragia	**Hipertensión**	**Diabetes**	**Traumatismo**
Hemorragia	1	0.3	0.5	0.8
Hipertensión	0.3	1	0.8	0.1
Diabetes	0.5	0.8	1	0.3
Traumatismo	0.8	0.1	0.3	1

2.9 Modelo Relacional Difuso de Triaje de Emergencia Hospitalaria

En este apartado se describe el modelo lógico de la base de datos relacional difusa de triaje.

Tabla Antecedentes

La tabla “Antecedentes” representa a la entidad que lleva por nombre *Antecedente.* Los campos de esta tabla son {codigo, descripcion} donde *codigo* representa la clave primaria de la tabla. La tabla antecedentes queda de la siguiente forma:

ANTECENTES (CODIGO, DESCRIPCION)

Tabla Doctores

La tabla “Doctores” representa a la entidad que lleva por nombre *Doctor.* Los campos de esta tabla son {codigo, nombre, apellido, especialidad} donde *codigo* representa la clave primaria de la tabla. La tabla doctores queda de la siguiente forma:

DOCTORES (CODIGO, NOMBRE, APELLIDO, ESPECIALIDAD)

Tabla Pacientes

La tabla “Pacientes” representa a la entidad que lleva por nombre *Paciente.* Los campos de esta tabla son {codigo, nombre, apellido, cedula, genero} donde *codigo* representa la clave primaria de la tabla. La tabla pacientes queda de la siguiente forma:

PACIENTES (CODIGO, NOMBRE, APELLIDO, CEDULA, GENERO)

Tabla Medicamentos

La tabla “Medicamentos” representa a la entidad que lleva por nombre *Medicamento.* Los campos de esta tabla son {codigo, categoria , descripcion} donde *codigo* representa la clave primaria de la tabla. La tabla medicamentos queda de la siguiente forma:

MEDICAMENTOS (CODIGO, CATEGORIA, DESCRIPCION)

Tabla Alergias

La tabla “Alergias” representa a la entidad que lleva por nombre *Alergia.* Los campos de esta tabla son {codigo, descripcion, tipo} donde *codigo* representa la clave primaria de la tabla. La tabla alergias queda de la siguiente forma:

ALERGIAS (CODIGO, DESCRIPCION, TIPO)

Tabla Motivos_urgencia

La tabla “Motivos_urgencia” representa a la entidad que lleva por nombre *Motivourgencia.* Los campos de esta tabla son {codigo, descripcion, duracion, inicio} donde *codigo* representa la clave primaria de la tabla. La tabla alergias queda de la siguiente forma:

MOTIVOS_URGENCIA (CODIGO, DESCRIPCION, DURACION T2, INICIO T2)

Dado que *duracion* e *inicio* son dos atributos difusos, la representación real de la tabla queda

MOTIVOS_URGENCIA (CODIGO, DESCRIPCION, DURACIONT, DURACION1, DURACION2, DURACION3, DURACION4, INICIOT, INICIO1, INICIO2, INICIO3, INICIO4)

Tabla Episodios_triaje

La tabla “Episodios_triaje” representa a la entidad que lleva por nombre *Triaje.* Los campos de esta tabla son {codigo, doctor_codigo, motivo_urgencia_codigo, paciente_codigo, fecha_hora_minuto} donde *codigo* representa la clave primaria de la tabla, doctor_codigo es una clave foránea a doctores, motivo_urgencia_codigo es una clave foranea a motivos_urgencia y paciente_codigo es una clave foránea a pacientes. La tabla episodios_triaje queda de la siguiente forma:

EPISODIOS_TRIAJE (CODIGO, DOCTOR_CODIGO, MOTIVO_URGENCIA_CODIGO, PACIENTE_CODIGO, FECHA_HORA_MINUTO)

Tabla Refleja

La tabla “Refleja” representa a la relación que lleva por nombre *Refleja.* Los atributos de esta relación son {signo_codigo, episodio_triaje_codigo} donde signo_codigo es una clave foranea a la tabla signos y episodio_triaje_codigo es una clave foranea a la tabla episodios_triaje. La tabla refleja queda de la siguiente forma:

REFLEJA (SIGNO_CODIGO, EPISODIO_TRIAJE_CODIGO)

Tabla Indica

La tabla “Indica” representa a la relación que lleva por nombre *Indica.* Los atributos de esta relación son {constante_codigo, episodio_triaje_codigo} donde constante_codigo es una clave foranea a la tabla constantes y episodio_triaje_codigo es una clave foranea a la tabla episodios_triaje. La tabla indica queda de la siguiente forma:

INDICA (CONSTANTE_CODIGO, EPISODIO_TRIAJE_CODIGO)

Tabla Se_agrupan

La tabla “Se_agrupan” representa a la relación que lleva por nombre *Indica.* Los atributos de esta relación son {sintoma_codigo, episodio_triaje_codigo} donde sintoma_codigo es una clave foranea a la tabla sintomas y episodio_triaje_codigo es una clave foranea a la tabla episodios_triaje. La tabla se_agrupan queda de la siguiente forma:

SE_AGRUPAN (SINTOMA_CODIGO, EPISODIO_TRIAJE_CODIGO)

Tabla Alergico_a

La tabla “Alergico_a” representa a la relación que lleva por nombre *Alergico_a.* Los atributos de esta relación son {paciente_codigo, alergia_codigo} donde paciente_codigo es una clave foranea a la tabla pacientes y alergia_codigo es una clave foranea a la tabla alergias. La tabla alergico_a queda de la siguiente forma:

ALERGICO_A (PACIENTE_CODIGO, ALERGIA_CODIGO)

Tabla A_padecido

La tabla “A_padecido” representa a la relación que lleva por nombre A_*padecido.* Los atributos de esta relación son {paciente_codigo, antecedente_codigo} donde paciente_codigo es una clave foranea a la tabla pacientes y antecedente_codigo es una clave foranea a la tabla antecedentes. La tabla alergico_a queda de la siguiente forma:

A_PADECIDO (PACIENTE_CODIGO, ANTECEDENTE_CODIGO)

Tabla Toma

La tabla “Toma” representa a la relación que lleva por nombre *Toma.* Los atributos de esta relación son {paciente_codigo, medicamento_codigo} donde paciente_codigo es una clave foranea a la tabla pacientes y medicamento_codigo es una clave foranea a la tabla medicamentos. La tabla toma queda de la siguiente forma:

TOMA (PACIENTE_CODIGO, MEDICAMENTO_CODIGO)

Tabla Sintomas

La tabla “Sintomas” representa a la entidad que lleva por nombre *Sintoma.* Los atributos de esta tabla son {codigo, inicio, descripcion, duracion, nivel, localizacion, repeticion, pregunta1, pregunta2} donde codigo representa la clave primaria de la tabla. La tabla sintomas queda de la siguiente forma:

SINTOMAS (CODIGO, INICIO T2, DESCRIPCION, DURACION T2, NIVEL T3, LOCALIZACION T2, REPETICION T2, PREGUNTA1, PREGUNTA2)

Dado que *duracion*, *inicio*, *nivel*, *localizacion* y *repetecion* son atributos difusos, la representación real de la tabla queda

SINTOMAS (CODIGO, INICIOT, INICIO1, INICIO2, INICIO3, INICIO4, DESCRIPCIONT, DURACION1, DURACION2, DURACION3, DURACION4, NIVELT, NIVELP1, NIVEL1,

LOCALIZACIONT, LOCALIZACION1, LOCALIZACION2, LOCALIZACION3, LOCALIZACION4, REPETICIONT, REPETICION1, REPETICION2, REPETICION3, REPETICION4 , PREGUNTA1, PREGUNTA2)

Tabla Constantes

La tabla "Constantes" representa a la entidad que lleva por nombre *Constante.* Los atributos de esta tabla son {codigo, temp, tas, fc, fr, sato2, glicap, nihss} donde codigo representa la clave primaria de la tabla. La tabla constantes queda de la siguiente forma:

CONSTANTES (CODIGO, TEMP T2, FC T2, FR T2, SATO2, GLICAP T2, NIHSS T2)

Dado que *temp*, *fc*, *fr*, *glicap* y *nihss* son atributos difusos, la representación real de la tabla queda

CONSTANTES (CODIGO, TEMPT, TEMP1, TEMP2, TEMP3, TEMP4, FCT, FC1, FC2, FC3, FC4, FRT, FR1, FR2, FR3, FR4, SATO2, GLICAPT, GLICAP1, GLICAP2, GLICAP3, GLICAP4, NIHSST, NIHSS1, NIHSS2, NIHSS3, NIHSS4)

Tabla Signos

La tabla "Signos" representa a la entidad que lleva por nombre *Signovital.* Los atributos de esta tabla son {codigo, tipo_piel, pulso_radial, pulso, respiracion, glasgow} donde codigo representa la clave primaria de la tabla. La tabla signos queda de la siguiente forma:

SIGNOS (CODIGO, TIPO_PIEL T3, PULSO_RADIAL T2, PULSO T2, RESPIRACION T2, GLASGOW)

Dado que *tipo_piel*, *pulso_radial*, *pulso* y *respiracion* son atributos difusos, la representación real de la tabla queda

SIGNOS (CODIGO, TIPO_PIELT, TIPO_PIELP1, TIPO_PIEL1, PULSO_RADIALT, PULSO_RADIAL1, PULSO_RADIAL2, PULSO_RADIAL3, PULSO_RADIAL4, PULSOT, PULSO1, PULSO2, PULSO3, PULSO4, RESPIRACIONT, RESPIRACION1, RESPIRACION2, RESPIRACION3, RESPIRACION4, GLASGOW)

Tabla Categorias

La tabla "Categorias" representa a la entidad que lleva por nombre *Protocolo.* Los atributos de esta tabla son {codigo, tipo, tipo_categoria T3} donde codigo representa la clave primaria de la tabla. La tabla signos queda de la siguiente forma:

CATEGORIAS (CODIGO, DESCRIPCION, TIPO_CATEGORIA T3)

Dado que *tipo_categoria es un* atributo difuso, la representación real de la tabla queda

CATEGORIAS (CODIGO, DESCRIPCION, TIPO_CATEGORIAT, TIPO_CATEGORIAP1, TIPO_CATEGORIA1)

La **Figura 7** representa el modelo relacional difuso de triaje.

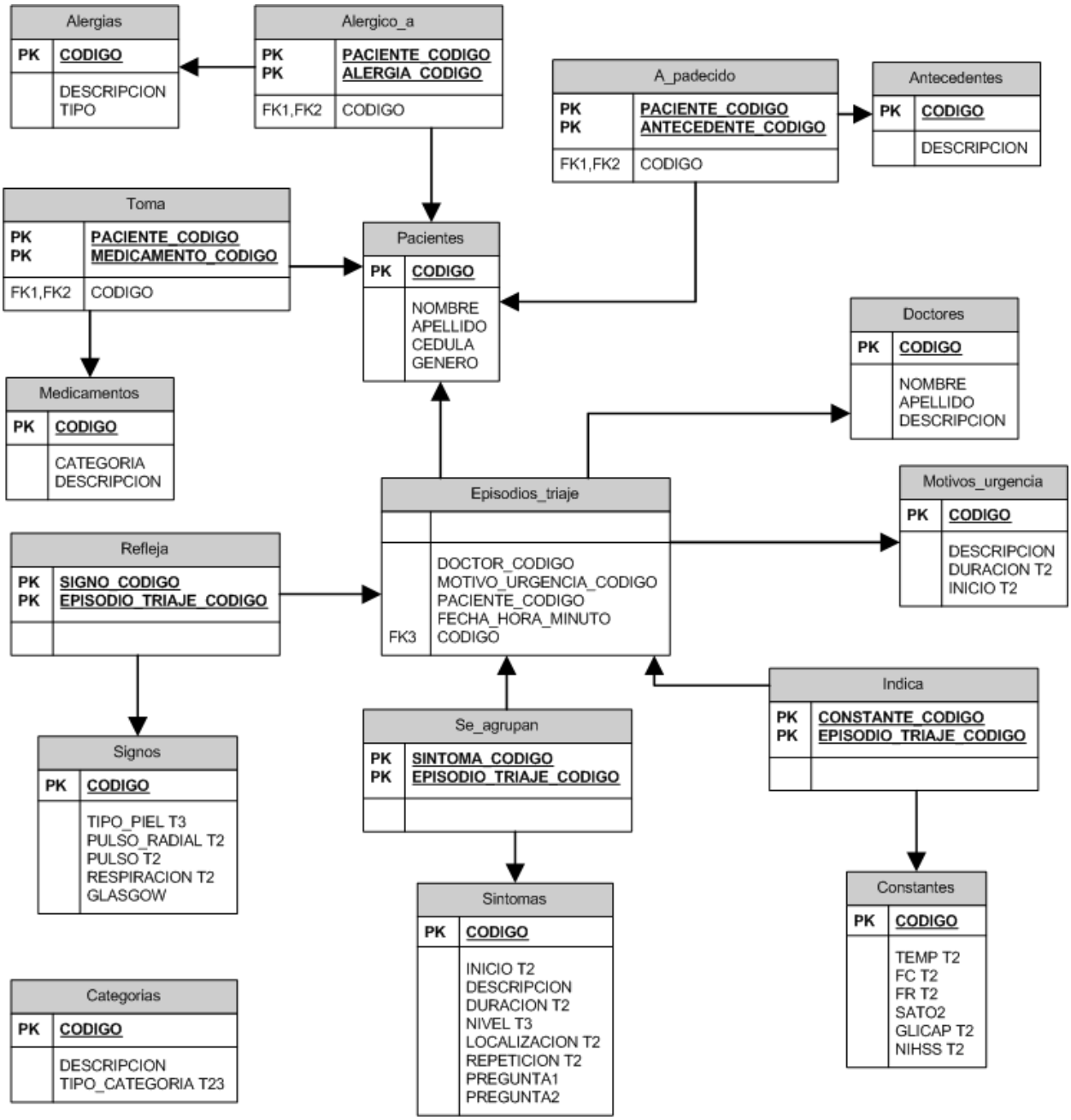

Figura 7. Modelo Relacional difuso de triaje

2.10 Tecnologías Web

La cantidad de personas que utilizan Internet está en aumento (Oliva Moreno, 2003). Desde un punto de vista técnico, este elevado crecimiento fomenta el desarrollo de nuevas tecnologías que aprovechen esta familiaridad. Esto hace prioritario dominar las últimas herramientas de software que se orienten en creación de páginas y aplicaciones Web profesionales, como por ejemplo HTML, PHP, ASP, JSP, JavaScript.

Páginas Web estáticas y dinámicas.

Inicialmente las páginas Web eran estáticas, en el sentido de que, a efectos de usuario, el único proceso realizado era el de la visualización de sus contenidos (escritos en lenguaje HTML) por parte del navegador del cliente. Las páginas estáticas se siguen utilizando ampliamente debido a que forman la base necesaria para la presentación de datos en muchos tipos de situaciones. También influye decisivamente la sencillez con que se pueden crear, instalar y mantener.

Las páginas Web dinámicas surgen por la necesidad de una nteracción mayor entre los usuarios y el sistema, tener un mejor procesamiento de las peticiones de usuario con el fin de ofrecer información más elaborada.

Existen maneras en las que se puede implementar características dinámicas a las páginas web:

- **Procesamiento en el equipo cliente**. Consiste en ejecutar código directamente en el navegador del cliente. Esto involucra la ejecución de elementos como JavaScript.
- **Procesamiento en el equipo servidor**. Consiste en ejecutar código del lado del servidor que será luego desplegado en el cliente. Existen muchas tecnologías capaces de realizar tareas del lado del servidor como asp, jsp, php.
- **Procesamiento mixto**. Consiste en construir aplicaciones que utilicen tanto el procesamiento en el cliente como procesamiento en el servidor. Estas son las aplicaciones más completas.

PHP esta aceptado en la comunidad de software libre como un lenguaje de desarrollo bien estructurado, capaz de resolver muchísimos de los requerimientos de varias industrias sin necesidad de pagar licencias adicionales. El siguiente tópico trata sobre este poderoso lenguaje de programación

PHP

Unas de las tecnologías Web más utilizada y soportada por la comunidad para crear páginas web activas en la actualidad es PHP, por esto se procede al estudio formal de este para averiguar si

este será útil para construir el sistema automatizado de triaje. En (Lerdorf, Hannes, & Olson, 2010) describen a PHP como un lenguaje de código abierto diseñado para construir páginas web dinámicas. PHP permite construir secciones de código embebido dentro de código HTML, es decir solo basta colocar unas etiquetas especiales (<?php ?>) dentro de la página HTML para incrustar un código PHP. Lo que distingue al lenguaje PHP de algún código del lado del cliente como Javascript, es que el código es ejecutado en el servidor, este genera HTML y se envía al cliente.

Un aspecto importante de PHP que sirve para salvaguardar las reglas del negocio, con ninguna posibilidad de determinar qué código ha producido el resultado obtenido. El servidor web puede ser incluso configurado para que procese todos los archivos HTML con PHP y entonces no hay manera que los usuarios puedan saber que se tiene corriendo por debajo.

PHP es un lenguaje interpretado, lo que significa que no es necesario realizar algún proceso de compilación para que el servidor web pueda entender el bloque de código. La principal ventaja de usar PHP es que es extremadamente simple, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. Para poder utilizar PHP es necesario tener instalado un servidor web y el modulo de PHP.

PHP puede ser utilizado en muchos sistemas operativos, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros.

PHP no se encuentra limitado a dar respuestas HTML, con PHP también se puede hacer cosas como: creación de imágenes, archivos PDF, presentar resultados en otros formatos como XHTML y cualquier otro tipo de archivo XML. PHP puede autogenerar éstos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla, creando un caché en el lado del servidor para contenido dinámico.

En general PHP tiene todo lo necesario para plantear reglas de negocio, como condicionales, ciclos además es posible establecer conexiones con las bases de datos de forma sencilla utilizando las capas pdo o adodb.

Modelo arquitectónico Cliente/Servidor

El modelo de arquitectura cliente servidor se compone de dos partes:

- Cliente: normalmente una navegador que transmite una petición y presenta la información al usuario final.

- Servidor: equipo que almacena, recoge, manipula y devuelve datos a una petición.

Dependiendo de la necesidad, se puede tener un modelo cliente/servidor de n-capas, la idea principal de esta especialización del modelo consiste en separar en funcionalidades los elementos dentro del software.

En un modelo de dos capas la capa servidor se encarga de tanto la lógica del negocio, como del almacenamiento y recuperación de los datos. En un modelo de tres capas se tratan por separado: el capa presentación (cliente), la capa lógica (capa intermedia) y la capa de datos.

2.11 Metodología de Desarrollo de Software

Las metodologías de desarrollo: son un conjunto de herramientas, modelos y métodos utilizados para estructurar, planificar y controlar el proceso de desarrollo de software. Según (Weitzenfeld, 2004) existen muchas metodologías, entre la que se puede encontrar la metodología de desarrollo prescriptivas, como lo son la metodología de cascada que consiste en un proceso secuencial de desarrollo donde los pasos son llevados a cabo de arriba hacia abajo y cualquier cambio que surja en alguna de las fases superiores, involucra la perdida de lo ya elaborado, la *metodología de desarrollo UP (Unified Process),* la cual es una metodología de desarrollo iterativo e incremental en la que se definen etapas y disciplinas para tener un mayor control del ciclo de desarrollo; esta metodología se caracteriza por tener una gran cantidad de entregables que en muchas ocasiones son innecesarios para algunos tipos de desarrollo, en general es una metodología de desarrollo que no debería ser utilizada en grupos con pocos desarrolladores.

Por otro lado tenemos las *metodologías ágiles, por ejemplo la metodología de desarrollo XP* por sus siglas en inglés *eXtreme Programming* que se caracteriza principalmente por ser una metodología que se focaliza en el desarrollo y no en la documentación lo que en algunos casos lleva a problemas de mantenimiento. Buscando un equilibrio entre estos dos mundos nace una metodología llamada UP Ágil que busca tener lo mejor de los dos mundos (UP y XP). En (Ambler & Kennaley, 2010) definen AUP (Agile UP) como una versión simplificada de RUP. Esta metodología describe de una manera simple y sencilla el desarrollo de software utilizando técnicas y conceptos ágiles conservando características primordiales de UP, la frase que define que esta metodología es *apenas lo suficientemente bueno* utilizada para referirse a los modelos y documentos asociados al proyecto.

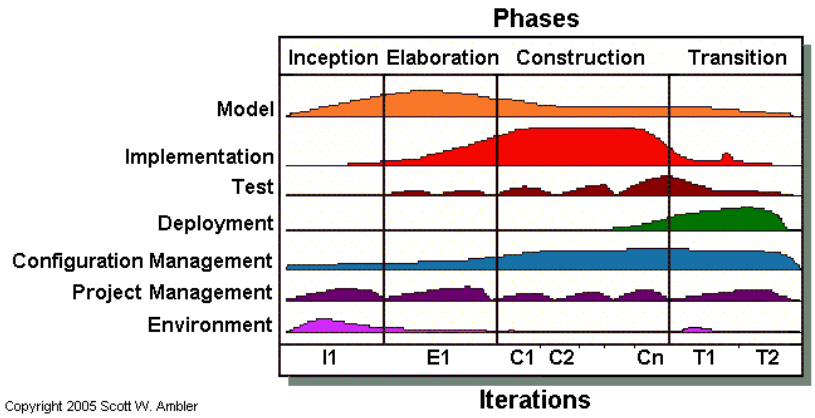

Figura 8. Ciclo de vida AUP. (Ambler, 2009)

El ciclo de vida de AUP consta de 4 fases y 7 disciplinas como podemos ver en la **Tabla 24**.

Tabla 24. Resumen de la metodología AUP

Entregables:
• Manual de usuario • Sistema o Programa • Código Fuente • Requerimientos
Roles:
• DBA • Modelador • Desarrollador • Stakeholder • Documentador • Equipo de prueba
Fases:
• Inicio • Elaboración • Construcción • Transición
Plan de Desarrollo:
• Iterativo e incremental, a medida que se va desarrollando el proyecto las iteraciones se hacen más pequeñas.
Requerimientos:
• Definidos y aceptados por todos antes de la fase de construcción, cualquier cambio de requerimiento involucrara un nuevo proyecto.
Recursos Humanos
• No existe un numero definido de personas, un individuo puede tomar varios roles.
Filosofía
• Solo se entrega apenas lo estrictamente necesario, planificación y mitigación de riesgos, metodología flexible no es necesario cumplir todas las definiciones a cabalidad.

Capítulo III – Marco Aplicativo

En este Capítulo se describen todas las actividades realizadas en cada fase del proceso de desarrollo utilizando la metodología AUP, estas fases son: Fase de Inicio, Fase de Elaboración, Fase de Construcción, y Fase de Transición.

3.1 Fase de Inicio

En la fase de Inicio se describe el análisis del problema desglosado en los hitos alcanzados según la metodología AUP, estos hitos son: Conocimiento del Negocio, Alcance, Requerimientos, Ambiente, Arquitectura, Viabilidad y Stakeholders.

3.1.1 Conocimiento del Negocio

El servicio de emergencia del Hospital Universitario de Caracas realiza varios procesos:

- Asistencial: Este proceso consta de varios pasos: el primero consiste en hacer una cola donde los pacientes van a ser atendidos por orden de llegada. El segundo se lleva a cabo cuando el paciente sale de la cola para ser atendido por un médico de triaje, aquí se toman los datos del paciente, se le pregunta el motivo de consulta, se realiza un examen rápido. Con estos datos se llena una hoja y el médico decide si el paciente debe quedarse para observación, si necesita algún examen paraclínico o si requiere ser hospitalizado. En este proceso la mayoría de los datos que se capturan son imprecisos como tiempos de los síntomas, o temperaturas tomadas sin los métodos/instrumentos adecuados.

- Control: Este ocurre una vez que el paciente ingresa al servicio de emergencia, en este proceso se debe medir el tiempo de duración dentro del recinto, cuantos médicos han visto al paciente, qué medicamentos se le ha dado y con estos datos estimar cual ha sido el costo del paciente.

- Seguimiento e Historia clínica: Consiste en almacenar todos los datos del paciente dentro de una historia clínica, por lo que debe anexarse a la que tiene actualmente o en caso de que no posea se debe crear una nueva.

Rodríguez (2007) indica que es posible mejorar el proceso asistencial aplicando el sistema de triaje (Ver Capitulo 1) en la sala de espera del servicio de emergencia.

3.1.2 Alcance

El alcance se enmarca en realizar un prototipo de sistema de triaje que permita mejorar el proceso asistencial del servicio de emergencia.

3.1.3 Requerimientos

Para cumplir con el alcance se requiere diseñar los siguientes requerimientos:

- Ingreso: Se deben capturar los datos personales del paciente y almacenarlos en la base de datos.

- Motivación previa (Antecedentes): Se debe capturar el motivo de consulta más importante por el que el paciente acude a la emergencia, qué medicamentos está tomando actualmente, a qué medicamentos es alérgico, si padece alguna patología crónica como diabetes, hipertensión, etc. La motivación previa afectará la cantidad u orden de preguntas del triaje.

- Triaje (Algoritmo): Es necesario capturar las variables que establece el modelo de triaje. Estas variables están dividas en grupos de tal forma que se pregunten en un cierto orden según su importancia dentro del triaje.

- Resultados de apoyo: El sistema debe presentar un conjunto de posibles diagnósticos médicos en donde cada diagnóstico tendrá asociado un nivel de urgencia (como es especificado en el sistema de triaje) junto con un cuadro resumen de los datos obtenidos en el triaje.

3.1.4 Plan de trabajo

Se plantea que para la realización de este proyecto se requieren de 3 meses de trabajo, donde se planifican un conjunto de reuniones con los involucrados del proyecto de tal forma que se puedan ir resolviendo las dudas con respecto al modelo que se está trabajando.

3.1.5 Ambiente

Actualmente existen dos desarrollos de FSQL el primero fue en Oracle8i (Galindo, 1999) el cual es una versión estable de FIRST, el segundo es un desarrollo en PostgreSQL actualmente liderado por Urrutia (2003) el cual pretende extender las funcionalidades hasta FIRST-2 (añade sentencias difusas para create, insert entre otros). Tras la evaluación de las herramientas se encontraron

varias fallas en el servidor de Postgres como vistas y funciones aun no traducidas de los PL/SQL de Oracle que pertenecen a FIRST, tampoco se logró establecer un vinculo claro de comunicación con los desarrolladores de esta versión lo que complicó el proceso de evaluación. En cuanto a la versión de Oracle8i la instalación del servidor se realizó bastante limpia, la herramienta FQ proporcionada por Galindo funcionó para comprobar la validez de la instalación y, por último, el vínculo de comunicación siempre fue muy estable con el desarrollador lo que facilitó todas las dudas que surgieron en cuanto al manejo y uso del y uso del servidor. Por otro lado, en php5 existen algunos conflictos para poder establecer la conexión con Oracle8i, tras probar con diferentes instalaciones bajo Windows y Ubuntu se encontró una solución la cual consistió en actualizar algunas librerías, como el caso de oci.dll, a las que vienen en la versión del cliente Oracle 10g xe, esta actualización logro establecer una conexión de datos bastante estable.

Por estas razones se plantea utilizar Oracle 8i, Windows XP, Apache, PHP5 y FIRST como ambiente para la aplicación de triaje.

3.1.6 Arquitectura

En la **Figura 9** se plantea una arquitectura en 3 capas

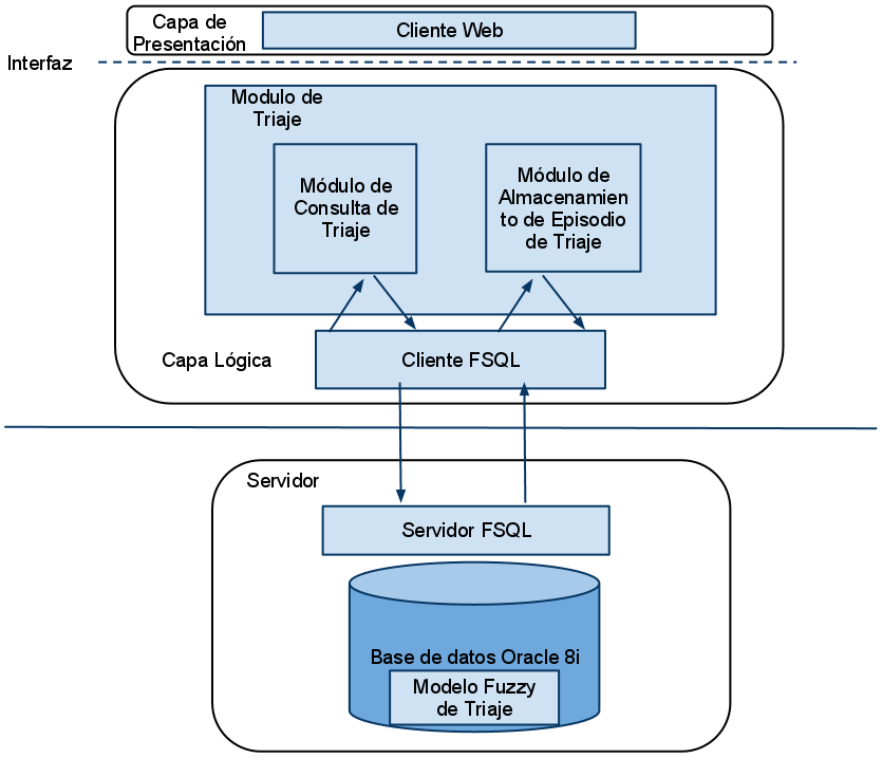

Figura 9. Arquitectura de tres capas para el proceso de triaje

Esta arquitectura estará compuesta de: la capa lógica, la capa de presentación, y el modelo fuzzy de triaje, las cuales constan de:

- **Cliente FSQL**: Este componente servirá como interfaz entre el servidor y el modulo de triaje. Este construye la sentencia INSERT del episodio de Triaje y la enviara al servidor FSQL.
- **Módulo de Triaje**: Este componente contiene toda la lógica del negocio asociada con el sistema de triaje, este componente construirá la consulta FSQL y la pasará de manera estructurada al Cliente FSQL. Dentro de este módulo se encuentra el **Módulo de Almacenamiento de Episodio de Triaje** (denotado como procesamiento)**,** el cual se encarga de recopilar toda la información relacionada con el triaje.
- **Cliente Web**: Interfaz Web que permitirá la comunicación del módulo de triaje con el usuario.
- **Modelo Fuzzy de Triaje**: Implementación física del modelo difuso de triaje.

3.1.7 Viabilidad

Después de estudiar el alcance, el ambiente y la arquitectura, se concluye que es viable construir un prototipo de un sistema Web de apoyo al triaje que utilice un sistema manejador de base de datos difusa.

3.1.8 Stakeholders

Se definen los siguientes roles dentro del proyecto

Nombre	**Descripción**	**Responsabilidades**
Víctor Rodríguez	Médico emergenciólogo, autor del sistema de triaje de la SVMED.	• Representa a todos los usuarios posibles del sistema. • Aprueba requisitos y funcionalidades • Proporciona conocimiento necesario para la realización del sistema
Cristina Martí	Médico cirujano, apoyo de conocimiento técnico.	• Representa a usuarios posibles del sistema. • Proporciona conocimiento necesario para la realización del sistema
Wuilfredo Rangel	Licenciado en computación, lider del proyecto.	• Velar por el cumplimiento del proyecto. • Revisa funcionalidades • Gestiona el canal de comunicación entre todos los involucrados
Jose Schmidt	Analista, modelador y desarrollador del proyecto.	• Analizar los requerimientos del sistema. • Modelar los procesos de negocios • Desarrollar los componentes de software del proyecto • Generar los documentos que apoyan al proyecto.

3.2 Fase de Elaboración

Esta fase busca la estabilidad de la visión del proyecto y de los modelos de arquitectura. Consta de dos iteraciones. En la primera iteración se desarrolla: Estabilidad de la Visión, Estabilidad de la arquitectura. En la segunda iteración se desarrolla: Proceso de triaje, Modelo de casos de uso, e Estabilidad de la arquitectura.

3.2.1 Primera Iteración

Se afianzo el alcance del proyecto, y comprobar la factibilidad según se planteó en la primera fase.

3.2.1.1 Estabilidad de la Visión

El alcance del proyecto se mantuvo, tras otra reunión con los *Stakeholders* el apoyo al proceso de triaje siguió siendo él primer punto a tratar dentro la sistematización de la emergencia. Sin embargo, se presentó un problema en cuanto a la puesta en prueba dentro del HUC dado que los directivos deben analizar la propuesta de triaje desde el punto de vista ético, siendo así señalado por el director de la emergencia, y pasa a ser la Sociedad Venezolana de Medicina de Emergencia y Desastre el ente que comprueba la funcionalidad del prototipo.

3.2.1.2 Estabilidad de la arquitectura

Con este hito se busca comprobar que la arquitectura planteada es viable y lo suficientemente específica como para poder realizar el proyecto.

Los componentes de la arquitectura parecen ser lo suficientemente robustos para modelar el proceso, por lo que se procede a refinar cada uno de estos componentes en elementos más detallados. La **Figura 10** propone la siguiente estructura de archivos para el sistema:

- **Cliente FSQL**: Constará de un archivo que será denominado FSQL.php .
- **Modulo de Consulta de Triaje**: Almacenara la lógica del negocio asociada con el sistema de triaje propuesto por la SVMED, este suministrara de manera estructurada los parámetros al Cliente FSQL.

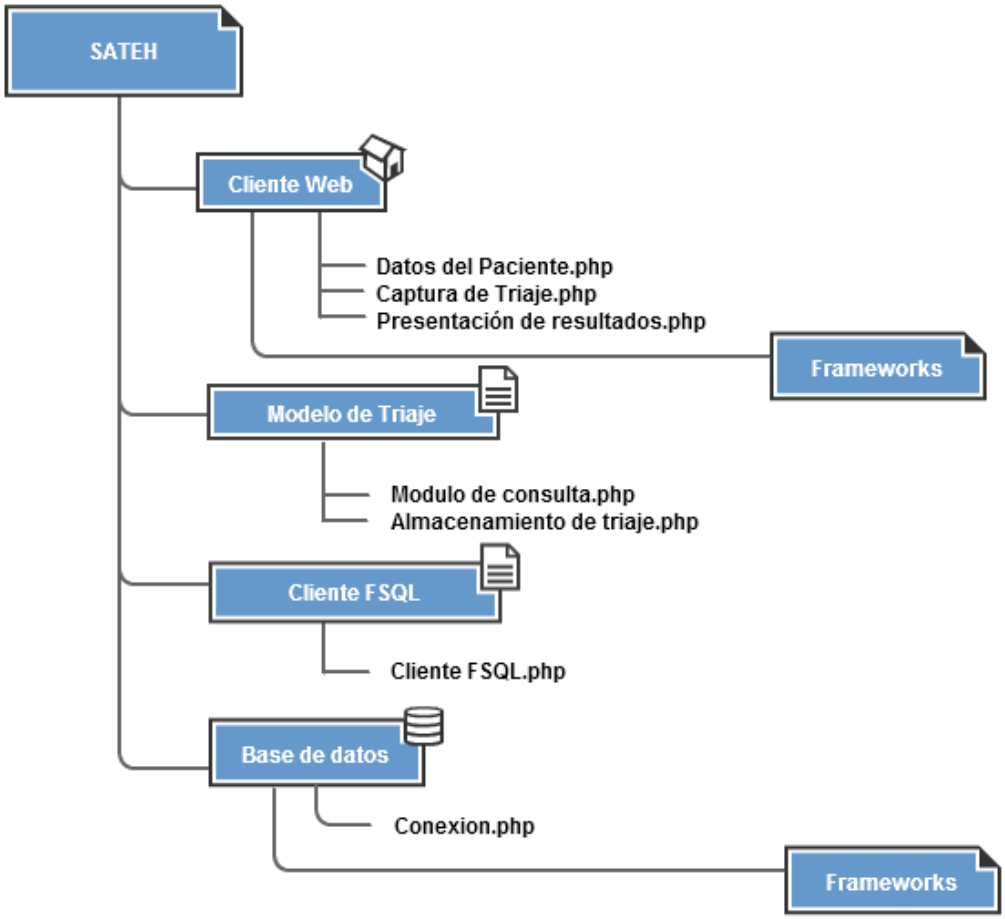

Figura 10. Arquitectura de archivos, primera iteración

- **Módulo de Almacenamiento de Episodio de Triaje**: Se encargara de recopilar toda la información relacionada con el triaje y esta será suministrada de manera estructurada al Cliente FSQL.
- **Cliente Web**: Interfaz Web que permitirá la comunicación del modulo de triaje con el usuario.
- **Modelo Fuzzy de Triaje**: Implementación física en la base de datos del modelo difuso de triaje.

3.2.2 Segunda Iteración

Esta iteración afianzo el conocimiento del negocio, de los procesos y las funcionalidades que se requirieron diseñar dentro del sistema, adicionalmente se extendió y mejoró la arquitectura de archivos antes propuesta.

3.2.2.1 Proceso de triaje

Se ha modelado el flujo de datos de un paciente dentro del proceso de triaje. En este flujo el paciente pasa por tres estados, estos estados son:

- En cola: Un paciente llega a la emergencia, se le toman sus datos y este pasa a la cola de espera dentro de la zona de triaje.

- Clasificado: Cuando el personal médico/asistencial culmina el proceso de triaje se le asigna una prioridad al paciente.

- Atendido: El paciente sale la zona de triaje y pasa al área correspondiente dentro del recinto hospitalario

Para que el paciente alcance el estado de clasificado, el personal médico/asistencial debe realizar el proceso de triaje. La **Figura 11** muestra el algoritmo que describe cómo es la realización de este proceso.

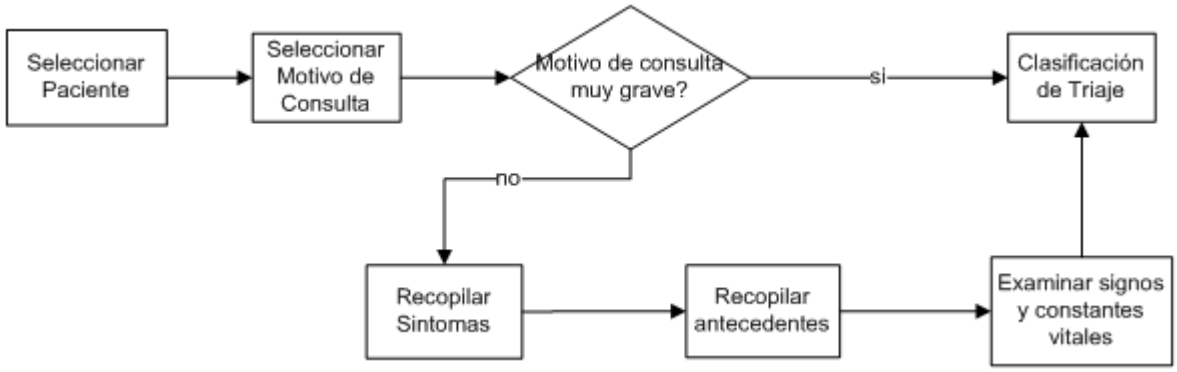

Figura 11. Algoritmo del proceso de triaje

Con el proceso desarrollado se procede a diseñar formalmente las funcionalidades expresadas en diagramas UML de casos de uso, éstas son:

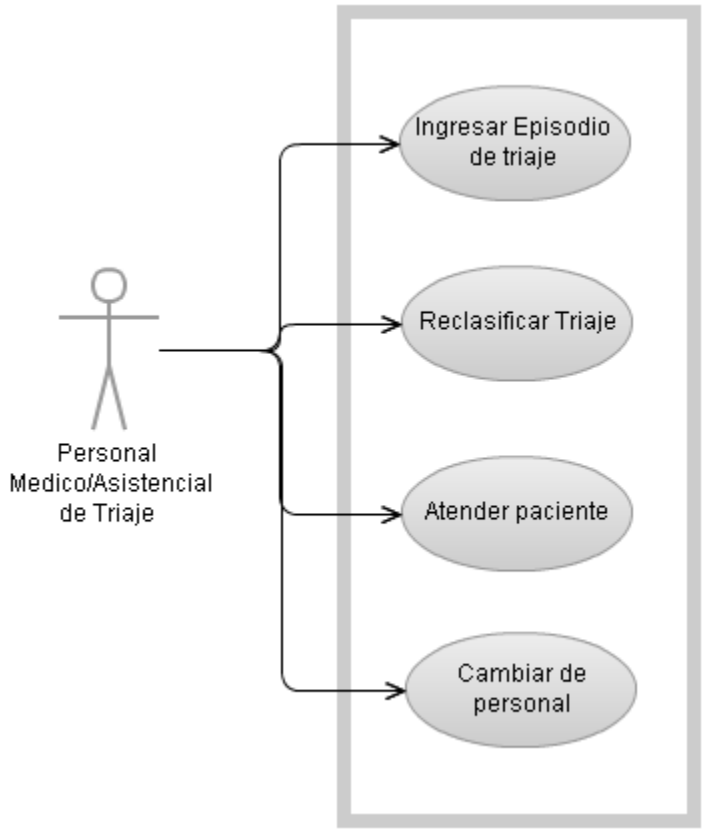

Figura 12. Casos de uso para el proceso de triaje.

CDU001

Caso de Uso ID:	CDU001
Nombre del caso de uso:	Ingresar Episodio de Triaje
Creado por:	Jose Schmidt

Actores:	Personal médico/asistencial de Triaje
Descripción:	Se debe capturar toda la información que lleva al paciente a la emergencia. Esta información consta de: • Motivo de consulta • Síntomas que refleja • Antecedentes que ha padecido • Signos y Constantes vitales

Precondiciones:	1. El paciente debe haber pasado por un ingreso administrativo. 2. El paciente no ha sido previamente clasificado
Postcondiciones:	1. El paciente pasa del estatus *en cola* a *clasificado*
Flujo Normal:	1. Se ingresa un motivo de consulta 2. Se ingresan los síntomas que refleja 3. Se ingresa un antecedente que padezca 4. Se toman los signos y constantes vitales 5. Se clasifica al paciente
Flujo Alternativo:	1. Se ingresa un motivo de consulta muy grave 2. Se clasifica al paciente
Prioridad:	ALTA PRIORIDAD
Frecuencia de uso:	ALTA

CDU002

Caso de Uso ID:	CDU002
Nombre del caso de uso:	Reclasificar Triaje
Creado por:	Jose Schmidt

Actores:	Personal médico/asistencial de Triaje
Descripción:	El paciente no era prioridad 1 o 2 y se ha cumplido el tiempo para éste sea atendido, entonces se debe realizar un nuevo proceso de clasificación. Esta información es: • Signos y Constantes vitales
Precondiciones:	1. No era un paciente grave 2. Transcurrió el tiempo para ser atendido.
Postconditions:	1. El paciente posee una nueva clasificación
Flujo Normal:	1. Se toman los signos y constantes vitales 2. Se clasifica al paciente

Flujo Alternativo:	
Prioridad:	ALTA PRIORIDAD
Frecuencia de uso:	ALTA

CDU003

Caso de Uso ID:	CDU003
Nombre del caso de uso:	Atender paciente
Creado por:	Jose Schmidt

Actores:	Personal médico/asistencial de Triaje
Descripción:	El paciente será atendido por el servicio de emergencia, es necesario quitarlo de la cola de clasificación.
Precondiciones:	1. El paciente debe haber pasado por el proceso de clasificación.
Postcondiciones:	1. El paciente sale del estatus en clasificado para pasar a atendido.
Flujo Normal:	1. Escoger un paciente de la lista
Flujo Alternativo:	
Prioridad:	ALTA PRIORIDAD
Frecuencia de uso:	ALTA

CDU004

Caso de Uso ID:	CDU004
Nombre del caso de uso:	Cambiar de personal
Creado por:	Jose Schmidt

Actores:	Personal médico/asistencial de Triaje
Descripción:	El personal médico/asistencial que realiza el proceso de triaje es variable, se debe poder cambiar el nombre de la persona que estará realizando el triaje.
Precondiciones:	1. No se debe realizar un cambio del personal mientras se esté realizando un proceso de triaje.
Postcondiciones:	1. El sistema cambia la variable del personal que se encuentra realizando el triaje.
Flujo Normal:	1. Escoger un medico de la lista
Flujo Alternativo:	
Prioridad:	MEDIA
Frecuencia de uso:	ALTA

La definición de la arquitectura de archivos de la primera iteración no es suficiente, es necesario detallar aun más esta arquitectura para comenzar con la fase de desarrollo.

3.2.2.2 Arquitectura de archivos

La **Figura 13** representa la arquitectura de archivos propuesta en esta iteración. El enfoque de esta arquitectura no es saber cuáles serán los archivos a construir sino establecer una arquitectura organizada que facilite el proceso de mantenimiento y extensibilidad de la aplicación.

Estudiando esta arquitectura se tiene:

- **Cliente Web**: Para capturar los datos se diseñarán formularios y estos deben localizarse dentro de la carpeta denominada *Formularios*. Para facilitar la mantenibilidad del código se diseñarán componentes reutilizables, estos componentes deben estar almacenados dentro de la carpeta *Partes*.

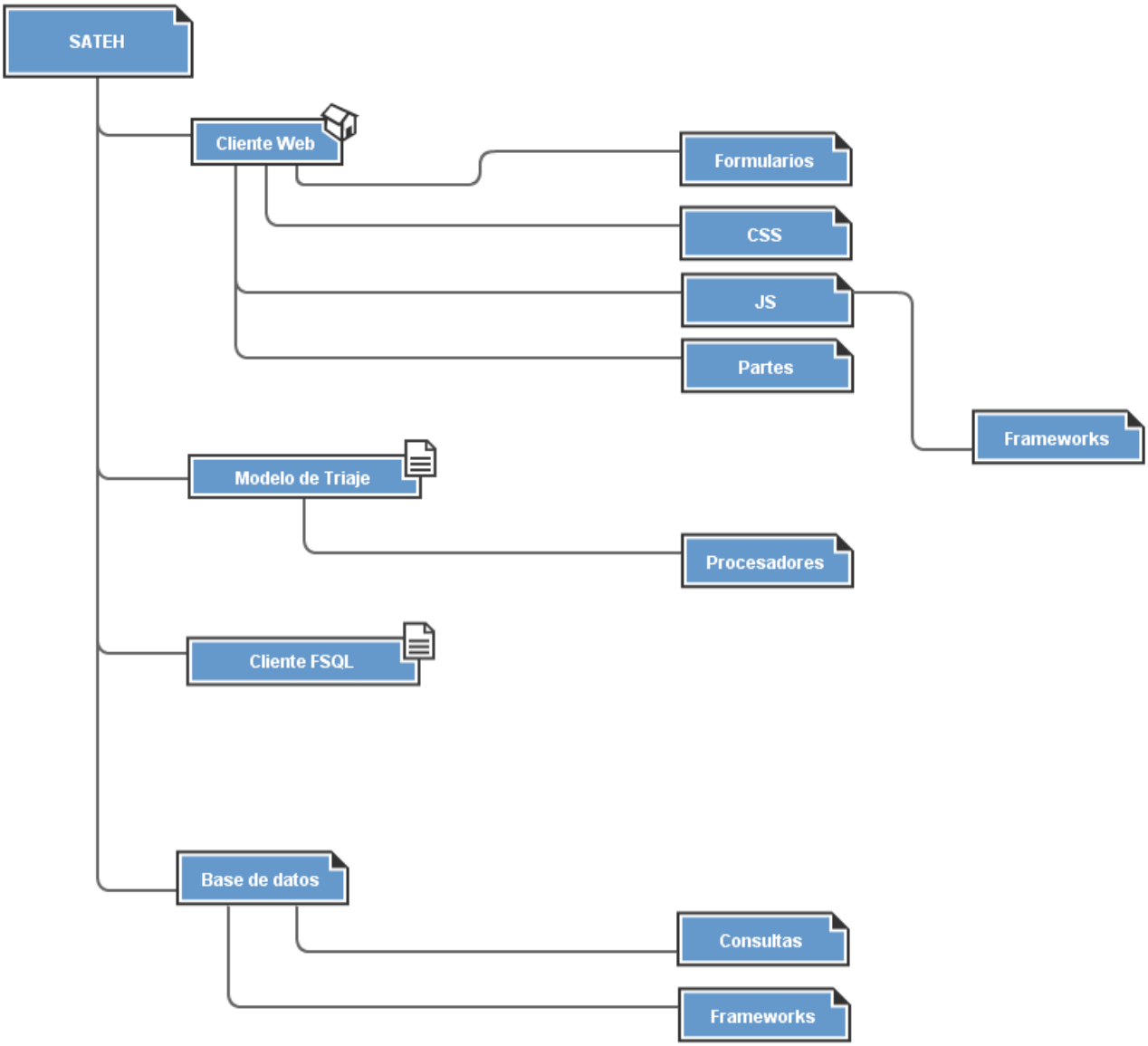

Figura 13. Arquitectura de archivos en la segunda iteración

El diseño de formularios que permita capturar datos difusos no puede ser alcanzado utilizando Forms estándar de HTML, por esta razón se desea utilizar un Framework que brinde extensibilidad a los Forms. Estos deberán encontrarse dentro de una carpeta denominada *js*.

- **Modulo de Triaje**: Existirán dos familias de archivos dentro de esta carpeta. La primera procesa la información de los formularios, estos estarán en una carpeta denominada *procesadores*. La segunda almacenará las reglas difusas del triaje.

- **Cliente FSQL**: Aquí se encontraran todos los archivos que procesan las reglas difusas de triaje, y los que transforman datos clásicos de los formularios a valores difusos.

- **Base de datos**: Archivos que contengan información de la conexión con el servidor de base de datos, consultas a bases de datos deberán encontrarse en esta carpeta. Así mismo cualquier framework necesario para poder facilitar los trabajos de conexión se encontrarán dentro de una carpeta denominada *frameworks.*

3.3 Fase de Construcción

El desarrollo de esta fase se dividió en cuatro (04) módulos: el primero se trata de la construcción del esquema relacional de base de datos difuso, el segundo es la construcción del Cliente FSQL, el tercero la construcción del módulo de triaje y por último está el cliente Web.

3.3.1 Esquema relacional difuso

Para construir el esquema relacional difuso de triaje se debe crear las tablas con un número adecuado de campos por cada atributo difuso, luego procedemos a almacenar la información de esos campos dentro de las seis tablas que componen al FMB.

3.3.2 Creación de los esquemas

La primera labor es construir los esquemas que permitan almacenar datos difusos, para explicar cómo se crean los esquemas difusos se hablará de cómo crear la tabla síntomas, la cual consta de cuatro atributos difusos, tres de los cuales son atributos tipo dos y uno es tipo tres.

- T2 DURACION: Se requieren 4 campos para poder almacenar datos difusos de duración, estos campos son:
    - DURACIONT: Campo numérico que almacena información de cuál de los 7 subtipos de datos difusos almacena ese registro, en el caso particular de esta aplicación esos valores serán 1 (UNDEFINED) ó 6 (Aproximación triangular).
    - DURACION1…4: Campos numéricos que se utilizan para almacenar el propio valor difuso. Cuando se utiliza 1 para el subtipo, los valores de estos campos son obviados. Cuando se utiliza el subtipo 6, el primer campo almacenará el valor central del triángulo, el segundo almacenará el mínimo izquierdo del triangulo, el tercero almacenara el valor derecho del triangulo y en cuarto estará cuanto es el margen entre el centro y los extremos.

Todos los atributos restantes tipo 2 de todas las tablas se crean de esta misma manera que DURACION.

- T2 INICIO:
  - INICIOT
  - INICIO1…4

- T2 REPETICION
  - REPETICIONT
  - REPETICION1…4

- T3 NIVEL: El número de campos que puede tener un atributo tipo 3 varía de acuerdo a la necesidad que esté modelando, donde se tiene que crear una dupla de campos por cada etiqueta contra la que se quiera comparar un registro. En el caso del atributo nivel sólo es necesario almacenar una comparación por lo que los campos quedan

  - NIVELT: Campo numero que almacena información de cuál de los 4 subtipos difusos almacena un registro. En el caso de NIVEL los posibles valores serán 1 (UNDEFINED) ó 3 el cual indica que se va a comparar contra solo una de las etiquetas definidas para este atributo.
  - NIVELP1: Campo numérico que almacena cual es el id de la etiqueta contra la que se va a comparar dentro del registro.
  - NIVEL1: Campo numérico que almacena el grado de pertenencia de ese registró a esa etiqueta.

El bloque de **Código 1** muestra la sentencia create para la tabla síntomas.

```
CREATE TABLE SINTOMAS(
CODIGO NUMBER(4) NOT NULL,
DESCRIPCION VARCHAR (150),
LOCALIZACION VARCHAR (150) DEFAULT 'NOAPLICA' NOT NULL,
PREGUNTA1 CHAR(2) DEFAULT 'NO' NOT NULL,
PREGUNTA2 CHAR(2) DEFAULT 'NO' NOT NULL,
-- LA DURACIÓN ES UN ATRIBUTO DIFUSO TIPO 2 QUE MIDE CUANTOS MINUTOS LE DURA
-- AL PACIENTE EL SINTOMA POR CADA REPETICIÓN
DURACIONT NUMBER(4) DEFAULT 0 NOT NULL,
DURACION1 NUMBER(3),
DURACION2 NUMBER(3),
DURACION3 NUMBER(3),
DURACION4 NUMBER(3),
-- LA FECHA DE INICIO ES UN ATRIBUTO DIFUSO TIPO 2, EN LA QUE SE PUEDEN
-- ALMACENAR VALORES COMO HACE POCO, HACE MUCHO, ETC
INICIOT NUMBER(1) DEFAULT 0 NOT NULL,
INICIO1 NUMBER(3),
INICIO2 NUMBER(3),
```

```
INICIO3 NUMBER(3),
INICIO4 NUMBER(3),
-- LA REPETICIÓN DEL SINTOMA DEL PACIENTE SE CUENTA POR DÍA, PUEDE TOMAR
-- VALORES COMO POCO, MAS O MENOS, O MUCHO
REPETICIONT NUMBER(1) DEFAULT 1 NOT NULL,
REPETICION1 NUMBER(3),
REPETICION2 NUMBER(3),
REPETICION3 NUMBER(3),
REPETICION4 NUMBER(3),
/* LA ESCALA DE NIVEL ES UN ATRIBUTO DIFUSO TIPO 3, EN EL QUE PODEMOS
ALMACENAR VALORES COMO LIGERO, MODERADO, INTENSO
-- ETC.*/
NIVELT NUMBER(1) DEFAULT 0 NOT NULL,
-- SOLO HACE FALTA COMPARAR LAS ETIQUETAS, POR LO QUE EXISTEN SOLO 2
-- COLUMNAS PARA LA DEFINICIÓN DEL ATRIBUTO
NIVELP1 NUMBER(3,2),
-- AQUI ALMACENA CUANTO ES PARECIDO A LA ETIQUETA DE
-- NIVEL
NIVEL1 NUMBER(3),
--AQUI ALMACENA EL ID DE LA ETIQUETA CONTRA LA QUE SE ESTA COMPARANDO
PRIMARY KEY (CODIGO)
) TABLESPACE TRIAJE;
```

Código 1. Sentencia create para la tabla síntomas.

El resto de los atributos tipo 3 de las demás tablas se manejan de esta misma manera.

Luego de construir los esquemas de la base de datos es necesario informar al FMB acerca de los campos difusos de hemos creado. Como lo sugiere el autor del FSQL Server, en esta fase se ha construido un PL/SQL para alcanzar este objetico el cual tiene como base la siguiente estructura

3.3.2 Almacenando información en la FMB.

Para poder almacenar la información del esquema difuso es necesario guardar dentro del FMB datos de: ID del objeto tabla que contiene por lo menos un atributo difuso; ID de la columna NOMBRET que contiene la información del subtipo difuso asociado; tipo de atributo difuso asociado: colocar si es tipo 1,2,3 o 4; lista de etiquetas: se deben almacenar todas las etiquetas tanto su descripción como los valores que representan; valores de proximidad: en caso de se vayan a almacenar registros donde alguno de sus atributos difusos tipo 2 del subtipo aproximación triangular es necesario almacenar cuál es el margen de el triangulo, y cuál es la distancia que se considera mínima para decir que dos valores difusos se encuentran muy lejos; compatibilidad entre atributos: en caso que se quiera establecer comparaciones entre atributos difusos tipo 3; grado de cercanía: para los atributos difusos tipo 3 es necesario almacenar cual es la relación entre cada una de las etiquetas del mismo objeto. Dado fue necesario escoger un tipo de dato difuso para almacenar los valores, se escogió de forma arbitraria la función triangular.

A continuación se presenta un bloque de código PL/SQL que nos permite almacenar la información de la tabla síntomas dentro del FMB

```
DECLARE
T_SINTOMAS              NUMBER;
-- TABLA SINTOMAS
C_INICIO_S              NUMBER;
C_NIVEL                 NUMBER;
C_DURACION_S            NUMBER;
C_REPETICION            NUMBER;
```

Código 2. Inicio del PL/SQL para la inserción dentro de la FMB

En este primer bloque de código se declaran las variables que utilizaremos a los largo del PL, dado que vamos a necesitar el ID de la tabla Síntomas y el ID de las columnas difusas de este objeto asignaremos esos valores a una variable.

```
SELECT OBJECT_ID INTO T_SINTOMAS FROM USER_OBJECTS WHERE
OBJECT_NAME='SINTOMAS';

SELECT COLUMN_ID INTO C_INICIO_S
   FROM USER_TAB_COLUMNS WHERE TABLE_NAME='SINTOMAS' AND
COLUMN_NAME='INICIOT';

SELECT COLUMN_ID INTO C_NIVEL
   FROM USER_TAB_COLUMNS WHERE TABLE_NAME='SINTOMAS' AND
COLUMN_NAME='NIVELT';

SELECT COLUMN_ID INTO C_DURACION_S
   FROM USER_TAB_COLUMNS WHERE TABLE_NAME='SINTOMAS' AND
COLUMN_NAME='DURACIONT';

SELECT COLUMN_ID INTO C_REPETICION
   FROM USER_TAB_COLUMNS WHERE TABLE_NAME='SINTOMAS' AND
COLUMN_NAME='REPETICIONT';
```

Código 3. Búsqueda de los IDS de tablas y columnas del PL/SQL para la inserción dentro de la FMB

En este Segundo bloque de código obtenemos los ids de las tablas difusas y luego de las columnas difusas.

```
INSERT INTO FCL VALUES (T_SINTOMAS,C_INICIO_S,2,1,USER||'.SINTOMAS.INICIO');
INSERT INTO FCL VALUES (T_SINTOMAS,C_DURACION_S,2,1,USER||'.SINTOMAS.DURACION');
INSERT INTO FCL VALUES (T_SINTOMAS,C_NIVEL,3,1,USER||'.SINTOMAS.NIVEL');
INSERT INTO FCL VALUES (T_SINTOMAS,C_REPETICION,2,1,USER||'.SINTOMAS.REPETICION');
```

Código 4. Almacenamiento dentro de la tabla FCL

En la tabla FCL se debe almacenar los valores de identificación de las columnas difusas, empezando por el id de la tabla donde se encuentra, el id de la columna del subtipo difuso, el tipo de atributo difuso, un valor que actualmente no utiliza el fuzzy server que se puede dejar en 1, y por último debemos especificar el alias con el que realmente vamos a identificar el atributo difuso. Por ejemplo para poder ver la data que contiene información acerca del nivel de un síntoma no haremos una consulta a nivelt o nivel1...4, haremos una consulta utilizando el identificador *nivel*.

```
INSERT INTO FOL VALUES (T_SINTOMAS,C_INICIO_S,0,'HACE_POCO',0);
INSERT INTO FOL VALUES (T_SINTOMAS,C_INICIO_S,1,'MAS_O_MENOS',0);
INSERT INTO FOL VALUES (T_SINTOMAS,C_INICIO_S,2,'HACE_MUCHO',0);

INSERT INTO FOL VALUES (T_SINTOMAS,C_NIVEL,0,'LIGERO',1);
INSERT INTO FOL VALUES (T_SINTOMAS,C_NIVEL,1,'MODERADO',1);
INSERT INTO FOL VALUES (T_SINTOMAS,C_NIVEL,2,'INTENSO',1);

INSERT INTO FOL VALUES (T_SINTOMAS,C_DURACION_S,0,'POCO',0);
INSERT INTO FOL VALUES (T_SINTOMAS,C_DURACION_S,1,'MAS_O_MENOS',0);
INSERT INTO FOL VALUES (T_SINTOMAS,C_DURACION_S,2,'MUCHO',0);

INSERT INTO FOL VALUES (T_SINTOMAS,C_REPETICION,0,'POCO',0);
INSERT INTO FOL VALUES (T_SINTOMAS,C_REPETICION,1,'MAS_O_MENOS',0);
INSERT INTO FOL VALUES (T_SINTOMAS,C_REPETICION,2,'MUCHO',0);
```

Código 5. Almacenamiento dentro de la tabla FOL.

El siguiente paso consiste en almacenar la información de las etiquetas de los atributos difusos dentro de la FOL, para lograrlo necesitamos colocar primero el id de la tabla donde se aplica, el id de la columna donde se aplica, una numeración que establece además de un identificador para la etiqueta un orden dentro de las etiquetas de este atributo, seguido de un conjunto de caracteres expresados en mayúsculas que representara el alias por el que podemos utilizarlos dentro de las consultas, y por ultimo un valor que indica si este atributo es tipo 1,2 o 3, se utiliza 0 para cualquier atributo tipo 1 o 2, y 1 para un atributo tipo 3.

```
INSERT INTO FLD VALUES (T_SINTOMAS,C_INICIO_S,0,0,1,14,24);
INSERT INTO FLD VALUES (T_SINTOMAS,C_INICIO_S,1,15,30,72,96);
INSERT INTO FLD VALUES (T_SINTOMAS,C_INICIO_S,2,84,96,10000,10001);

INSERT INTO FLD VALUES (T_SINTOMAS,C_DURACION_S,0,0,1,5,10);
INSERT INTO FLD VALUES (T_SINTOMAS,C_DURACION_S,1,8,15,25,28);
INSERT INTO FLD VALUES (T_SINTOMAS,C_DURACION_S,2,25,30,10000,10001);

-- LA REPETICION MIDE EN CUANTAS VECES POR DIA REPITE EL SINTOMA
INSERT INTO FLD VALUES (T_SINTOMAS,C_REPETICION,0,0,1,5,10);
INSERT INTO FLD VALUES (T_SINTOMAS,C_REPETICION,1,6,8,12,15);
INSERT INTO FLD VALUES (T_SINTOMAS,C_REPETICION,2,10,12,10000,10001);
```

Código 6. Almacenamiento dentro de la tabla FLD

El valor de las etiquetas para atributos difusos tipo 2 se almacena en la tabla FLD de la siguiente manera: primero almacenamos el valor tanto de la tabla como de la columna donde a la que pertenece esta etiqueta, luego el identificador de la etiqueta como se definió en el bloque de código anterior, por último los valores del trapecio que pertenecen a la etiqueta.

```
INSERT INTO FAM VALUES (T_SINTOMAS,C_INICIO_S,8,32);
INSERT INTO FAM VALUES (T_SINTOMAS,C_DURACION_S,3,15);
INSERT INTO FAM VALUES (T_SINTOMAS,C_REPETICION,3,15);
```

Código 7. Almacenamiento dentro de la tabla FAM

Dado que en nuestra aplicación estaremos utilizando registros de aproximación triangular, es necesario guardar estos valores dentro de la FAM de la siguiente manera: id de la tabla, el id de la columna, el valor del margen y por ultimo un valor que denota la distancia mínima para que dos valores se encuentren muy lejanos.

```
INSERT INTO FND VALUES (T_SINTOMAS,C_NIVEL,0,1,0.5);
INSERT INTO FND VALUES (T_SINTOMAS,C_NIVEL,0,2,0);
INSERT INTO FND VALUES (T_SINTOMAS,C_NIVEL,1,2,0.5);
```

Código 8. Almacenamiento dentro de la tabla FND

Las relaciones entre las etiquetas de los atributos difusos tipo 3 se almacenan dentro de la FND de la siguiente forma: id de la tabla, id de la columna, etiqueta de la que se quiere comparar, etiqueta contra la que se quiere comparar y el valor de la comparación.

Con ese último bloque termina la definición de los atributos difusos para la tabla síntoma, el resto de las tablas siguen los mismos pasos.

Definido ya el esquema difuso de base de datos, el siguiente paso es la construcción del cliente FSQL el cual se encargara de entender las consultas difusas, y de transformar los valores de los formularios para en valores difusos.

3.3.3 Cliente FSQL

Como se expresó en la fase anterior, dentro del cliente FSQL se requieren construir dos familias de archivos, la primera que se encargara de transformar los datos recibidos dentro de los formularios en datos difusos y almacenarlos así dentro de la base de datos, y la segunda que se encargará de procesar las consultas difusas y retornar el valor de los campos especificados dentro de la consulta

3.3.3.1 **Transformación a datos difusos**

Los datos que vienen de los formularios son en su mayoría números clásicos, para transformar estos datos en números difusos, se tienen dos alternativas, la primera para transformar atributos difusos tipo 2 la cual utiliza una aproximación triangular de tal forma que se aplican los márgenes al número clásico y así se obtienen los datos difusos, la segunda para transformar valores atributos difusos tipo 3, donde se le asignará un valor de posibilidad a la etiqueta donde ese valor tenga más pertenencia.

El paciente puede presentar uno o más síntomas y se debe hacer un *insert* dentro de la base de datos por cada uno. El siguiente bloque de código muestra cómo hacer un *insert* para uno de los síntomas más completos que es dolor

```
//Valor de los márgenes para los atributos difusos duración, inicio y repetición.
$DURACION_TRIANGULO=3;
$INICIO_TRIANGULO=8;
$REPETICION_TRIANGULO=3;

//Todos los síntomas tienen una duración, un inicio y una cantidad de repeticiones, el siguiente bloque de código busca
//reducir la cantidad de líneas de código de cada síntoma.

//Se debe almacenar el subtipo del valor difuso para el atributo, podrá ser 1 cuando no está definido ó 6 cuando está definido,  //se debe almacenar el valor central del triangulo, Sumar  y restar el margen al valor central y por ultimo guardar el margen.
$D_I_R_SQL=
'\''.$sintoma['DURACIONT'].'\',\''.$sintoma['DURACION'].'\',\''.($sintoma['DURACION']-
$DURACION_TRIANGULO).'\',\''.($sintoma['DURACION']+$DURACION_TRIANGULO).'\',\''.$DURACION_TRIANGULO.'\',

 \''.$sintoma['INICIOT'].'\',\''.$sintoma['INICIO'].'\',\''.($sintoma['INICIO']-
$INICIO_TRIANGULO).'\',\''.($sintoma['INICIO']+$INICIO_TRIANGULO).'\',\''.$INICIO_TRIANGULO.'\',

\''.$sintoma['REPETICIONT'].'\',\''.$sintoma['REPETICION'].'\',\''.($sintoma['REPETICION']-
$REPETICION_TRIANGULO).'\',\''.($sintoma['REPETICION']+$REPETICION_TRIANGULO).'\',\''.$REPETICION_TRIANGULO.'\',';

      SWITCH($sintoma['NIVEL']){
          //Transformamos la escala de dolor en alguno de los tres valores del atributo nivel.
          case 1:  $DOLOR_P=0;$DOLOR_VALOR=1;  BREAK;
          case 2:  $DOLOR_P=0;$DOLOR_VALOR=0.6;BREAK;
          case 3:  $DOLOR_P=0;$DOLOR_VALOR=0.3;BREAK;
          case 4:  $DOLOR_P=1;$DOLOR_VALOR=0.6;BREAK;
          case 5:  $DOLOR_P=1;$DOLOR_VALOR=1;  BREAK;
          case 6:  $DOLOR_P=1;$DOLOR_VALOR=0.6;BREAK;
          case 7:  $DOLOR_P=2;$DOLOR_VALOR=0.7;BREAK;
          case 8:  $DOLOR_P=2;$DOLOR_VALOR=0.8;BREAK;
          case 9:  $DOLOR_P=2;$DOLOR_VALOR=0.9;BREAK;
          case 10: $DOLOR_P=2;$DOLOR_VALOR=1;  BREAK;
          }

       //Construcción del insert difuso
       $DOLOR_sql='insert into SINTOMAS values
(\'\',\''.$sintoma['DESCRIPCION'].'\','.$sintoma['LOCALIZACION'].',\'NO\',\'NO\',';
       $DOLOR_sql= $DOLOR_sql.$D_I_R_SQL;
       $DOLOR_sql= $DOLOR_sql.'\'3\',\''.$DOLOR_VALOR.'\',\''.$DOLOR_P.'\') ';
```

Código 9. Transformación a un valor difuso de dolor

De esta forma el string $DOLOR_sql contiene la sentencia *insert* con los valores difusos. El resto de los síntomas, las constantes y signos vitales pasan por un proceso de transformación muy similar al de dolor.

Para poder ejecutar consultas difusas es necesario construir una interfaz con el servidor FSQL, como se explica en el siguiente punto.

3.3.3.2 Ejecución de consulta difusa

El servidor FSQL consta de un conjunto de funciones y procedimientos dentro del manejador de base de datos, que se encargan de procesar la consulta difusa. Una de las funciones del servidor FSQL es FSQL2SQL la cual puede ser utilizada para facilitar este proceso de traducción.

El proceso para ejecutar una consulta difusa utilizando la función FSQL2SQL es la siguiente:

- La función FSQL2SQL recibe como parámetro un string el cual debe contener la consulta difusa.
- Si esta función devuelve 0 significa que se ha transformado correctamente la consulta difusa. En caso contrario significa que existe algún tipo de error en la consulta.
- La vista FSQL_QUERY muestra la traducción de la consulta difusa en términos de las funciones PL/SQL, sin embargo este resultado se encuentra disgregado en varios registros del campo atributo por lo que es necesario realizar un proceso de concatenación de este atributo para poder tener la consulta que realmente se va a ejecutar.

Para poder ejecutar la función FSQL2SQL se debe preparar un pequeño bloque PL, donde adicionalmente se utiliza un enlace que guarde el resultado de la función

```
$fsql2sql = "
    DECLARE
        numero number;
        c     string(2000);
    BEGIN
        c :='".$diagnostico."';
        :numero := FSQL_PKG.FSQL2SQL (c);
        DBMS_OUTPUT.PUT_LINE(numero);
    END;
";
$fsql = $dbh -> PrepareSP($fsql2sql);
$dbh -> OutParameter($fsql,$numero,'numero');
```

Código 10. Llamado a la función FSQL2SQL

Donde la variable $diagnostico contiene la consulta difusa y la variable numero almacenará el valor retornado por la función.

La concatenación del campo atributo de la tabla FSQL_QUERY se logra utilizando un ciclo, donde después de cada iteración se le agregara el valor del campo atributo seguido de un espacio en blanco

```
foreach($concatenar_sql as $j => $palabra){
  if (strstr($palabra['ATRIBUTO'], """)){
    $palabra['ATRIBUTO']=str_replace("""","""",$palabra['ATRIBUTO']);
  }
  $sql = $sql.$palabra['ATRIBUTO'].' ';
```

Código 11. Concatenación de la columna atributo

Por último, este componente almacenará los resultados o posibles diagnósticos de los pacientes dentro de la variable de sesión $_SESSION ['DIAGNOSTICOS'].

Ya establecido el esquema relacional difuso y el cliente FSQL que se conecte al esquema el siguiente paso consta de armar un módulo donde se establezcan las reglas de triaje.

3.3.4 Módulo de triaje

En este módulo se encontrarán dos tipos de archivos, el primer tipo son las consultas o reglas difusas de triaje, y el segundo son los archivos procesadores de los formularios. El **Anexo 1** muestra los valores que se utilizaron como muestra para establecer estas reglas.

3.3.4.1 **Reglas**

El diseño adoptado en la construcción de las reglas de triaje es: establecer una variable donde se encuentre las tablas que pertenecen al *from*, y otra donde se encuentran las columnas que pertenecen al *where*, de esta forma sólo es necesario crear el string que tenga la regla difusa y concatenar esto con las variables del *from* y el *where*.

```
$from='
        SINTOMAS SINT,
        MOTIVOS_URGENCIA M_U,
        EPISODIOS_TRIAJE ET,
        PACIENTES P,
        A_PADECIDO A_P,
        ANTECEDENTES ANT,
        INDICA IND,
        CONSTANTES CONS,
        SIGNOS SIG,
        REFLEJA REF,
        SE_AGRUPAN S_A';

$join='
      P.CODIGO='.$_SESSION['paciente'][0].' AND
```

```
            ET.PACIENTE_CODIGO='.$_SESSION['paciente'][0].' AND
            ET.CODIGO='.$_SESSION['IDS']['EPISODIO_ID'].' AND
            A_P.PACIENTE_CODIGO='.$_SESSION['paciente'][0].' AND
            A_P.ANTECEDENTE_CODIGO=ANT.CODIGO AND
            IND.EPISODIO_TRIAJE_CODIGO='.$_SESSION['IDS']['EPISODIO_ID'].' AND
            CONS.CODIGO=IND.CONSTANTE_CODIGO AND
            REF.SIGNO_CODIGO=SIG.CODIGO AND
            REF.EPISODIO_TRIAJE_CODIGO='.$_SESSION['IDS']['EPISODIO_ID'].' AND
            S_A.EPISODIO_TRIAJE_CODIGO='.$_SESSION['IDS']['EPISODIO_ID'].' AND
            S_A.SINTOMA_CODIGO=SINT.CODIGO AND';
```

Código 12. Bloque de código para el from y el where de un query estándar

```
$diagnosticos[0]="SELECT ''URGENCIA HIPERTENSIVA'' AS CATEGORIA, ANT.DESCRIPCION,
CDEG(*)
            FROM ".$from."
            WHERE ".$join."

          CONS.TAS FGT 170 AND
          CONS.NIHSS FEQ \$ALERTA AND
          CONS.FR FEQ \$NORMAL AND
          M_U.DESCRIPCION <> ''FIEBRE'' AND
          M_U.DESCRIPCION <> ''DIARREA'' AND
          SINT.DESCRIPCION LIKE ''DOLOR %'' AND SINT.LOCALIZACION=''CABEZA''

";
```

Código 13. Bloque de código que representa una regla de triaje

Las variables del arreglo $diagnosticos almacenan la construcción de la consultas difusas, el bloque de arriba muestra como se representa una regla difusa en una consulta FSQL.

En algunos casos, puede que para unas reglas de triaje se requiera consultar más de un síntoma, por esta razón se construyeron las variables $from_N_sintomas y $from_N_sintomas donde N es el numero de síntomas que requiere la regla y tiene como máximo el valor 4(ej: $from_2_sintomas).

Algunas reglas de triaje son sumamente directas, como son “motivos de consulta muy graves”; el tratamiento de estas reglas son tratadas aparte dentro de un componente interno denominado *reglas*. Este componente altera el flujo normal de triaje, de tal forma que no sea necesario completar todo el cuestionario de triaje. El estilo de elementos dentro de este componente es el siguiente:

```
 if ($motivo == 'POLITRAUMATISMO SEVERO' or $motivo == 'NO RESPIRA' or $motivo == 'SIN PULSO'
or $motivo == 'ENTUBACIÓN PREHOSPITALARIA' or $motivo == 'PACIENTE NO RESPONDE' ){
      if ($motivo == 'SIN PULSO'){
            $_SESSION['DIAGNOSTICOS'][0]=array('CATEGORIA'=>'PARO CARDIACO','CDEG(*)'=>1);}
        ELSEIF($motivo == 'NO RESPIRA'){
            $_SESSION['DIAGNOSTICOS'][0]=array('CATEGORIA'=>'PARO
RESPIRATORIO','CDEG(*)'=>1);}
        ELSE{
```

```
            $_SESSION['DIAGNOSTICOS'][0]=array('CATEGORIA'=>'PRIORIDAD 1','CDEG(*)'=>1);}
   echo ($motivo);
   echo $_SESSION['DIAGNOSTICOS'][0][0];
   header('Location: ../Cliente_Web/resultado.php');
```

Código 14. Reglas de triaje no difusas que alteran el flujo normal de datos

El otro componente dentro del modulo es el encargado de procesar los formularios del proceso de triaje.

3.3.4.2 Procesadores de formularios

Este componente es utilizado para procesar los valores obtenidos de los formularios del proceso de triaje y enviar estos datos al cliente FSQL para ser procesados.

Siguiendo con el ejemplo de síntoma, se requiere verificar si se ha escogido o no un síntoma, un bloque de código dentro de este componente es el siguiente

```
if (($_POST['DOLOR'])== "ON"){
    echo 'Tengo dolor :)';
    $_SESSION['SINTOMAS'][$i]=array( 'DESCRIPCION' =>
$_POST['DOLOR_DESCRIPCION'],'LOCALIZACION' =>
$_POST['DOLOR_LOCALIZACION'],'PREGUNTA1' => NULL,'PREGUNTA2' => NULL,'NIVEL' =>
$_POST['DOLOR_nivel'],'DURACION' => $_POST['DOLOR_duracion'],'INICIO' =>
$_POST['DOLOR_inicio'], 'REPETICION' => $_POST['DOLOR_repeticion']);
    $i++;
}
```

Código 15. Bloque de código donde se procesa la información del formulario de síntoma

Donde la variable $_SESSION[‘SINTOMAS’] guarda un arreglo con los valores de cada síntoma seleccionado. Si el paciente no selecciona ningún síntoma entonces al paciente se le asocia un registro denomina *sinsintoma*.

Ya definido cómo será el procesamiento de las reglas y formularios, pasamos al último punto dentro de la arquitectura, el cliente Web.

3.3.5 Cliente Web

Este módulo se encarga de recoger los datos relacionados con el proceso de triaje. El reto presentado en esta fase fue definir cuál era la mejor interfaz para poder capturar la esencia del escenario difuso, sin embargo el objetivo fue alcanzado utilizando los *sliders* de la librería del *framework* de javascript *dojo*. Las vistas fueron divididas en 4 partes, la primera es el *header*, la segunda menú, la tercera centro y la última *footer*, cada uno de esos elementos son considerados

partes de la página Web, aunque los formularios se encuentran ubicados en una carpeta denominada formularios.

3.3.5.1 Header

El header es utilizado con dos objetivos, el primero unificar la vista de todas las páginas dentro del sistema triaje, el segundo es compartir los *scripts* necesarios para el despliegue de la aplicación dentro de todas las páginas. La estructura del *header* es la siguiente:

```
   ...
<head>
      <script type="text/javascript" src="js/dojo/dojo.xd.js" djConfig="parOnLoad: true"></script>
      <!-- sliders para sintomas -->
      <!--
      En el siguiente bloque se configuran los sliders
      -->
      <?php if (strstr($_SERVER['PHP_SELF'],'sintomas')){?>

      <?php $nombre_sintoma= 'DOLOR'; $nivel['N_ETIQUETAS']=3; $nivel['ETIQUETAS']='"Poco
dolor','Dolor moderado','Dolor insorportable"'; $nivel['MINIMO']=1; $nivel['MAXIMO']=10; include
'partes/slider.php';?>
      ...
      <?php } ?>
     ...
      <link rel="stylesheet" href="js/dijit/themes/claro/claro.css"/>
      <link rel="stylesheet" href="css/sateh.css" />
      ...
      <title>SATEH - <?php echo $pagina; ?></title>
   </head>
   <body class="claro">
   <div id="contenido" >
      <!-- el div de contenido termina en el Footer -->
      <div class="encabezado">
         <img src="css/image001.png" class="encabezado-left" alt="Sociedad Venezolana de
Medicina de Emergencia y Desastre" onclick="window.location.href='http://www.svmed.org'"/>
         <img src="css/cisi.png" class="encabezado-right" alt="Centro de investación de sistemas de
información - UCV"/>
      Universidad Central de Venezuela<br/>
      Facultad de Ciencias<br/>
      Escuela de Computación<br/>
      Centro de Investigación de Sistemas de Información<br/>
      Sociedad Venezolana de medicina de Emergencia y Desastre<br/>

      </div>
      <div class="centro_y_footer">
      <?php if ($_SESSION['logeado'] == TRUE) { ?>
      <div class="bienvenido">
         <span>Actulamente atendiendo a:<?php echo $_SESSION['paciente'][1].'
'.$_SESSION['paciente'][2]; ?></span>
         Bienvenid@ <?php echo $_SESSION['medico'][1].' '.$_SESSION['medico'][2]; ?>
      </div>
      <?php } else { ?>
      <div class="bienvenido">
         Por favor ingrese al sistema.
```

```
</div>
<?php }?>
```

Código 16. Bloque de código del header

Dentro de la etiqueta <head> se encuentran todos los scripts y hojas de estilo que se utilizan dentro de la aplicación. Las páginas de síntomas y constantes utilizan unos scripts adicionales los cuales permiten utilizar los sliders, para reducir el *overhead* o sobrecarga de la página se utiliza un condicional que evalúa el nombre de la página actual. Cuando la evaluación da verdadera se procede a cargar los slider de la siguiente forma:

- Cargar la librería de dojo *js/dojo/dojo.xd.js.*
- Parametrizar los sliders según sea el caso.
- Incluir el bloque *partes/slider.php.*

Después de la sección de scripts continua el contenido de la página, la variable $pagina es utilizada para darle valor a la etiqueta <title> con lo que termina la sección de <head>. Los datos del encabezado se encuentran dentro de una etiqueta de división identificada con la clase "encabezado", ésta contiene información estática como es el texto de Universidad Central de Venezuela, las imágenes del centro de investigación, etc. Por último un div identificado por la clase "bienvenido" contiene un pequeño script php utilizado para colocar el nombre del personal médico/asistencial que hace uso del sistema. La **Figura 14** muestra la sección de *header*.

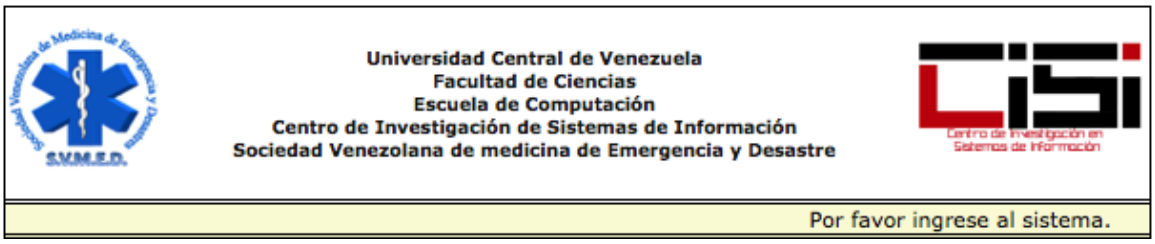

Figura 14. Sección del Header de SATEH

3.3.5.2 Menú

La sección de menú se encuentra justo debajo del *header*, ésta tiene dos estados dependiendo si el personal médico/asistencial ha iniciado sesión o no. La **Figura 15** ilustra el menú cuando no se ha iniciado sesión, la **Figura 16** ilustra el menú cuando se ha iniciado sesión. La siguiente es la estructura del bloque de menú:

```
<div class="cuerpo_menu">
<div class="menu">
```

```
   <ul>
      <?php
      if ($_SESSION['logeado'] == TRUE){
      ?>
      <a href="medico.php" >cambiar usuario</a>
      <?php
      }else {?>
      <a href="medico.php">entrar al sistema</a>
      <?php } ?>
   </ul>
   <ul>
      <a href="paciente_atendido.php" target="_blank">actualizar paciente</a>
   </ul>

   <ul>
   <a href="salir.php">salir</a>
   </ul>

</div>
</div>
```

Código 17. Bloque de código de menú

Figura 15. Estado del menú cuando no se ha iniciado sesión

Figura 16. Estado del menú cuando se ha iniciado sesión

Funcionalidades del menú

Cuando se ha iniciado una sesión el menú presenta 3 funcionalidades del sistema.

- CAMBIAR USUARIO: representa al caso de uso CD004, el cual cierra la sesión del actual usuario del sistema y carga la ventana de selección de persona.

- ACTUALIZAR PACIENTE: representa al caso de uso CD003, el cual toma un paciente al que previamente se le ha realizado un proceso de triaje y le otorga el estatus de atendido.

- SALIR: usado para terminar de usar el sistema, esta funcionalidad limpia todas las variables del sistema, cierra tanto la sesión actual del usuario como la del paciente al que se le está realizando el triaje y carga la pantalla de inicio.

Cuando no se ha iniciado una sesión la funcionalidad "ENTRAR AL SISTEMA" carga la pantalla de selección del personal médico/asistencial.

3.3.5.3 Centro

Es en esta sección donde se coloca realmente lo que se quiere presentar, por lo que el contenido es variable. Los formularios deben ser colocados dentro de esta sección, el siguiente es un ejemplo de una de las secciones centrales perteneciente a la página selección de paciente

```
include_once ('../Base de datos/consultas/lista_pacientes.php');
...
<div class="centro">
<form name="ingreso" action="../Modulo de Triaje/procesadores/ingreso_paciente.php"
method="POST">
   <span>Seleccione el nombre del paciente:<br/> <select name="Nombre">
<?php
   foreach  ($resulset as $paciente){ ?>
            <option>
            <?php echo $paciente['CODIGO'].' '.$paciente['NOMBRE'].' '.$paciente['APELLIDO'].'
'.$paciente['GENERO'].' '.$paciente['EDAD']; ?>
            </option>
   <?php } ?>
         </select>
   </span>
   <br/><br/>
   <input type="submit" value="Aceptar"/>
</form>
</div>
```

Código 18. Formulario de pacientes aún sin clasificar

Este formulario lee de la lista de pacientes que aún no se han clasificado por triaje y llena la lista con los datos necesarios de estos pacientes.

3.3.5.4 Footer

El *Footer* es la sección ubicada al final de una página. El contenido estático se encuentra dentro de una división identificada por la clase "pie_pagina", la **Figura 17** muestra la visualización del *Footer*. La estructura es la siguiente:

```
<div class="pie_pagina">
   Sistema Automatizado de Triaje para la Emergencia Hospitalaria<br/>
   Desarrollado por Jose Schmidt<br/>
</div>
<!-- el próximo </div> es el fin de contendio y footer -->
</div>
<!-- el próximo </div> es el fin de contendio -->
</div>
</body>
</html>
```

Código 19. Bloque de HTML correspondiente al *Footer*

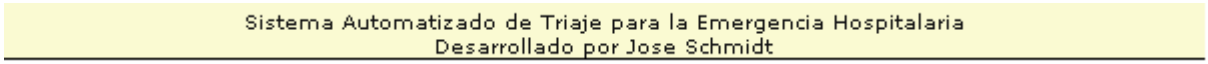

Figura 17. Imagen del *Footer*

3.3.5.5 Construcción de páginas

Utilizando los elementos antes mencionados la construcción de una página resulta simple. Basta con agregar un *include* del *Header*, menú, centro y *Footer*. El siguiente bloque muestra cómo está compuesta la pantalla de inicio.

```
<?php
$pagina='Inicio';
include_once ("partes/Header.php");
include_once ("partes/menu.php");
include_once ('partes/inicio.php');
include_once ('partes/Footer.php');
?>
```

Código 20. Bloque correspondiente a la página de inicio.

3.3.5.6 Sliders

Para facilitar los procesos de mantenimiento y extensión de las secciones que utilizan *slider*, se diseñaron dos estructuras de código PHP: una de las estructuras para los síntomas y otra para los signos y constantes vitales. Ambos estructuras fueron parametrizadas de la siguiente manera: Nombre del síntoma, Valor mínimo del *slider*, Valor máximo del *slider*, cantidad de etiquetas dentro del *slider*, valor de las etiquetas dentro del *slider*. El siguiente bloque muestra la estructura de un *slider*.

```
        dojo.require("dijit.form.Slider");
        dojo.require("dijit.form.HorizontalRule");
        dojo.addOnLoad(function() {
        <?php if($nombre_sintoma != 'TOS' AND $nombre_sintoma != 'DIARREA'){ ?>
            var horizontal_nivel = dojo.byId("<?php echo $nombre_sintoma.'_'; ?>nivel");
            var rulesNode_nivel = document.createElement('div');
            horizontal_nivel.appendChild(rulesNode_nivel);
            var sliderRules_nivel = new dijit.form.HorizontalRule({count:<?php echo
($nivel['MAXIMO']-$nivel['MINIMO']+1);
?>,style:"height:0.5em",container:"bottomDecoration"},rulesNode_nivel);

            var rulesNodeLabels_niveles = document.createElement('div');
            horizontal_nivel.appendChild(rulesNodeLabels_niveles);
            var sliderRulesLabels_nivel = new dijit.form.HorizontalRuleLabels({count:<?php echo
$nivel['N_ETIQUETAS'] ?>3,style:"height:1.2em;font-size:75%;color:gray;",labels:[<?php echo
$nivel['ETIQUETAS']; ?>]},rulesNodeLabels_niveles)
                var <?php echo $nombre_sintoma.'_'; ?>nivel = new dijit.form.HorizontalSlider({
                name: "<?php echo $nombre_sintoma.'_'; ?>nivel",
                value: <?php echo $nivel['MINIMO']; ?>,
```

```
            minimum: <?php echo $nivel['MINIMO']; ?>,
            maximum: <?php echo $nivel['MAXIMO']; ?>,
            discreteValues: <?php echo ($nivel['MAXIMO']-$nivel['MINIMO']+1); ?>,
            intermediateChanges: true,
            style: "height:30px;width:400px; margin-left: 30px; margin-top: 20px;",
            showButtons:false,
            onChange: function(value) {
                dojo.byId("<?php echo $nombre_sintoma.'_'; ?>nivelValue").value = value;
            }
        },
        "<?php echo $nombre_sintoma.'_'; ?>nivel");
    <?php } ?>
```

Código 21. Bloque de código para representar el nivel de algunos síntomas.

3.3.5.7 Ventanas

Las ventanas desarrolladas para la aplicación fueron las siguientes:

Inicio

Esta página no contiene requerimientos funcionales. Posee sólo un enlace para iniciar sesión dentro del sistema. La **Figura 18** muestra la página de inicio del sistema.

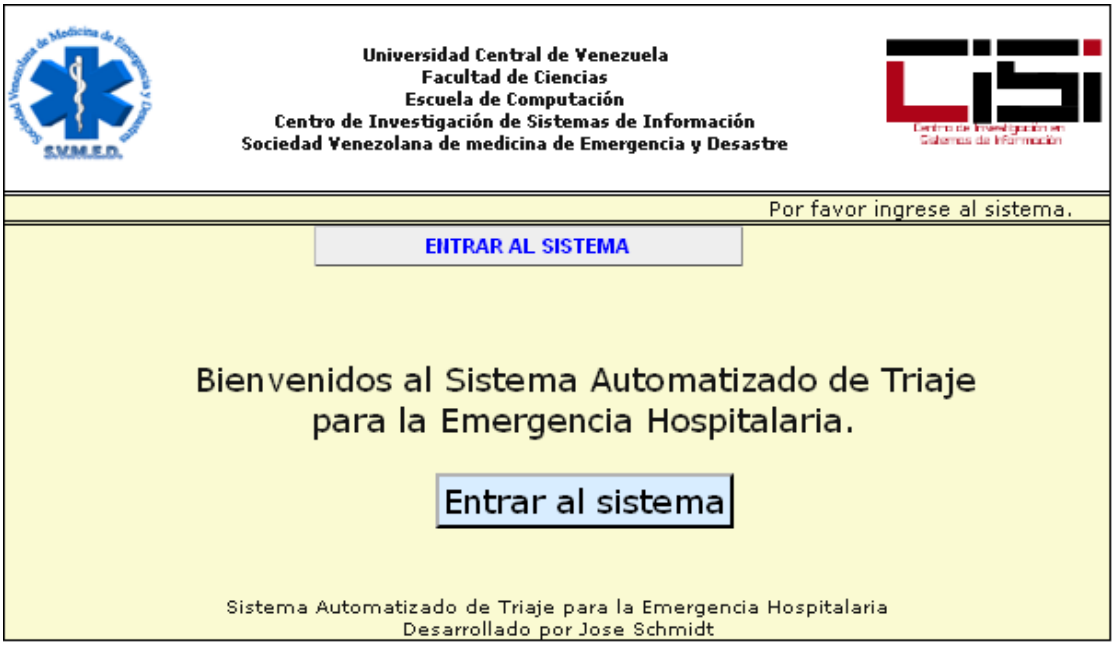

Figura 18. Página de inicio del sistema SATEH

Ingreso al sistema

En esta página una lista desplegable contiene los datos del personal médico/asistencial registrados en el sistema. Para poder continuar con la aplicación se debe seleccionar un usuario y presionar el botón aceptar. La variable de entorno $_SESSION['medico'] toma el valor seleccionado y esta se utiliza para desplegar el nombre del usuario en el *header*. La **Figura 19** muestra la página de ingreso del sistema.

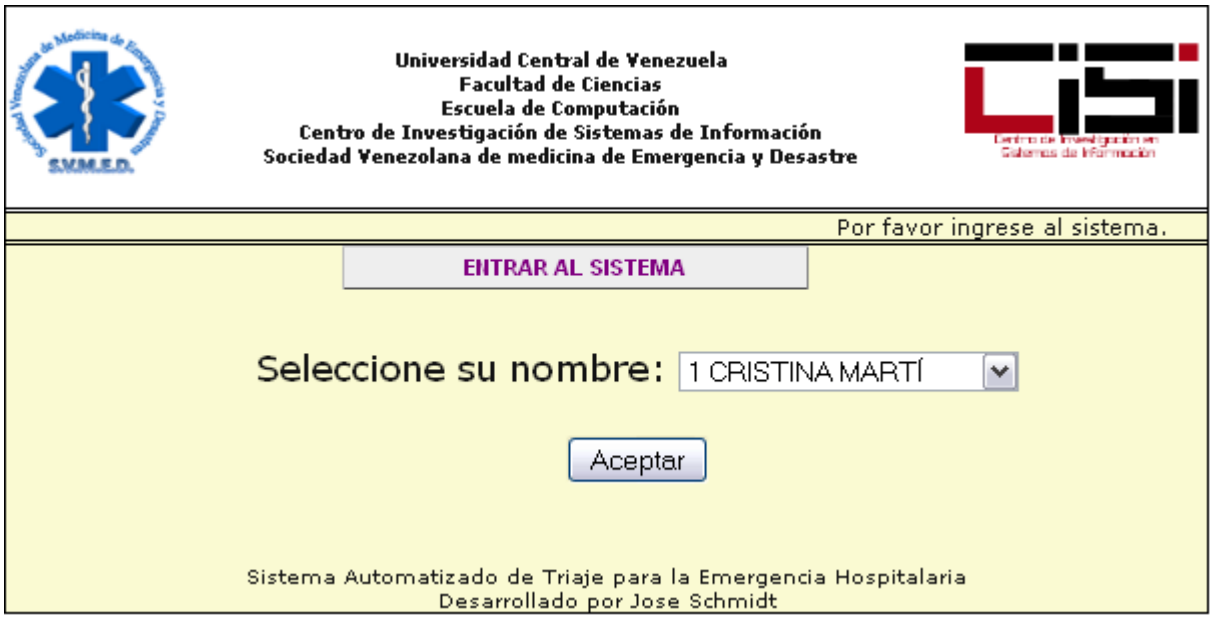

Figura 19. Página de inicio de sesión del sistema SATEH

Selección de paciente

Esta vista contiene dos funcionalidades, la primera permite tomar un paciente al que aún no se le ha aplicado triaje (denominado como pacientes en cola), la segunda permite seleccionar un paciente para realizar el proceso de reclasificación. Los datos del paciente son almacenados en la variable de sesión $_SESSION['paciente'] para que sus datos puedan ser desplegados dentro del Header. La **Figura 20** muestra la página para seleccionar un paciente.

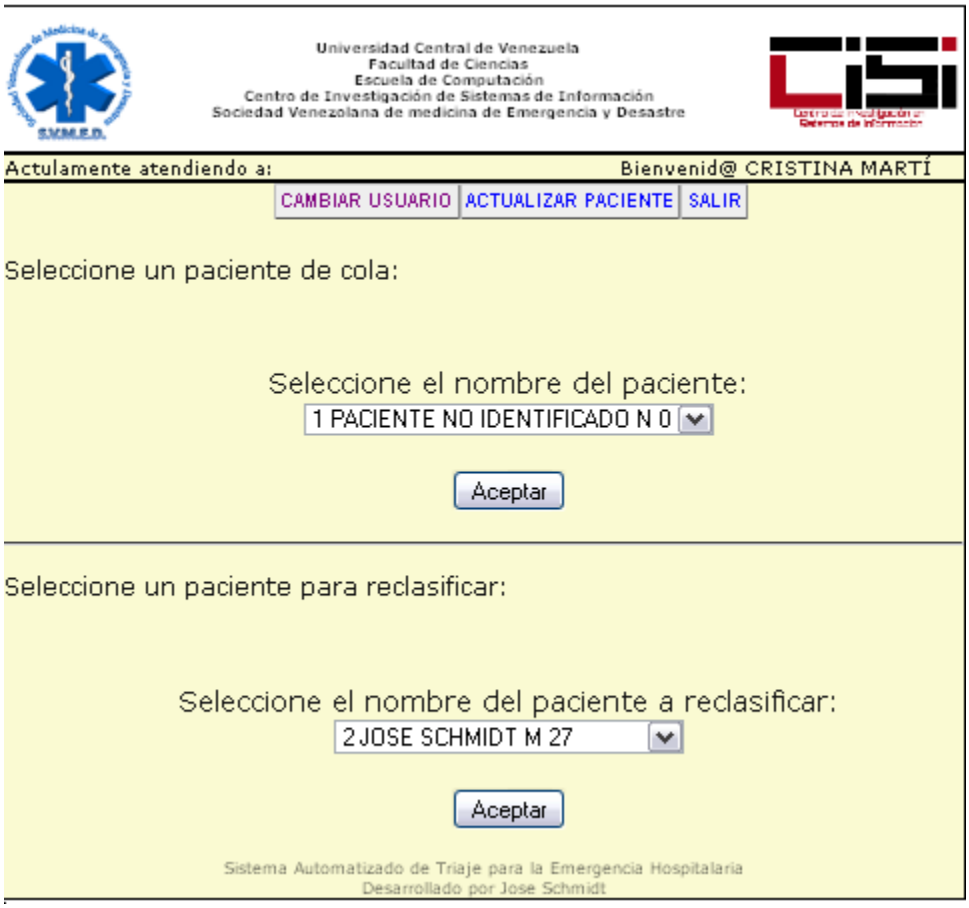

Figura 20. Página de selección de pacientes

Motivo de consulta

En esta página existen dos tipos de motivo de consulta separados por el grado de urgencia. Los motivos de urgencia más severos son representados con botones de *submit*, cuando alguno de estos motivos es utilizado el sistema automáticamente arroja un resultado de emergencia con prioridad tipo 1 indicando que el paciente debe ser atendido inmediatamente. El segundo conjunto de motivos de consulta se encuentran representados dentro de una lista de selección, una vez seleccionado un motivo de consulta se debe presionar el botón continuar. La **Figura 21** muestra la página para seleccionar el motivo de consulta.

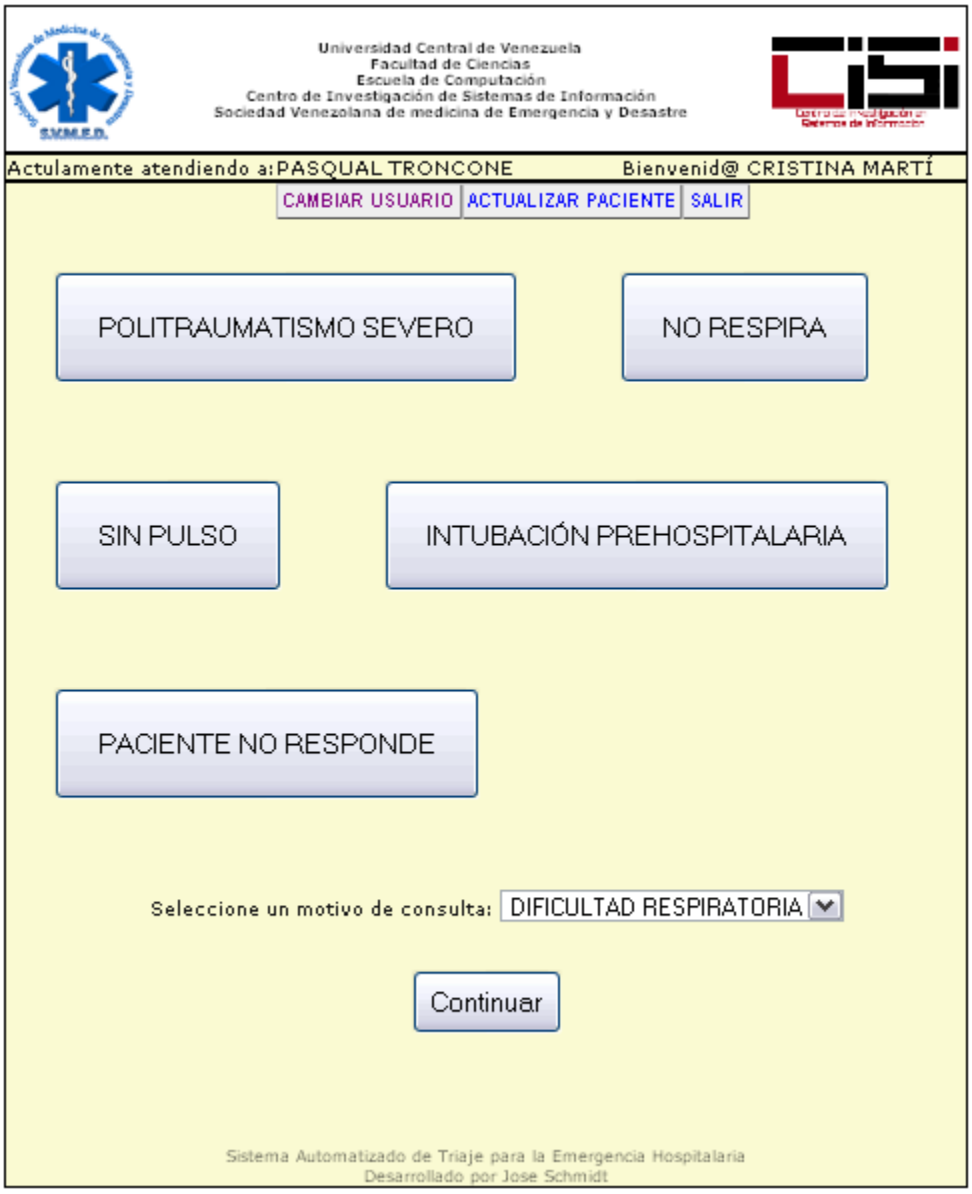

Figura 21. Página de selección del motivo de consulta

Síntomas

En esta pantalla se capturan los síntomas que presenta el paciente. El sistema permite que el paciente indique ninguno o todos los síntomas, sin embargo es necesario habilitar el *check* en los síntomas deseados. En caso de que se ingresen valores a algunos de los síntomas y no se haya habilitado el *check* estos valores no serán tomados en cuenta dentro del proceso de triaje. Una vez seleccionado todos los valores se continua el proceso de triaje utilizando el botón continuar. La **Figura 22** muestra la página para seleccionar síntomas.

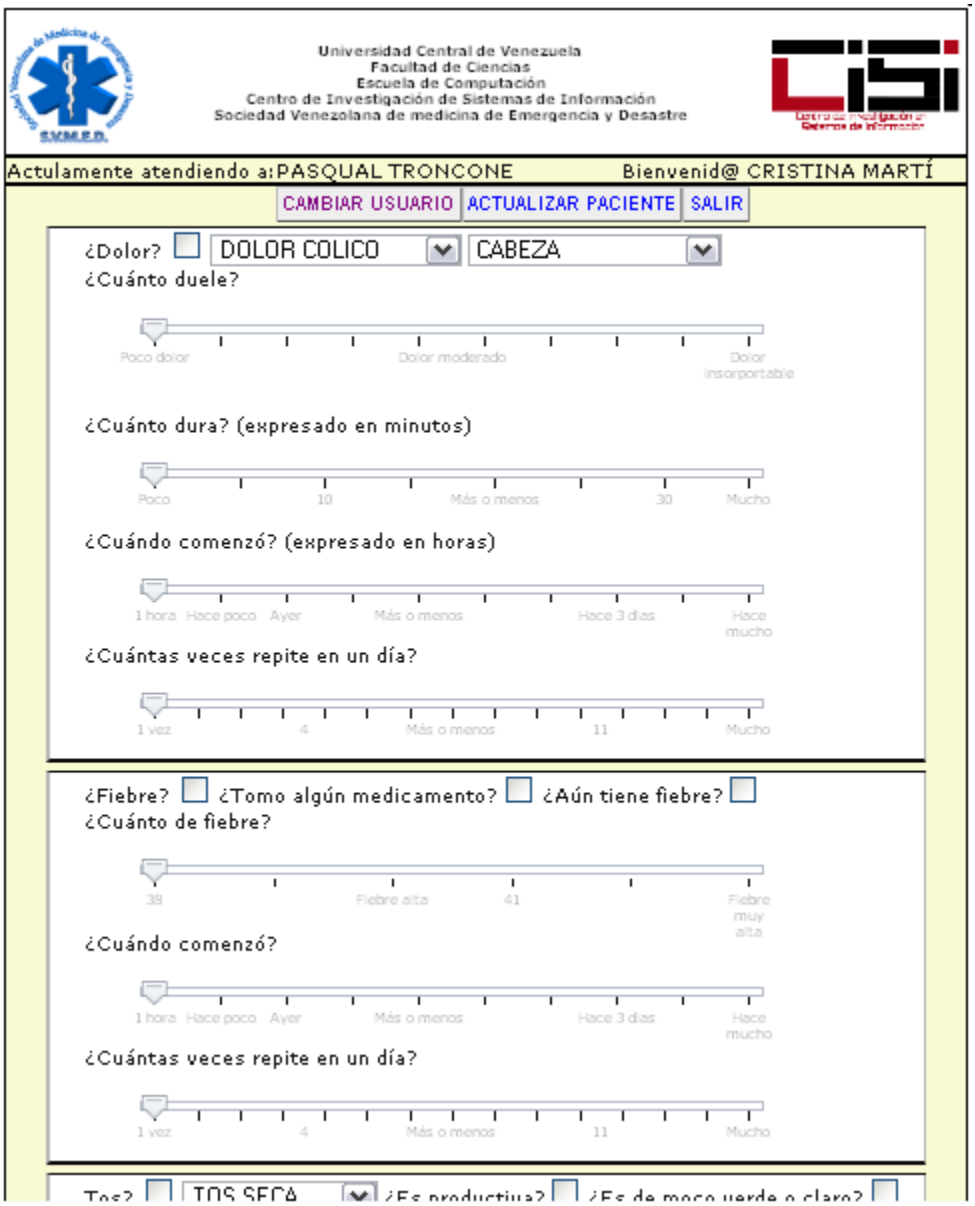

Figura 22. Página de selección de síntomas

Antecedentes

En esta página se captura el antecedente más resaltante del paciente, si es diabético, hipertenso, etc. Seleccionado el antecedente se utiliza el botón aceptar. La **Figura 23** muestra la página para seleccionar antecedentes.

Figura 23. Página para seleccionar antecedente

Signos y Constantes Vitales

Ésta es la última pantalla del proceso de captura de triaje. En esta se miden los signos y constantes vitales que posee actualmente el paciente. Por defecto, los valores se encuentran dentro del rango normal. Una vez seleccionado todos los valores se procede a ejecutar el algoritmo de clasificación de triaje oprimiendo el botón terminar. La **Figura 24** muestra la página para capturar los signos y constantes vitales.

Clasificación

Al terminar el proceso de captura de datos, el sistema genera una pantalla en la que se indica los posibles diagnósticos y grado de urgencia del paciente. Adicionalmente, se anexa un cuadro resumen con todos los valores capturados en el proceso. Para realizar el siguiente proceso de triaje se oprime el botón "Próximo Triaje". La **Figura 25** muestra la página que contiene los resultados del triaje.

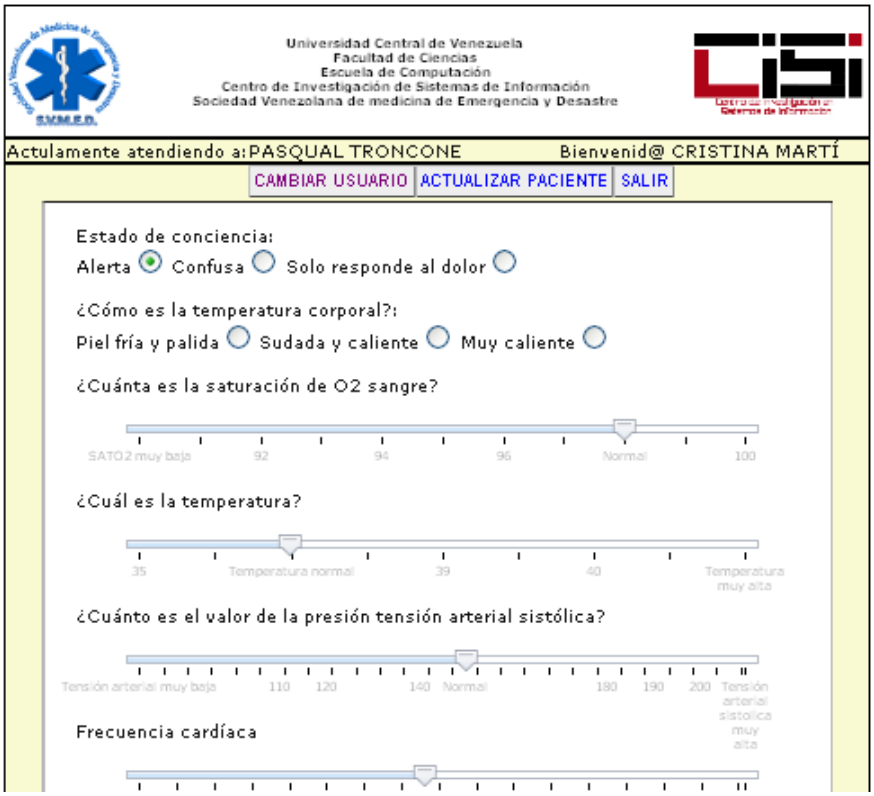

Figura 24. Pantalla de captura para signos y constantes vitales

Figura 25. Página que muestra los resultados del proceso de triaje

3.4 Fase de Transición

El sistema fue sometido a una fase de prueba, de la que se han extraido 4 para presentar en este documento, sin embargo los sistemas de triaje y en general los sistemas CDSS requieren de varios meses de prueba e incluso varios trabajos posteriores que determinan la exactitud de estos sistemas, en el caso particular de SATEH estas pruebas ayudaron a mejorar algunas reglas difusas y la forma de almacenar los datos. Las pruebas fueron realizadas por el Dr. Victor Rodriguez de forma remota, a continuación se presentan los resultados.

3.4.1 Pruebas de aceptación

CDP001

Caso de Prueba ID:	CDP001
Caso de Uso ID:	CDU001
Nombre del caso de uso:	Ingresar Episodio de Triaje
Descripción	Un paciente desconocido llega a la emergencia por ambulancia sin respirar
Precondiciones:	Un personal médico/asistencial ingresa al sistema, selecciona paciente desconocido y motivo de consulta no respira.
Resultado Esperado:	El paciente debe ser atendido con un grado de urgencia nivel 1, inmediatamente.
Evaluación de la prueba	Prueba superada con éxito.

CDP002

Caso de Prueba ID:	CDP002
Caso de Uso ID:	CDU002
Nombre del caso de uso:	Reclasificar Triaje
Descripción	Un paciente ya ha sido clasificado por el proceso de triaje con nivel 3 o más y ha culminado el tiempo de espera. El personal médico/asistencial debe clasificar al paciente nuevamente.
Precondiciones:	Un personal médico/asistencial ingresa al sistema y existe por lo menos un paciente que ya ha sido clasificado, que ya culminó su tiempo de espera y aún no ha sido atendido, se selecciona el paciente de la lista de reclasificación y se vuelve a ejecutar el proceso de triaje.
Resultado Esperado:	El paciente tiene un nuevo episodio de triaje que puede tener un nivel igual o distinto al anterior.
Evaluación de la prueba	Prueba superada con éxito.

CDP003

Caso de Prueba ID:	CDP003
Caso de Uso ID:	CDU003
Nombre del caso de uso:	Atender paciente
Descripción	Un paciente ya ha sido clasificado por el proceso de triaje sale de la cola y pasa al servicio de emergencia.
Precondiciones:	Un personal médico/asistencial ingresa al sistema y existe por lo menos un paciente que ya ha sido clasificado, se debe utilizar la opción del menú actualizar paciente y este ya no debe estar en la lista de reclasificación.
Resultado Esperado:	El paciente ya no debe estar en la lista de reclasificación.
Evaluación de la prueba	Prueba superada con éxito.

CDP004

Caso de Prueba ID:	CDP004
Caso de Uso ID:	CDU004
Nombre del caso de uso:	Cambiar de personal
Descripción	Cuando un personal médico/asistencial empieza su trabajo dentro del proceso de triaje y existe otro usuario iniciado en el sistema este debe cambiar para ingresar con su usuario y el sistema debe reconocer dicho cambio.
Precondiciones:	Un personal médico/asistencial ha ingresado dentro del sistema y se desea realizar un cambio de personal, este selecciona del menú cambiar usuario y el sistema lo lleva a la pantalla de selección de personal.
Resultado Esperado:	El sistema modifico la variable de sesión del personal médico/asistencial
Evaluación de la prueba	Prueba superada con éxito.

Conclusiones

Se desarrolló el prototipo de sistema que ejecuta el proceso de clasificación de triaje. Actualmente el sistema está siendo evaluado por la Sociedad Venezolana de Medicina de Emergencia y Desastre.

Fue posible implementar el modelo conceptual de datos, debido a que sólo hace uso de los atributos difusos tipo 1,2 y 3 y estos están totalmente soportados dentro del servidor de FSQL. Los protocolos o categorías no pudieron ser representados como un atributo tipo 3, por lo que fueron construidas bajo reglas difusas que daban como resultado un grado de pertenencia a estas categorías.

Los esquemas de bases de datos que representan el modelo lógico difuso de triaje fueron implementados en modelos relacionales usando el motor FSQL Server para transformar estos en esquemas relacionales difusos.

El sistema se construyó utilizando el lenguaje de programación PHP5 y la tecnología de base de datos difusa FSQL Server, el cual realiza inferencias sobre los datos capturados y propone varias alternativas de clasificación según el protocolo de triaje. El sistema utiliza el ambiente de trabajo de Apache para la interacción con los usuarios, siendo este quien controla la cantidad de usuarios que pueden acceder al sistema. En cuanto al tamaño de la base de datos solo aplican las restricciones de la base de datos Oracle, tales como tamaño de los *datafiles*, etc.

Los hitos propuestos como alcanzables dentro de las diferentes fases de la metodología AUP junto con la minimización de entregables fueron claves para alcanzar el desarrollo del prototipo en el tiempo planificado.

El análisis de los requerimientos funcionales basado en el modelo fuzzy de triaje de la SVMED llevado a cabo en la fase de inicio se mantuvieron vigentes a lo largo del desarrollo.

La fase de Construcción del sistema se llevó a cabo en su totalidad, el cual puede ser accedido desde el url http://sateh.no-ip.org:8080/SATEH/Cliente_Web/

Se realizaron un pool de pruebas de aceptación las cuales fueron realizadas y avaladas por el Dr. Víctor Rodríguez, como se especificó en los roles y responsabilidades de los Stakeholders.

La función triangular funciono adecuadamente dentro de la aplicación, de forma que se obtuvieron los resultados esperados al aplicar las reglas.

Recomendaciones

- Es recomendable realizar un proceso de migración del servidor FSQL de Oracle 8i a una versión más reciente y accesible de forma gratuita como 10g xe o 11g xe, así como también certificar la propuesta de FSQL Postgres realizada por el equipo de investigación de chile.

- La inserción manual de datos difusos con sentencias *insert* tradicionales, es un proceso que está sujeto a errores, se sugiere agregar *constraint* de integridad al servidor FSQL para almacenar datos que puedan ser comparables dentro del FMB

- La inserción manual dentro del FMB es un proceso que está sujeto a errores, se recomienda crear *constraint* que no permitan almacenar valores dentro de este diccionario que estén construidos de manera errada.

- Como trabajos futuros que se realicen para continuar con este proyecto, se recomienda crear un módulo de gestión de conocimiento que le permita a los expertos poder agregar nuevas reglas de triaje separadas del código fuente.

- Se recomienda la creación de un módulo de atención de las emergencias que se alimente de la clasificación realizada por el sistema de triaje.

- Se recomienda la construcción de un módulo que permita agregar pacientes y personal médico/asistencial, de tal forma que en los centros asistenciales donde no existan sistemas administrativos o de admisión se pueda utilizar el sistema de triaje.

- Se necesita construir un mecanismo de control de recursos que permita eliminar los problemas de concurrencia acarreados por un sistema multiusuario, particularmente bloquear un paciente del sistema cuando éste sea atendido por un personal médico/asistencial.

Referencias

Ambler, S., & Kennaley, M. (2010). *SDLC 3.0 Beyond a Tacit Understanding of Agile*.

Bosc, P., Pivert, O. (1995). “SQLf: a relational database language for fuzzy querying”. *IEEE Transactions on Fuzzy Systems,* 2, pp.1-17.

Galindo, J. (2006). Introduction to Fuzzy Logic. In J. Galindo, A. Urrurtia, & M. Piattini, *Fuzzy Databases Modeling, Design and Implementation* (pp. 1-44). Hershey: Idea Group Publishing.

Galindo, J. (1999). Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión del Modelo y Adaptación de los SGBD Actuales. Tesis Doctoral. Dpto. de Ciencias de la Computación e Inteligencia Artificial. Universidad de Granada.

Garg, A. X., Adhikari, N. K., & McDonald, H. (2005). Effects of Computarized Clinical Decision Support Systems on Practitioner Performance and Patient Outcome: A Systematic Review. JAMA , 1223-1238.

Gómez, J. (2003). “Clasificación de pacientes en los servicios de urgencias y emergencias: hacia un modelo de triaje estructurado de urgencias y emergencias.” Revista Emergencias. 15:165-74. España.

Gómez, J. (2009). Retrieved Septiembre 09, 2010, from Universidad de Antioquia | Salud Pública: http://guajiros.udea.edu.co/fnsp/congresosp6/memorias6/Miercoles%2010/Tarde/Organizacion%20y%20Gestion%20de%20Urgencias_Medellin.pdf

Guterman JJ, Mankovich NJ, Hiller J (2003). “Assessing the effectiveness of a computer-based decision support system for emergency department triage”, Engineering in Medicine and Biology Society, 1993. Proceedings of the 15th Annual International Conference of the IEEE pp. 592 – 593.

Holmstrom, Inger (2007). Decision aid software porgrams in telenursing: not used as intended? Experiences of Swedish telenurses. Nursing and Health Siciences, 9, pp. 23-28.

Iserson, K. V., & Moskop, J. C. (2007). Triage in Medicine, Part I: Concept, History, and Types. *Annals of Emergency Medicine , 49* (3), 275-281.

Larrey, D. J. (1987). *Memoirs of Military Surgery, and Campaigns of the French Armies* (Vol. 2). Baltimore: Classics of Medicine Library.

Referencias

Lerdorf, R., Hannes, M., & Olson, P. (2010). PHP: Que es PHP? - Manual. Retrieved Agosto 28, 2010, from PHP: Hypertext Preprocessor: http://www.php.net/manual/es/intro-whatis.php

Marklund, B (2000). “Symptom, Rad, Atgard”. Vanersborg: Vardutveckling AB.

Medina, J. (1994). GEFRED. A Generalized Model of Fuzzy Relational Data Bases. Ver. 1.1. Information Sciences (pp. 87-109).

Oliva Moreno, R. F. (2003). *VFSQLDoc.pdf.* Retrieved 08 15, 2010, from FSQL (Fuzzy SQL) a Fuzzy Query Language: http://www.lcc.uma.es/~ppgg/PFC/VisualFSQL/VFSQLDoc.pdf

Rangel, W., Matteo, A. (2010). Modelo conceptual de datos difusos de triaje para emergencia hopitalaria representado con FuzzyEER. *Enl@ce: Revista Venezolana de Información, Tecnología y Conocimiento, 7 (2), 83-98*.

Rodriguez, V. (2007). *Diseño de un sistema de Triaje Hospitalario*. Recuperado el 07 de Octubre del 2007 del sitio web de la Sociedad Venezolana de Emergencia y Desastre:http://www.svmed.org.ve/svmed/images/svmed/triagehospitalario.ppt

SVMED. (2010). Historia Prehospitalaria. Retrieved Julio 26, 2010, from Sociedad Venezolana Medicina Emergencia y Desastres: http://svmed.org/index.php?option=com_content &view=article&id=7&Itemid=12&lang=es

Urrutia, A. (2003) *Definición de un Modelo Conceptual para Bases de Datos Difusas*. Tesis Doctoral, Universidad de Castilla-La Mancha, Ciudad Real, España.

Weitzenfeld, A. (2004). *Ingenieria del Software orientada a objetos con UML, Java e Internet*. Thomson.

Zadeh, L.A. (1965). Fuzzy Sets. *Information and Control*, 338-353.