Obtención y consolidación de datos para Unidades Ejecutoras de ASSE

Proyecto de Grado - Informe Final

Estudiantes:
Alejandro Álvarez
Agustín Gründel
Carolina Irigoyen
Rosina Piccardo

Tutor:
Ariel Sabiguero

Responsable por parte de ASSE:
Federico Ramos

Tribunal:
Lorena Etcheverry
Antonio López
Daniel Meerhoff

Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera Ingeniería en Computación

Montevideo, 2013

RESUMEN

Este trabajo corresponde al Proyecto de Grado de la carrera Ingeniería en Computación de la UdelaR y fue posible gracias a la cooperación entre la Facultad de Ingeniería y la Administración de los Servicios de Salud del Estado (ASSE). ASSE necesita formalizar herramientas informáticas que sistematicen la captura de datos para la generación de información y así disminuir el trabajo manual.

En la actualidad la gestión de los egresos hospitalarios se origina cuando el médico completa la hoja de cierre de historia clínica al egreso de un paciente, ya sea por alta, fallecimiento o traslado. A continuación le corresponde a los encargados de registros médicos de cada Unidad Ejecutora codificar los diagnósticos y procedimientos del mencionado formulario (según la clasificación CIE10 y CIE9 MC respectivamente[1]), y luego pasar los datos a una planilla de cálculo que es enviada al Departamento de Gestión de la Información de ASSE. Todos los meses llegan (vía email) al Departamento de Gestión de la Información de ASSE más de 60 planillas de cálculo correspondientes a los Egresos del mes de cada Unidad Ejecutora. Éstas contienen entre 15 y 1000 filas (donde cada fila corresponde a un egreso) y cuenta a su vez con casi medio centenar de columnas de datos.

Los objetivos principales del proyecto son: Investigar y elegir un framework web open source que facilite la realización de formularios web para la recolección de datos obtenidos en distintas Unidades Ejecutoras y desarrollar una herramienta web utilizando el framework elegido que facilite la recolección y el procesamiento de dichos datos ayudando a que la calidad de los mismos sea la mejor posible.

Para el cumplimiento del primer objetivo se estudió el estado del arte de herramientas web. En particular se estudiaron herramientas de prototipado rápido similares a una que se tomó como punto de comparación y que contaran con interfaz web.

Luego de seleccionada la herramienta, se desarrolló una aplicación llamada Sistema de Codificación de Egresos (SCE) que permite informatizar la recolección de los datos correspondiente a los egresos de las diferentes Unidades Ejecutoras. En el proceso de desarrollo de la aplicación se subsanaron diversas carencias detectadas en la metodología que hasta entonces aplicaba ASSE, mejorando significativamente la calidad de los datos. La información recolectada por el SCE fue posteriormente integrada a un Data Warehouse automatizando así la consolidación de los egresos y permitiendo de esta manera la visualización de la información mediante reportes y vistas de análisis.

Finalizado el desarrollo del SCE, ASSE seleccionó cuatro Unidades Ejecutoras de Montevideo, Maldonado, Salto y Paysandú que trabajaron en un plan piloto. Tras comprobar la efectividad del sistema se sumaron paulatinamente más Unidades Ejecutoras con la intención de llegar a su total implantación en todo el país.

Cabe destacar que a pesar de la dimensión y complejidad de una institución pública como ASSE, el proyecto logró de todas formas dejar un sistema en producción que facilitó el proceso de trabajo en un área específica, así como le pudo brindar a la organización elementos que le ayuden a la micro y macro gestión de sus Unidades Ejecutoras.

1 CIE-10: Clasificación Internacional de Enfermedades décima revisión OPS/OMS y CIE-9 MC: Clasificación Internacional de Enfermedades novena revisión Modificación Clínica OPS/OMS

TABLA DE CONTENIDO

1 INTRODUCCIÓN

El presente trabajo corresponde al Proyecto de Grado de la carrera Ingeniería en Computación de la Universidad de la República (UdelaR). Este fue posible gracias a la cooperación entre la Facultad de Ingeniería y la Administración de los Servicios de Salud del Estado (de aquí en adelante ASSE), encontrándose enmarcado dentro de las actividades de investigación y desarrollo llevadas a cabo por este último organismo en su afán de apoyar las reformas sanitarias a nivel nacional con sistemas de información confiables y de valor agregado.

ASSE es un organismo descentralizado que cuenta con una Red de Atención Integral a la Salud constituida por Unidades Ejecutoras (U.E).

Al momento de comenzar el proyecto, ASSE cuenta con 47 U.E en el interior del país, de las cuales 18 son Centros Departamentales y el resto Centros Auxiliares, así como con 18 Redes de Atención Primaria de las que dependen unas 650 policlínicas. En Montevideo se cuenta con 13 U.E. de referencia nacional como ser H. Maciel, H. Pasteur, Pereira Rossell, H. de Ojos “José Martí” entre otros, así como la Red de Atención del Primer Nivel, integrada por 12 Centros de Salud y una red de más de 100 policlínicas.

ASSE es el Prestador Integral de Salud del país con mayor cantidad de usuarios: 1.242.067 a Abril de 2013.

ASSE establece como misión ser el prestador público de referencia, basado en la Atención Primaria, con equidad, eficiencia y calidad, y con capacidad para responder a las necesidades de su población usuaria, en un marco de políticas de equidad social. Por otro lado su visión es liderar el proceso de atención a la salud de los habitantes del país, contribuyendo de ese modo a la calidad de vida de la población, poniendo énfasis en la ecuación y promoción, prevención, diagnóstico precoz, recuperación y rehabilitación.

Para administrar la innumerable cantidad de datos que arrojan los servicios de ASSE, el organismo cuenta con la Dirección de Sistemas de Información y dentro de la misma con el Departamento de Gestión de la Información responsable, entre otras áreas, de Egresos Hospitalarios – Morbimortalidad. Dicha área se encarga de recibir, revisar y analizar los datos de los egresos hospitalarios para generar información de valor para la institución.

1.1 MOTIVACIÓN

ASSE necesita buscar herramientas informáticas que ayuden a sistematizar la captura de datos para la generación de información así como disminuir el trabajo manual.

En la actualidad la gestión de los egresos hospitalarios se origina cuando el médico completa la hoja de cierre de historia clínica (Ver Anexo 1 - Hoja de Cierre de Historia Clínica) al momento del egreso de un paciente, ya sea este producido por alta, fallecimiento o el traslado a otro Hospital o Institución de asistencia para completar su internación. En el siguiente paso le corresponde a los encargados de registros médicos de cada Unidad Ejecutora codificar los diagnósticos y procedimientos del mencionado

formulario (según la clasificación CIE10 y CIE9 respectivamente), para más adelante pasar todos los datos a una planilla de cálculo la cual es enviada al Departamento de Gestión de la Información de ASSE.

Es preciso tener en cuenta que quienes completan estos datos en ocasiones no saben cómo usar las herramientas informáticas pues sus funciones no son de digitadores sino de codificadores de registros médicos.

Todos los meses llegan (vía email) al Departamento de Gestión de la Información de ASSE más de 60 planillas de cálculo correspondientes a los Egresos del mes de cada Unidad Ejecutora. Para dimensionar el volumen de información, basta decir que cada una de estas planillas puede contener entre 15 y 1000 filas; donde cada fila, que corresponde a un egreso, cuenta a su vez con casi medio centenar de columnas de datos.

El sistema utilizado hasta el día de hoy demuestra gran cantidad de debilidades que ponen en riesgo la calidad de la información, como por ejemplo:

- Los datos son cargados en una planilla en las cuales no hay validaciones; problemas de sintaxis e inconsistencia no son detectados, por ejemplo códigos que indican un diagnóstico de una afección femenina en un paciente masculino, etc.
- Muchas veces se deben descartar egresos ya que traen información inconsistente que no tiene sentido (debido a que contienen errores)
- Pueden existir datos repetidos que pasan inadvertidos dado la incomodidad del usuario para visualizar las filas en la planilla de cálculo y/o idem al problema detallado en primer punto.
- Se ha comprobado la dificultad al realizar búsquedas dado que estas son largas y tediosas.
- Como los datos son completados en distintas Unidades Ejecutoras y enviados vía email en una planilla de cálculo, lleva mucho tiempo la recolección y consolidación de los mismos con todos los riesgos que conlleva el manejo de tal cantidad de archivos separados.
- El equipo de Gestión de la Información obtiene datos estadísticos a partir de todas las planillas recibidas. Actualmente se procesan las planillas de forma prácticamente manual y luego se alimenta el programa SPSS2 [1] para el análisis.

Como otra motivación se destaca la necesidad de ASSE de desarrollar con software libre. En Uruguay a partir del 2011 se estudia la "Ley de Software Libre y Formatos Abiertos en el Estado" [2] la que recibió media sanción en la Cámara de Diputados el 19 de Diciembre de 2012. En el artículo 2º del proyecto de ley se menciona: "En las instituciones y dependencias del Estado ... cuando se contraten licencias de software se dará preferencia a licenciamientos de software libre. En caso de que se opte por software privativo se deberá fundamentar la razón basada en aspectos técnicos que no puedan ser resueltos con software libre."

Según el artículo 5º se define Software Libre como:

"El software libre es el que está licenciado de forma que cumpla las siguientes condiciones:

1. Pueda ser usado para cualquier propósito.
2. Se tiene acceso a su código fuente de forma que puede ser estudiado y cambiado para adaptarlo a las necesidades.
3. Pueda ser copiado y distribuido.
4. Sea posible la mejora del programa y la liberación de dichas mejoras a la ciudadanía."

1.2 OBJETIVOS

Investigar y elegir un framework web open source que facilite la realización de formularios web para la recolección de datos obtenidos en distintas Unidades Ejecutoras. Debe cumplir ser multiplataforma, tener una estética atractiva y moderna, poder utilizar autenticación con LDAP, generar formularios web de forma ágil, prototipado rápido y vitalidad adecuada.

Desarrollar una herramienta web sobre el framework elegido que ayude a:

- Facilitar la recolección de los datos en las diferentes Unidades Ejecutoras.
- Ayudar a que la calidad de los datos sea la mejor posible.
- Facilitar la consolidación de la información enviada por distintas Unidades Ejecutoras.
- Facilitar el procesamiento de los datos recolectados.

1.3 RESULTADOS ESPERADOS

- Elección de una herramienta que facilite la realización de formularios web. Para esto se deberá hacer un estudio del estado del arte de las herramientas existentes.
- Implementación de prototipo para el área Egresos Hospitalarios utilizando la herramienta seleccionada. Se deberá seguir una metodología de desarrollo y prototipado rápido con la intención de validar continuamente el mismo.
- Implantación del prototipo en un subconjunto de Unidades Ejecutoras.
- Incorporación de los datos recolectados por el prototipo a la herramienta de Datawarehousing existente en la organización.

1.4 ORGANIZACIÓN DEL DOCUMENTO

El presente informe cuenta con seis capítulos cuyo contenido se esboza a continuación.

El primer capítulo contiene una introducción inicial, la motivación y las características del problema a resolver así como los objetivos del proyecto y los resultados esperados.

A continuación, en el segundo capítulo se detalla el estado del arte de herramientas o frameworks web que faciliten la realización de formularios web para la recolección y consolidación de datos y se definen conceptos básicos necesarios para el análisis de las mismas.

En el capítulo tercero se realiza el análisis y diseño del producto desarrollado. Se plantea la arquitectura del sistema, requerimientos funcionales y no funcionales. También se detallan las decisiones de diseño tomadas así como el modelo de datos.

El Capítulo 4 contiene la especificación del producto implementado detallando el entorno de desarrollo, funcionalidades y el plan de pruebas realizado. Este capítulo también incluye las dificultades técnicas encontradas durante el proyecto y las validaciones con el cliente.

En el comienzo del Capítulo 5 se hace una introducción básica de conceptos y definición de data warehouse Seguidamente se describe el requerimiento relevado y su correspondiente implementación.

Finalmente en el Capítulo 6 se describen las conclusiones generales y resultados obtenidos en el proyecto, una sección para la gestión del proyecto y se mencionan las posibles líneas de trabajo a futuro.

En forma adicional se entregan los siguientes anexos:

Anexo 1: Hoja de Cierre de Historia Clínica

Anexo 2: Funcionalidades del Sistema de Codificación de Egresos

Anexo 3: Tablas de la base de datos

Anexo 4: Tabla de la base de datos códigos

Anexo 5: Detalle técnico de las interfaces

Anexo 6: Procedimiento de instalación

2 ESTADO DEL ARTE DE HERRAMIENTAS IDE/WEB

En esta sección se busca evaluar el estado del arte de herramientas o frameworks web que faciliten la recolección y consolidación de los datos en las distintas Unidades Ejecutoras.

La selección de las mismas para el análisis se hizo teniendo en cuenta varios puntos que se exigen como necesidades específicas de ASSE.

Se buscaron herramientas o frameworks web que puedan cumplir con la mayor cantidad posible de las siguientes características requeridas:

- Software libre: ASSE promueve este tipo de software en todas sus áreas de trabajo.
- Multiplataforma: El Sistema Operativo debería ser independiente de la plataforma, pero deberá tener soporte específico para Linux, ya que éste es el adoptado por el organismo.
- Estética (look and feel) atractiva y moderna: Si bien este punto puede llegar a ser subjetivo, se quiere que la herramienta posea características de un framework web de hoy en día, como por ejemplo el uso de AJAX que le brinden buena interactividad, velocidad y usabilidad.
- Consumir Web Services: Si bien este punto es relevante debido a la arquitectura de los distintos Sistemas que mantiene ASSE y sus necesidades de integración entre los mismos, cualquier framework web actual posee mecanismos para consumir un Web Service.
- Autenticación con LDAP: ASSE mantiene actualmente un LDAP para guardar la información de autenticación (usuario y password) que se debe poder consultar.
- Generar formularios para ABM's de forma ágil: Se pretende poder realizar formularios de forma rápida, intentando ahorrar líneas de código y/o configuraciones excesivas.
- Usuarios no expertos: Es deseable que la herramienta puede ser utilizada (en alguna medida) por usuarios sin conocimientos en el desarrollo de aplicaciones web.
- Prototipado rápido: El desarrollo rápido de aplicaciones (RAD) fomenta el prototipado en una etapa temprana del proceso, lo que permite lograr un aumento de la interactividad con el cliente, la usabilidad y la rapidez de ejecución.
- Vitalidad adecuada: El proyecto o herramienta elegida debería contar con buena documentación, que actualmente siga en desarrollo y una comunidad sustentable para cualquier consulta y/o aporte hacia el futuro.

2.1 CONCEPTOS RELACIONADOS

Antes de describir las herramientas que fueron evaluadas, definiremos algunos conceptos previos utilizados que ayudan al análisis posterior.

¿Qué es un framework?

Un framework que se podría traducir literalmente del inglés como “marco de trabajo” es en términos generales un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un problema en particular y que sirve como referencia para enfrentar problemas de similares características.

¿Qué es un framework web?

En el área del desarrollo de software y más precisamente en aplicaciones web, el término framework se asocia con una plataforma de software diseñada para apoyar la construcción de sitios web dinámicos, aplicaciones web y/o servicios web. El objetivo principal de usar un framework es alivianar la carga de trabajo en aquellos aspectos comunes a las aplicaciones web que ya han sido resueltos. Muchos de ellos proveen por ejemplo bibliotecas para el acceso a bases de datos, gestión de sesiones de usuarios, entre otros, promoviendo así la reutilización de código y por ende un ahorro sustantivo en el tiempo de desarrollo. La elección de un adecuado framework de aplicaciones web se debería hacer con precaución. Si la curva de aprendizaje del mismo es muy elevada, la ganancia de tiempo al utilizarlo se perdería en aprender a usar la herramienta y sería contraproducente.

¿Qué es RAD?

El desarrollo rápido de aplicaciones (Rapid Application Development) es un término originalmente utilizado para describir un proceso de desarrollo de software introducido por James Martin en 1991. Esta metodología implica un desarrollo iterativo con la construcción de prototipos. Actualmente el término es utilizado en un sentido más amplio abarcando una gran variedad de técnicas y herramientas para acelerar el desarrollo de aplicaciones.

¿Qué es Web service?

La World Wide Web Consortium lo define como “…un sistema de software diseñado para soportar interacción interoperable máquina a máquina sobre una red. Este tiene una interface descrita en un formato procesable por una máquina (específicamente WSDL). Otros sistemas interactúan con el servicios web en una manera prescrita por su descripción usando mensajes SOAP, típicamente enviados usando HTTP con una serialización XML en relación con otros estándares relacionados con la web”. [3]

¿Qué es LDAP?

Protocolo Ligero de Acceso a Directorios (Lightweight Directory Access Protocol) es un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

¿Qué es un Data Warehouse?

Un Data Warehouse es una base de datos corporativa de apoyo a la toma de decisiones que se caracteriza por integrar datos crudos de una o más fuentes distintas, depurando y almacenando la información necesaria de forma organizada para luego procesarla, permitiendo su análisis desde múltiples perspectivas y con grandes velocidades de respuesta. [4]

¿Qué es un sistema de Data Warehousing?

Un sistema de Data Warehousing es un sistema informático capaz de ofrecer información para toma de decisiones, y cuya pieza principal es un Data Warehouse. Una de las ventajas principales de estos sistemas es la generación **dinámica** de reportes. [5]

2.2 HERRAMIENTAS EVALUADAS

Antes de comenzar con el análisis se conocía la existencia de Oracle Application Express (Oracle Apex). Esta herramienta cumple con casi todos los objetivos necesarios para la elección de un framework web, excepto el primero y quizás más importante de ellos: no es software libre.

Ésta es una herramienta RAD para una base de datos Oracle. Usando solamente un navegador web se puede desarrollar y desplegar aplicaciones web profesionales, rápidas y seguras. La misma está orientada tanto a usuarios finales como a desarrolladores. Permite a usuarios con limitada o ninguna experiencia en programación APEX crear formularios rápidamente y diseñar aplicaciones centradas en una base de datos. Para los desarrolladores les permite centrarse en la lógica de negocio de su aplicación sin tener que preocuparse por los detalles relacionados a la interfaz de usuario.

A pesar de que se descarta esta opción de todas formas, al ser una completa herramienta sostenida por una gran organización como Oracle, la misma se puso como ejemplo o modelo a buscar en otras herramientas similares de software libre ya que la misma es un gestor de base de datos así como un generador de formularios.

2.2.1 NUBUILDER

La plataforma de software libre nuBuilder es un generador de aplicaciones web similar en concepto a Oracle Aplication Express y MS Access escrito sobre PHP/MySQL/JavaScript.

La principal característica que nuBuilder brinda al desarrollador es la capacidad de crear rápidamente una aplicación. Uno de los conceptos o filosofías que hace a nuBuilder una aplicación RAD es que sigue un modelo de navegación conocido como “Two Screen Model” o modelo de las dos pantallas. Este patrón se ha desarrollado en base al simple principio de que sólo hay dos cosas para hacer en una aplicación back-office: buscar un registro (Search Screen) y editar ese registro (Edit Screen).

La pantalla de búsqueda muestra una lista de registros diseñada para administrar fácilmente una gran cantidad de registros, permitiendo búsquedas y ordenaciones por columnas.

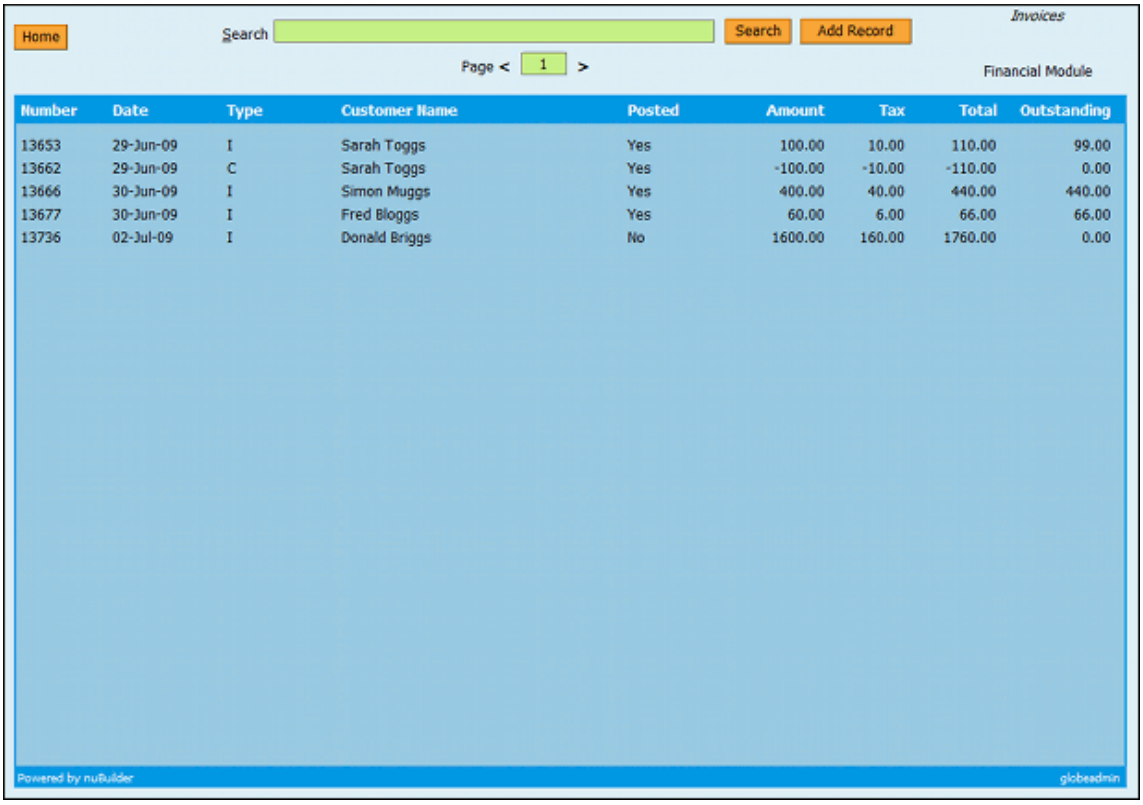


Figura 1: Ejemplo de Pantalla de Búsqueda (Search Screen)

Al seleccionar un registro en esta pantalla se puede ver la información del mismo así como editarla (Edit Screen).

En esta pantalla de edición la información de un registro particular es presentada y editada a través de campos usuales de un formulario (cajas de texto, combos, listas, botones, selectores de fecha, etc).

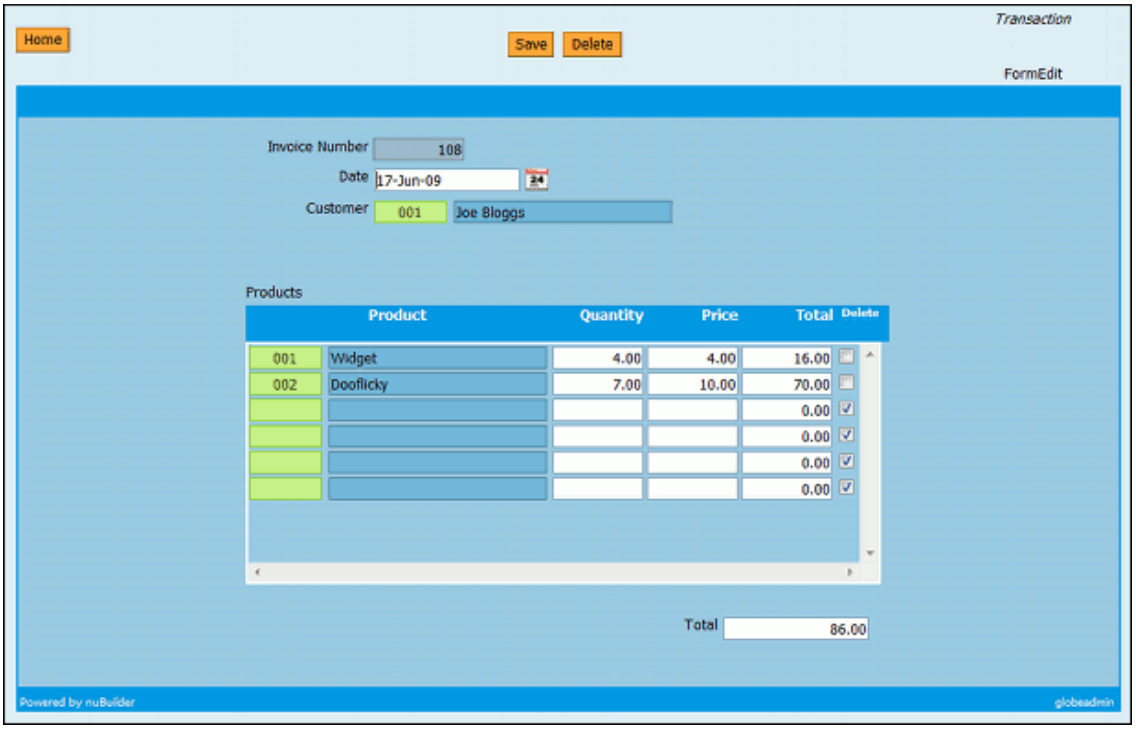


Figura 2: Ejemplo de Pantalla de Edición (Edit Screen)

VITALIDAD DEL PROYECTO AL MOMENTO DEL ANÁLISIS:

- Sitio: http://www.nubuilder.com/
- Tamaño comunidad integrantes del foro oficial: 24700 integrantes
- Foro Oficial: http://forums.nubuilder.com/
- Versión estable: 2.7 al momento del análisis
- Fecha Release : 5/6/2012
- Licencia : GPLv3

VENTAJAS Y DESVENTAJAS:

Si bien cumple con ser un software libre y multiplataforma (debido a que está concebido con PHP, MySQL y Javascript) el look and feel de las aplicaciones generadas (que puede llegar a customizarse en parte) luce un poco obsoleto para las aplicaciones web de hoy en día.

La ventaja de poder crear formularios para ABM's de forma rápida la cumple claramente, ya que está diseñado para agregar y editar registros de una base de datos ágilmente a través del modelo de dos pantallas. Este modelo simple parece un poco estricto y limitado para el tipo de aplicaciones que se necesitan en ASSE.

Si bien la consumición de Web Services y la autentitación LDAP no están integrados de forma nativa a la herramienta, las mismas no parecen ser un problema ya que podrían implementarse con bibliotecas PHP por fuera de la misma.

Debido a la arquitectura que maneja la herramienta, el prototipado con este software no parece ser una tarea fácil.

En cuanto a la vitalidad, si bien se cuenta con buena documentación en el sitio oficial de la herramienta así como con un foro de 24 mil integrantes (a Junio de 2012), el último release estable es de Setiembre del 2009.

2.2.2 XATAFACE

Xataface (anteriormente llamado Dataface) es una capa flexible y moldeable sobre MySQL. Genera automáticamente los formularios correspondientes, listas y menús para que un usuario pueda interactuar con la base de datos sin tener que saber nada de SQL. Se trata de un framework web que ofrece a los desarrolladores la flexibilidad para personalizar las características y el comportamiento de su aplicación a través de archivos de configuración (utilizando el sencillo método de archivos INI), plantillas y complementos. Una aplicación genérica, sin personalizaciones es completamente funcional, pero el desarrollador es libre de personalizar las cosas a su antojo. Xataface está orientado a desarrolladores web y administradores de bases de datos que les gustaría construir un front-end para su base de datos MySQL. Sin embargo, las aplicaciones resultantes están dirigidas a usuarios no técnicos.

Al contrario de la mayoría de los frameworks que requieren una cantidad sustancial de desarrollo antes de llegar a una aplicación usable, Xataface proporciona una aplicación totalmente funcional con tan sólo 4 líneas de código PHP. Como un framework de desarrollo, Xataface se parece más a Django, un framework Python para la construcción de aplicaciones basadas en datos. Como una aplicación, Xataface se parece más a Filemaker, una base de datos relacional muy popular que hace que sea fácil para el usuario final crear diseños y administrar sus datos.

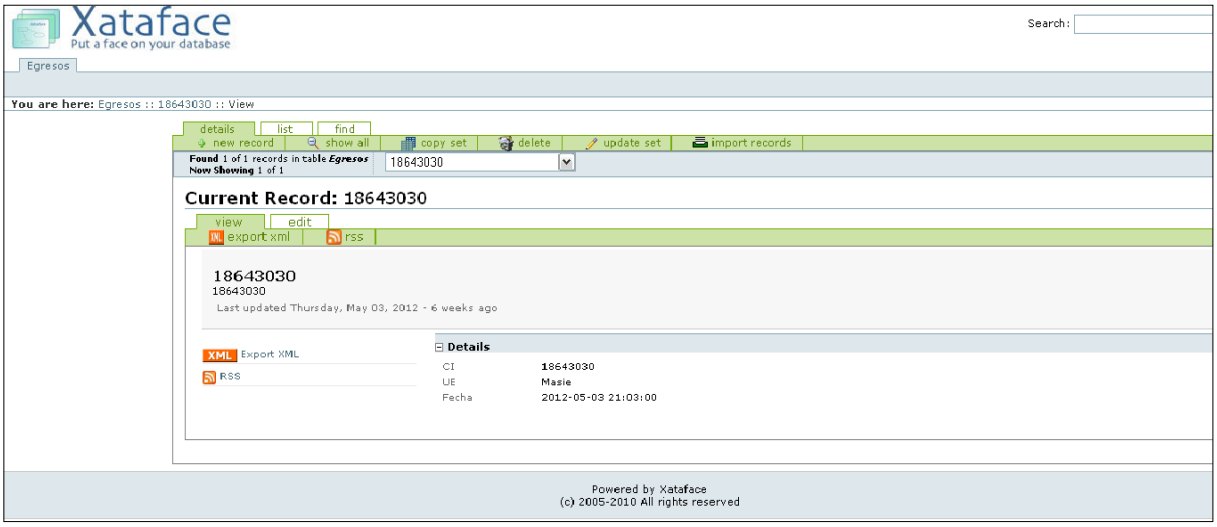


Figura 3: Ejemplo de aplicación creada para una base de datos con una única tabla llamada "Egresos"

El archivo de configuración .ini para esta aplicación contiene en su cuerpo:

```
[_database]
host = "localhost"
user = "root"
password = "1234"
name = "pruebaprot"
[_tables]
egresos = "Egresos"
```

El index.php contiene:

```
<?php
ini_set('memory_limit','32M');
require_once 'C:\AppServ\www\xataface-1.3.2\dataface-public-api.php';
df_init(__FILE__, 'http://localhost/xataface-1.3.2')->display();
?>
```

VITALIDAD DEL PROYECTO AL MOMENTO DEL ANÁLISIS:

- http://www.xataface.com/admin.php?-action=home
- Tamaño comunidad integrantes del foro oficial: 890 integrantes
- Foro Oficial: http://www.xataface.com/forum/index.php
- Versión estable: 1.3.2 al momento del análisis
- Fecha Release : 29/01/2012
- Licencia : GPL

VENTAJAS Y DESVENTAJAS:

Si bien para el usuario es muy rápido crear una aplicación, la misma es bastante rígida y la configuración por medio de archivos de configuración .ini la hace un poco tediosa de mantener.

Por otro lado, si bien el último release es del 29 de Enero de 2012, el tamaño del foro oficial es de aproximadamente 900 integrantes (a Junio de 2012), lo cual parece no muy grande para cualquier consulta en el futuro.

2.2.3 VAADIN

Vaadin es un framework de aplicaciones web para aplicaciones de Internet enriquecidas (RIA). En contraste con soluciones basadas en librerías Javascript, cuenta con una robusta arquitectura de servidor. Esto significa que la mayor parte de la lógica de la aplicación se ejecuta de forma segura en el servidor. Google Web Toolkit (GWT) se utiliza en el lado del cliente para asegurar una experiencia de usuario rica y fluída. Vaadin es una gran colección de componentes de interfaz de usuario. La interfaz de usuario de la aplicación es un conjunto de componentes, tales como botones, tablas, árboles, etc. Los componentes utilizan eventos, listeners y enlace de datos para comunicarse entre sí y con su lógica de negocio. Vaadin es una arquitectura robusta para el desarrollo rápido de aplicaciones. La arquitectura basada en componentes junto con el lenguaje Java de tipo estático y las características de enlace de datos ayudará a construir aplicaciones que sean fáciles de modular, según sea necesario. El IDE y el apoyo de herramientas como herramienta visual de diseño ayudará a construir la interfaz de usuario de forma rápida.

VITALIDAD DEL PROYECTO AL MOMENTO DEL ANÁLISIS:

- Sitio: https://vaadin.com/home
- Tamaño comunidad integrantes del foro oficial: 3766 integrantes
- Foro Oficial: https://vaadin.com/forum
- Versión estable: 6.7.6 al momento del análisis
- Fecha Release : 6/3/2012
- Licencia : Apache

VENTAJAS Y DESVENTAJAS:

No tiene su IDE propio, por lo que se recomienda el uso de Eclipse (existe un plugin de Vaadin para Eclipse) o NetBeans.

Como framework en sí, requiere de una alta curva de aprendizaje para un usuario con poco conocimiento, alejándose del requerimiento de una herramienta RAD para una prototipación rápida.

2.2.4 WAVEMAKER

WaveMaker es un framework para el desarrollo rápido de aplicaciones web. Permite crear aplicaciones web sin necesidad de tener conocimiento previo en programación web.

La herramienta integra ACEGI, Dojo 1.0, autenticación, LDAP, ActiveDirectory y POJOs.

Está formado por WaveMaker Studio y WaveMaker Runtime.

WaveMaker Studio es un entorno de desarrolo drag-and-drop siguiendo el modelo MVC y se ejecuta en un navegador. Su instalación incluye varias aplicaciones web incluyendo un servidor web (Tomcat).

WaveMaker Runtime es un servidor de tiempo de ejecución y un cliente que está dentro de la aplicación creada por el WaveMaker Studio. Las aplicaciones se ejecutan en un servidor estándar de Java basado en Apache Tomcat, Dojo Toolkit, Spring e Hibernate. Además, las mismas usan estándares de Internet, incluyendo los componentes de SpringSource que proporcionan escalabilidad, rendimiento y portabilidad.

CARACTERÍSTICAS

- Contiene un entorno de desarrollo drag-and-drop; permite insertar todo tipo de elementos tan solo arrastrándolos hacia el formulario o pantalla que se está diseñando.
- WaveMaker Studio se ejecuta en un navegador. Las aplicaciones creadas con WaveMaker se pueden ejecutar mientras se están construyendo, lo que permite una rápida visualización de lo que se va haciendo en pocos segundos y de forma sencilla.
- WaveMaker construye aplicaciones mediante widgets, contiene una gran variedad de widgets configurables.
- Luego de definir una base de datos o importar una existente, se crea automáticamente un widget de datos para cada tabla en la base de datos. Esto hace que sea fácil crear formularios web.
- Las aplicaciones son manejados por eventos. El desarrollador es quien controla lo que pasa según las acciones que se toman. Por ejemplo, cuando un usuario hace click en un botón, elige un elemento de un menú o introduce datos en un widget de edición, uno o más eventos son disparados.
- Los widgets tienen una serie de propiedades que las define cada desarrollador para especificar su comportamiento.
- Los widgets también tienen funciones que se pueden llamar directamente a través de Javascript. Estas funciones permiten a los desarrolladores avanzados extender el comportamiento de un widget.
- Aprovechamiento de CSS, HTML y Java.
- Es multiplataforma y open source, existiendo versiones para Mac, Windows y Linux.
- Contiene servicios de base de datos (importar una base de datos con todas las funciones básicas CRUD), servicios web (conectarse con Web Services), servicios de Java (crear clases propias en Java), servicios de javascript (crear e integrar las funciones y widgets) y servicios de seguridad (crear fácilmente un acceso con diferentes formas de autenticación).
- Soporta los siguientes navegadores: Internet Explorer, Firefox, Safari y Chrome.
- Se puede instalar en Windows 7/XP/Vista/Server 2003, Mac OS X, Red Hat Enterprise Linux, CentOS, Debian o Ubuntu.
- WaveMaker soporta los siguientes servidores de aplicaciones: Tomcat, JBoss, GlassFish, WebSphere y WebLogic. WaveMaker se ejecuta en cualquier entorno J2EE.
- Soporta las siguientes base de datos: MySQL, PostgreSQL, HSQLDB, Oracle, Microsoft SQL Server e IBM DB2.

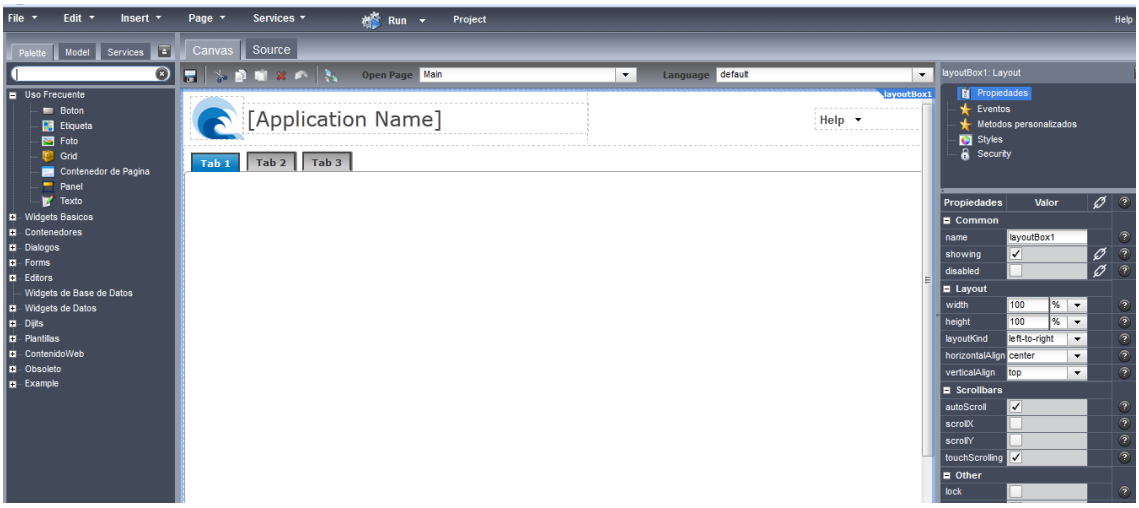


Figura 4: Imagen de ejemplo del WaveMaker Studio

VITALIDAD DEL PROYECTO AL MOMENTO DEL ANÁLISIS:

- Sitio: http://www.wavemaker.com
- Tamaño comunidad : 30.000 integrantes aprox.
- Foro Oficial: http://dev.wavemaker.com/forums/
- Versión estable: 6.4.5 al momento del análisis
- Fecha Release : 21/03/2012
- Licencia : Apache 2.0

VENTAJAS Y DESVENTAJAS

La principal ventaja de esta herramienta que motiva a la elección de este framework como framework de desarrollo, es el atractivo editor visual que permite un rápido prototipado de la aplicación con la posibilidad de hacer drag and drop de los widgets sobre el formulario.

Además se cuenta con muy buena documentación en la web, con tutoriales, videos, ejemplos y comunidad hispana.

El proyecto cuenta con una vitalidad considerable. Posee una comunidad sustentable de 30.000 integrantes, además se encuentra en continuo desarrollo (último release al momento del análisis v6.4.5 21/3/2012)

Se buscó una competencia al WaveMaker y se encontró AppFlower sobre PHP que tiene su propio IDE de forma similar a WaveMaker Studio.

2.2.5 APPFLOWER

AppFlower es un constructor de aplicaciones de forma rápida diseñado para crear soluciones fácilmente. Para crear las mismas se puede utilizar un ambiente de desarrollo integrado basado en la web: AppFlower proporciona un diseñador visual (Studio) para desarrollar aplicaciones sin conocimiento previo de programación, usando drag and drop y la filosofía de no codificar. Usuarios avanzados pueden llegar a crear aplicaciones más complejas.

La herramienta está construída sobre el framework MVC de PHP Symfony y su objetivo primordial es el de crear modernas aplicaciones web 2.0 con menor tiempo de tiempo de desarrollo que otros frameworks similares.

El look and feel de los componentes son atractivos, como se puede ver en la Figura 5.

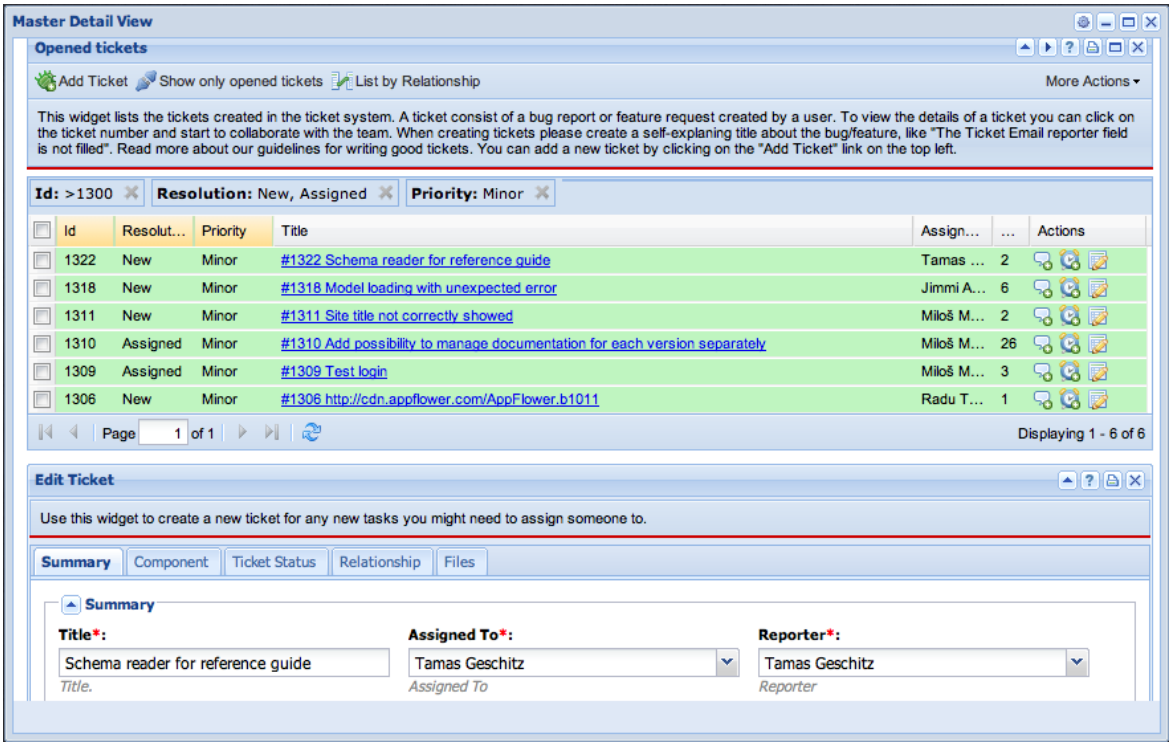


Figura 5: ejemplo de un formulario del estilo Master-Detail

VITALIDAD DEL PROYECTO AL MOMENTO DEL ANÁLISIS:

- Sitio: http://www.appflower.com/cms/home
- Foro Oficial: http://www.appflower.com/forum
- Versión estable: 1.2 al momento del análisis
- Fecha Release : 21/12/2011
- Licencia : MIT

VENTAJAS Y DESVENTAJAS

Tiene su propio IDE moderno y visualmente atractivo. Esto permite, según las premisas, un rápido prototipado para un usuario con poco conocimiento de programación, casi con cero líneas de código.

El diseñador visual se encuentra en versión Beta 2 (con fecha de release de Noviembre de 2011), lo que no lo hace del todo confiable mientras que la instalación del ambiente requiere cierto grado de complejidad. Solo aseguran su correcto funcionamiento en un ambiente instalado en una máquina virtual VMWare disponible para descargar del sitio oficial de la herramienta.

2.3 RESUMEN HERRAMIENTAS

Luego del análisis realizado se llegó a la conclusión de que existen herramientas libres que pueden ser utilizadas para desarrollar aplicaciones que faciliten la recolección de datos. Existen comunidades con personas trabajando en las herramientas estudiadas que proveen soporte y de esta forma dan vitalidad a las mismas.

En el momento de elegir una herramienta es necesario conocer las características que desea tener el producto a desarrollar a partir de las mismas, ya que no todas sirven para la resolución de los mismos problemas.

A partir de las características deseadas en las herramientas, que fueron mencionadas al comienzo de este capítulo, se definieron los ejes de comparación que se aprecian en la Tabla 1.

Basándonos en este estudio en el próximo capítulo detallaremos la elección de la herramienta.

	Ejes de Comparación	nuBuilder	Xataface	Vaadin	WaveMaker	AppFlower
	Software libre	Si	Si	Si	Si	Si
	Multiplataforma	Si	Si	Si	Si	Si
	Licencia	GPLv3	GPL	Apache	Apache 2.0	MIT
	Look and feel de aplicación generada (Excelente, Muy Bueno, Bueno, Regular, Malo)	Regular	Regular	sin dato	Muy Bueno	Muy Bueno
	Facilidad para hacer aplicaciones (Excelente, Muy Bueno, Bueno, Regular, Malo)	Muy Bueno (Los tipos de aplicaciones que se hacen son ABM de tablas)	Bueno (gestor de base de datos)	Regular (no tiene IDE)	Muy Bueno	Muy Bueno
Vitalidad	Tamaño de la comunidad	Foro con 24700 integrantes	foro con 890 integrantes	foro con 3766 integrantes	30.000 integrantes aprox.	sin dato
	Fecha última release	05/06/2012 (última estable 2009)	29/01/12	06/03/12	21/03/12	21/12/11
	Documentación (Excelente, Muy buena, Buena, Regultar)	Buena	Regular	sin dato	Muy Buena (cuenta con videos, manuales, foro en ingles y foro en español)	Buena
RAD	Prototipado Rápido de pantallas que no sean ABM	No	No	No	Si	Si
	poder crear formularios para ABM's de forma rápida	Si	Si	No	Si	sin dato
Servicios Externos	consumición de Web Services	Si (No integrado de forma nativa a la herramienta)	Si (No integrado de forma nativa a la herramienta)	sin dato	Si (de manera muy fácil)	Si
	autentitación LDAP	Si (No integrado de forma nativa a la herramienta)	Si (No integrado de forma nativa a la herramienta)	sin dato	Si	Si
	Lenguaje desarrollado	PHP	PHP	Java	Java	PHP
	Comentarios	Debido a la arquitectura que maneja la herramienta, el prototipado con este software no parece ser una tarea fácil.	Si bien para el usuario es muy rápido crear una aplicación, la misma es bastante rígida y la configuración por medio de archivos de configuración .ini la hace un poco tediosa de mantener.		Contiene un entorno de desarrollo drag-and-drop; permite insertar todo tipo de elementos tan solo arrastrándolos hacia el formulario o pantalla que se está diseñando.	La instalación del ambiente requiere cierto grado de complejidad. Solo aseguran su correcto funcionamiento en un ambiente instalado en una máquina virtual VMWare disponible para descargar del sitio oficial de la herramienta.

Tabla 1: Cuadro comparativo de herramientas evaluadas

3 ESPECIFICACIÓN DE LA SOLUCIÓN

En este capítulo se describe la solución desarrollada que permite informatizar la recolección de los datos correspondiente a los egresos de las diferentes Unidades Ejecutoras y ayudar a que la calidad de éstos sea la mejor posible. Ésta consiste en la implementación de una aplicación web, denominada Sistema de Codificación de Egresos (SCE).

En el transcurso de la mencionada descripción presentaremos la herramienta seleccionada para el desarrollo del prototipo, detallaremos los requerimientos funcionales y no funcionales así como la arquitectura utilizada y expondremos las decisiones de diseño tomadas.

3.1 ELECCIÓN DE LA HERRAMIENTA

En definitiva, tras hacer un análisis de las distintas herramientas, se llegó a la conclusión de que WaveMaker es el framework más adecuado para desarrollar este proyecto. En el cuadro comparativo del Capítulo 2 se evidencian las virtudes que nos inclinan a elegirlo. Entre sus ventajas más destacables se encuentra la capacidad de poder hacer prototipado rápido de pantallas que no sean ABM; característica que también tiene la herramienta AppFlower, aunque de este último no se obtuvieron datos que indiquen que se permite crear formularios para ABM's de forma rápida, cosa que sí lo permite WaveMaker.

A propósito de la herramienta AppFlower, el análisis llevó a concluir que en la mayoría de las características comparadas ha demostrado estar a la altura de WaveMaker. Sin embargo se seleccionó WaveMaker ya que de AppFlower no se obtuvieron datos del tamaño de la comunidad, de la posibilidad de consumir Web Services ni de si permite autenticación LDAP. No menos importante resulta el inconveniente de que la instalación de AppFlower requiere conocimientos avanzados pues encierra cierto grado de complejidad y solo aseguran su correcto funcionamiento en un ambiente instalado en una máquina virtual VMWare.

En cuanto al resto de las herramientas estudiadas, fueron descartadas pues no han logrado cumplir con la mayoría de las características requeridas.

3.2 REQUERIMIENTOS

Teniendo en cuenta los objetivos y el estado del arte del proyecto se definieron los requerimientos de la aplicación Sistema de Codificación de Egresos.

El relevamiento de los mismos se realizó con el responsable del proyecto por parte de ASSE mediante reuniones quincenales. Estos requerimientos fueron validados mediante la metodología de prototipado rápido.

3.2.1 REQUERIMIENTOS FUNCIONALES

- Usuarios: Deben existir al menos dos tipos de roles para los usuarios, administrador y registros médicos. Los usuarios con rol registros médicos deben tener asociada una unidad ejecutara correspondiente al lugar donde trabajan. Solo los usuarios con rol administrador deberán tener permisos para administrar otros

usuarios de la aplicación. El nombre del usuario y contraseña deben coincidir con los usuarios del correo de ASSE. Se deben poder inactivar los usuarios en la aplicación, un usuario inactivo puede entrar a la aplicación con su usuario y contraseña del correo pero no debe tener permisos sobre alguna funcionalidad.

- Encabezado: En el encabezado se debe dar la posibilidad de salir e ir al inicio de la aplicación.

- Egresos: Se deben poder crear, ver, modificar y auditar los egresos. El usuario con rol administrador debe poder gestionar todos los egresos de la aplicación; en cambio los usuarios con rol registros médicos solo podrán dar de alta y modificar egresos de su unidad ejecutora. Se deben de poder buscar egresos por cédula de identidad del paciente y fecha de egreso. No se puede ingresar ni modificar egresos con más de 3 meses de antigüedad. El egreso debe contener los mismos campos que la hoja de cierre de historia clínica, datos del paciente, datos de la hospitalización, diagnósticos, procedimientos, etc.

- Datos del usuario: los datos del paciente de un egreso deben ser cargados desde el padrón de usuarios existente en ASSE, no se deben poder modificar y se tiene que saber el estado de los mismos al momento del registro del egreso. La edad del paciente debe ser la edad que tenía la persona al momento del egreso.

- Códigos CIE9 y CIE10: A cada diagnóstico y procedimiento asociado al egreso se le debe asociar su código CIE10 o CIE9 respectivamente. Se debe poder buscar el código ingresando por lo menos los 3 primeros caracteres del mismo, para los códigos seleccionados se debe mostrar su descripción.

- Médicos: Un egreso puede tener asociado uno o ningún médico. El médico se debe buscar en el padrón de médicos de ASSE a partir del número de caja profesional, luego de seleccionarlo se deben mostrar los datos nombre, apellido y cédula de identidad del mismo.

- Data Warehouse: La información obtenida a partir del sistema de codificación de egresos se debe incorporar al Data Warehouse existente en ASSE. Ver Capítulo 5 para más detalles de este requerimiento.

- Informe: Se debe contar con una funcionalidad que permita generar un reporte para que los usuarios puedan obtener la información de los egresos que fueron registrados en su UE en un período determinado.

3.2.2 REQUERIMIENTOS NO FUNCIONALES

- Colores: Los colores que se deben utilizar en la interfaz del sistema de codificación de egresos son en tonos de celeste y amarillo.

- Resolución: La resolución mínima recomendada se definió en 1024 x 768 píxeles aunque el sistema fue desarrollado para una resolución óptima de 1366 x 768 píxeles.

- Formulario egreso: El llenado del formulario de egreso debe ser lo más parecido posible a la hoja de cierre de historia clínica. Se debe poder completar rápidamente, avanzar con tabulador entre los campos.

- Encabezado: Se deben mostrar los datos del usuario que ha iniciado la sesión, nombre y unidad ejecutora.

- Cantidades: La aplicación debe poder manejar 10.000 egresos por mes y 160 usuarios con rol registros médicos.

3.3 ARQUITECTURA

A continuación se explica la arquitectura utilizada, teniendo en cuenta que la plataforma seleccionada es WaveMaker.

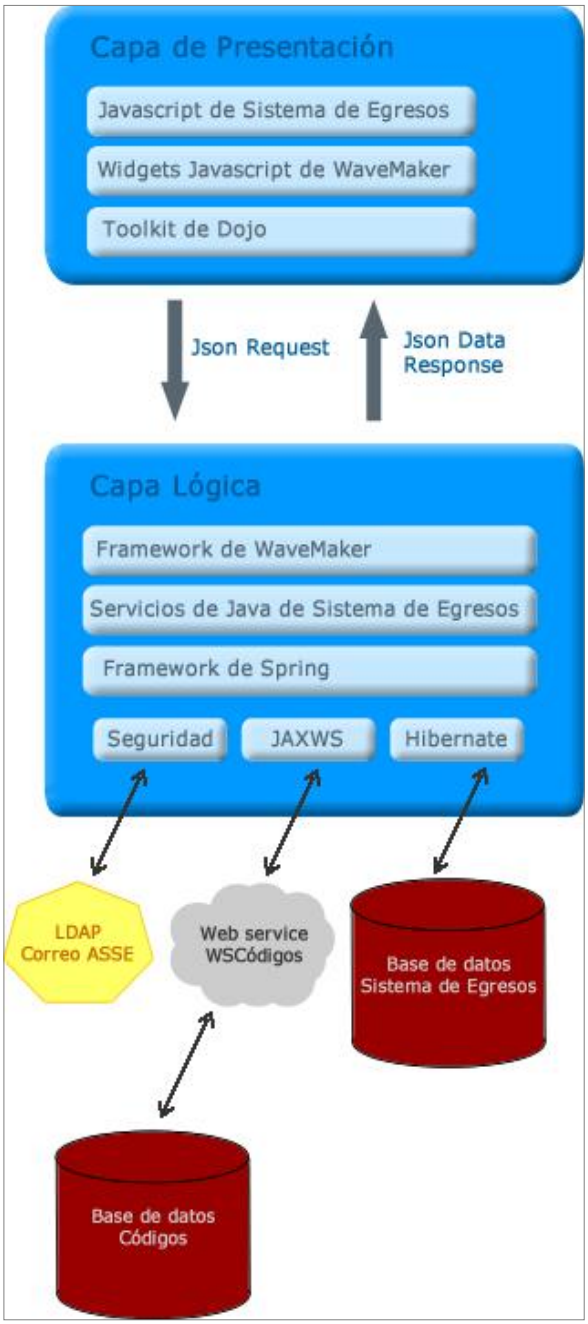


Figura 6: Componentes de la arquitectura

Dentro de los componentes de la arquitectura se encuentran los propios de WaveMaker, presentes en todas las aplicaciones desarrolladas en esta plataforma, y los del Sistema de Codificación de Egresos.

Componentes principales de WaveMaker:

- Dojo Toolkit: Framework utilizado por WaveMaker para generar de forma eficiente los Widget de interfaz de usuario (menús, pestañas, efecto de arrastrar y tirar, etc.) [6]
- JSON: Formato ligero para intercambio de datos. [7]
- Spring Framework: Actúa de soporte para desarrollar aplicaciones en java. Los servcios del proyecto son creados en XML definidos como Spring beans. Los beans son objetos instanciados, ensamblados y administrados por un contenedor Spring. [8]
- Seguridad: Incluye 4 tipos diferentes de autenticación de usuarios, LDAP, DEMO, base de datos y JOSSO. Nosotros el que usamos es LDAP configurado con los parámetros de conexión del LDAP del correo de ASSE.
- JaxWS: API de Java para la creación o invocación de servicios web. [9]
- Hibernate: Acceso de base de datos, realiza la correspondencia entre las clases java y la base de datos relacional.

Componentes del Sistema de Codificación de Egresos:

- Javascript de Sistema de Egresos: Por cada página web de la aplicación se generan dos archivos con extensión jscript, uno que define los widgets de la página y otro que define la lógica (eventos, chequeos, etc.).
- Servicios de Java de Sistema de Egresos: Punto de entrada para ejecutar código Java del lado del servidor. Permite más seguridad y más control (validaciones del lado del servidor, acceso a la base de datos).

3.4 DECISIONES DE DISEÑO

En esta sección se describen las decisiones de diseño que fueron tomadas durante el desarrollo del producto para satisfacer los objetivos planteados

3.4.1 INTERFACES CON SERVICIOS EXTERNOS

Dada la necesidad del sistema en obtener datos externos, ajenos a él, o mejor dicho, datos al cual él no provee mantenimiento o se encuentran actualmente en otros sistemas, surge la necesidad de interfaces entre estos.

Se detallan aquí las interfaces necesarias a ser implementadas para integrar el Sistema de Codificación de Egresos con los aplicativos de ASSE que deban convivir con él. El análisis que se hace en esta sección es de alto nivel. El detalle técnico de las interfaces con los diferentes sistemas, se encuentra en el Anexo 3.

Esta forma de integración entre los sistemas, evita entre otras cosas, la duplicidad de los datos y por ende bajar el costo de mantenimiento, dada la unicidad de los mismos.

Fue necesaria la construcción de algunos repositorios de datos para proveer a la institución de dicha información centralizada.

En la Tabla 2 se detallan las integraciones necesarias entre los sistemas.

ID Integración	Tipo Acción	Entidad	Sistema / Aplicativo de Origen	Invocado por	Tipo Integración
01	Consulta	Cie 9	Repositorio 1	SCE	WS
02	Consulta	Lista Cie 9	Repositorio 1	SCE	WS
03	Consulta	Cie 10	Repositorio 1	SCE	WS
04	Consulta	Lista Cie 10	Repositorio 1	SCE	WS
05	Consulta	Medico	Repositorio 1	SCE	WS
06	Consulta	Unidad Ejecutora	Repositorio 1	SCE	WS
07	Consulta	Lista Unid.Ejecutoras	Repositorio 1	SCE	WS
08	Consulta	Externos	Repositorio 1	SCE	WS
09	Consulta	Paciente	Padrón Usuarios de ASSE	SCE	WS
10	Consulta	Usuario Aplicativo	Repositorio 2	SCE	LDAP
11	Crear	Egreso	SCE	Pentaho	Batch

Tabla 2: Integraciones entre los sistemas

ID Integración: Número que identifica la transacción en el documento.

Tipo Acción: Acción que realiza el sistema invocador con los datos obtenidos del sistema origen.

Entidad: Objeto representado por los sistemas.

Sistema / Aplicativo de Origen: Sistema o Aplicación propietaria de los datos.

- Repositorio 1: Base de datos donde se encuentran los códigos requeridos en el que ASSE no tenía disponible para otros sistemas para consultar online.
- Padrón de Usuarios de ASSE: Aplicativo del cual se obtiene la información de los pacientes.
- Repositorio 2: Aplicativo con el cual se valida las credenciales del usuario al conectarse. Los datos están almacenados en directorios (LDAP).
- SCE: Sistema de Codificación de Egresos.

Invocado por: Sistema que requiere la integración. SCE Sistema de Codificación de Egresos, Pentaho (aplicativo de Data Warehouse).

Tipo Integración: Forma o tecnología que se utiliza para el envío de datos.

- WS Web Service (Tecnología de intercambio de información entre maquinas conectadas en red, tipo internet).
- LDAP: Protocolo de acceso a servicio de directorios. La información se encuentra en forma de directorio.
- Batch: Proceso repetitivo y secuencial para la obtención de datos desde un archivo o una conexión a una base de datos.

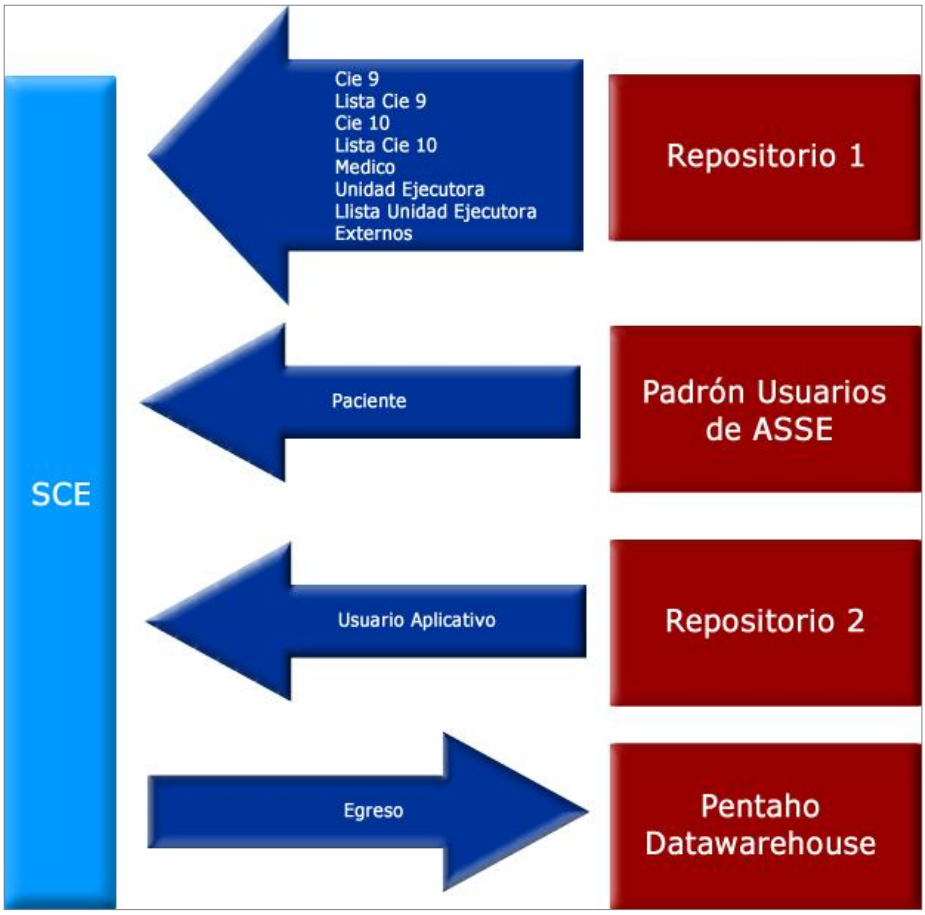


Figura 7: Representación esquemática de las interfaces a implementar

3.4.2 WAVEMAKER

- Alta y modificación de egresos: Al modificar un egreso se deben cargar todos los datos del egreso en un formulario igual al utilizado al dar de alta a un egreso. Por esta razón se decidió hacer uso de la misma página para ambas funcionalidades, alta y modificación de egreso. Así, en caso de hacer alguna modificación en los campos del egreso, sólo se deben hacer los cambios en un lugar solo.
- Auditoría de egreso: Se decidió realizar auditoría sobre los cambios de los egresos registrando quién hizo el cambio, sobre qué egreso y la fecha. Se evaluó la posibilidad de realizar la auditoría por campo de egreso pero junto con el cliente se llegó a la conclusión de que no era necesario.
- Sesión: Por motivos de seguridad, se decidió que al momento de caducar la sesión del usuario, se muestre un mensaje de error y se cargue la pantalla de inicio de sesión.

- Usuarios inactivos: Los usuarios nunca se eliminan físicamente de la base de datos del sistema. En su lugar, son marcados como inactivos quitándoles así los permisos sobre todas las funcionalidades. Cualquier usuario de LDAP del correo de ASSE puede iniciar sesión en el sistema; en caso de que no se haya dado de alta en la aplicación de egresos no contará con permisos. Esto se debe a que WaveMaker no brinda la posibilidad de personalizar la autenticación LDAP sin modificar el código de la propia herramienta.
- Servicios de Java: WaveMaker cuenta con variables llamadas "live variables" para el manejo de la base de datos (alta, baja, consulta y modificación). Sin embargo, para realizar estas acciones, se definieron servicios de java, los cuales se pueden consumir mediante una "service variable" de WaveMaker. De esta forma se obtuvo más control sobre la lógica del lado del servidor y se permitió realizar más validaciones.

3.5 MODELO DE DATOS

3.5.1 DISEÑO DE BASES DE DATOS

Como se mencionó en la sección 3.4.1, fue necesario la construcción de un repositorio de datos para proveer a ASSE información centralizada. Para este fin se creó una base de datos independiente a la utilizada por el SCE y Web Services capaces de consumir estos datos. Cabe destacar que el mantenimiento de las tablas de estas bases de datos no formó parte del alcance de este proyecto. Esta base de datos llamada "codigos" contiene la información de: Unidades Ejecutoras, Unidades Asistenciales, Médicos, Códigos CIE9 y Códigos CIE10.

La base de datos "egresos" contiene la información mantenida por el SCE. Al momento del alta de un egreso se obtienen los datos del paciente del padrón de usuarios de ASSE y se guardan junto con éste y no serán modificados a través de la aplicación. Por razones de simplicidad se mantuvieron los mismos tipos de datos para los campos obtenidos del padrón.

Excede el objetivo del proyecto mejorar el diseño de estos tipos de datos. Por ejemplo se observa que el sexo del paciente está definido como varchar 255, a pesar de que los valores posibles sean sólo 3 (Masculino, Femenino y Sin Dato).

Si en el futuro cambian los tipos de datos que devuelve el Web Service del padrón, se deberá modificar nuestro sistema.

En las figuras 8 y 9 se observan los diagramas de las bases de datos correspondientes a "egresos" y "codigos" respectivamente.

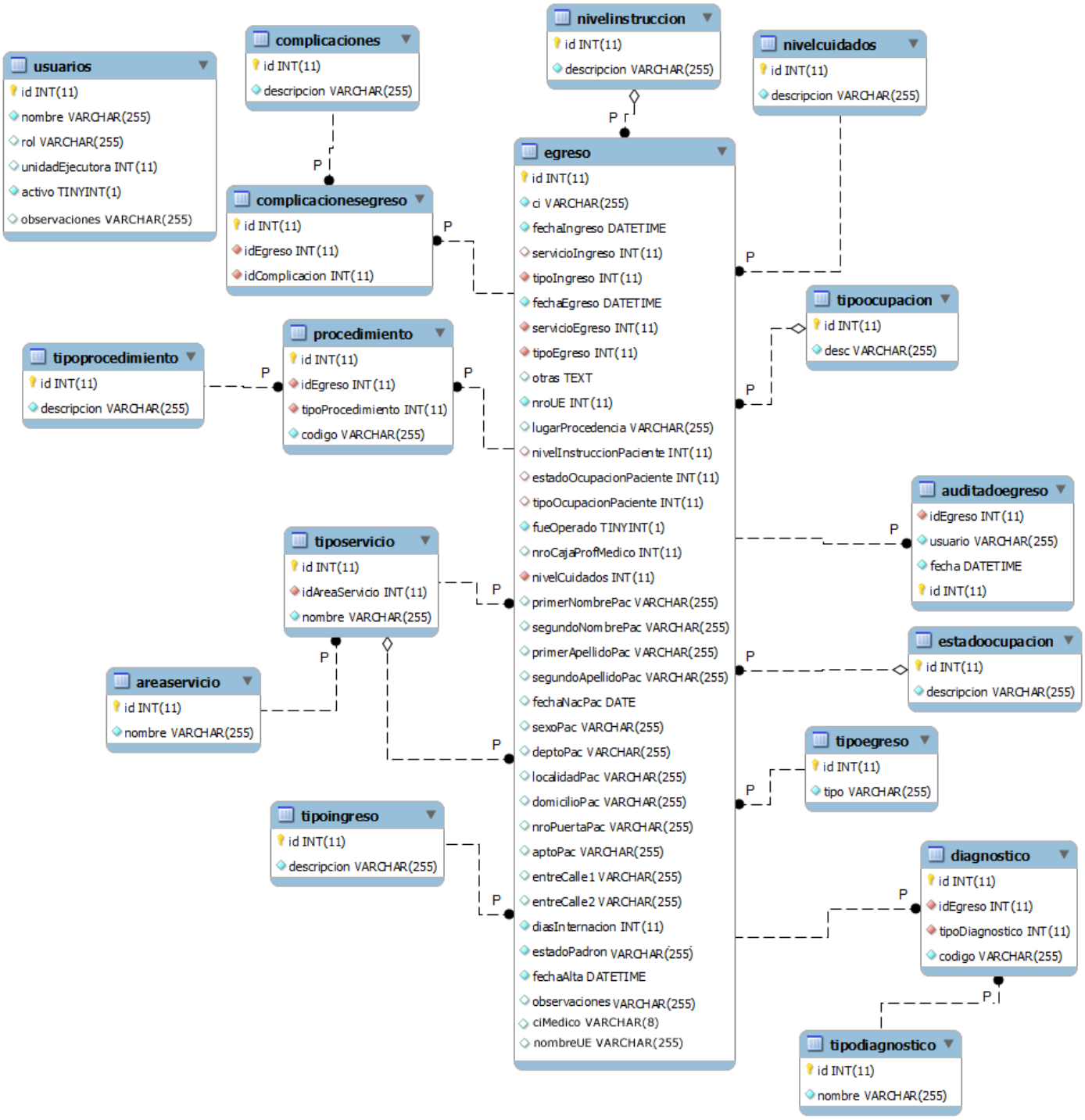


Figura 8: Tablas de la base de datos "egresos"

Figura 9: Tablas de la base de datos "codigos"

4 IMPLEMENTACIÓN DEL PRODUCTO

En este capítulo se describirá el entorno de desarrollo que se utilizó para la realización del producto así como las funcionalidades y el plan de pruebas que se llevó a cabo para garantizar su funcionamiento.

4.1 ENTORNO DE DESARROLLO

La aplicación se implementó con WaveMaker [10] versión 6.4.5 para el servidor de aplicaciones Apache Tomcat versión 7.0 [11].

Las bases de datos 'egresos' y 'codigos' son bases de datos relacionales MariaDB (versión 5.0).

Se usó un repositorio de código fuente con versionado. Durante el desarrollo de la solución se utilizó un repositorio subversión instalado en ASSE para mantener el código fuente de manera centralizada y trabajar en forma paralela y organizada. El cliente SVN utilizado fue Tortoise.

Para la creación del informe que se puede generar dentro del Sistema de Codificación de Egresos se utilizó la biblioteca de Java llamada JasperReports (versión 4.7.1) [12]. La edición y el diseño del informe se hizo con iReport 5.0.1.[13].

4.2 FUNCIONALIDADES

A partir de los requerimientos funcionales mencionados en la sección 3.2.1 de este documento se definieron las funcionalidades implementadas para el Sistema de Codificación de Egresos. Éstas son:

- Alta / Baja / Modificación / Visualización de Usuarios
- Alta / Modificación / Visualización de Egresos
- Auditoría de Egresos
- Informe de Cantidad de Egresos

En el Anexo 2 se detallan las mismas. Se mencionan los actores intervinientes, las pre y post condiciones así como una descripción para cada una de ellas.

4.3 PRUEBAS

A lo largo del desarrollo del producto se realizaron pruebas para verificar el correcto funcionamiento del mismo. En esta sección abordaremos las pruebas funcionales y no funcionales que fueron realizadas en los meses 8 y 11 en el ambiente de testing instalado en ASSE.

Las pruebas se realizaron en ASSE en computadoras con las siguientes características:

- Intel Core i7, 8 GB de RAM - Windows 7
- Intel Dual core, 2 GB de RAM - Ubuntu 12.04

Los navegadores web utilizados fueron Google Chrome (versión 19.0), Mozilla Firefox (versión 13.0) y Internet Explorer (versión 9.0).

4.3.1 PRUEBAS FUNCIONALES

Para cada prueba funcional se definió la funcionalidad a probar, el objetivo de la prueba, una descripción de la misma y el resultado esperado.

Funcionalidad	**Inicio de Sesión**
Objetivo de la prueba	Probar que cuando un usuario inicia sesión se muestran los botones y datos correspondientes.
Descripción	1. Iniciar sesión en el sistema con rol ADMIN/REGISTROS MÉDICOS.
Resultado Esperado	Cuando un usuario inicia sesión en el sistema se despliega, en la página principal y en el menú, los botones correspondientes a las secciones Registro Egreso, Ver/Modificar Egreso y Auditoría Egresos y en el encabezado de cada página se despliega el nombre del usuario y la opción para salir. Si el usuario que ha iniciado la sesión tiene rol REGISTROS MÉDICOS, en el encabezado también se muestra el nombre de la Unidad Ejecutora del usuario. Si el usuario que ha iniciado la sesión tiene rol ADMIN, en la página principal y en el menú también se muestra la sección Usuarios.

Funcionalidad	**Crear Usuario**
Objetivo de la prueba	Probar que el alta de usuario funciona correctamente
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Usuarios y seleccionar Añadir. 3. Ingresar todos los datos del usuario.

	4. Presionar el botón de Guardar. nota: se realizaron las pruebas creando un usuario con rol REGISTROS MÉDICOS y otra prueba con un usuario con rol ADMIN.
Resultado esperado	El usuario queda guardado correctamente.

Funcionalidad	**Crear Usuario**
Objetivo de la prueba	Probar que se verifiquen los campos obligatorios al dar de alta un usuario
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Usuarios y seleccionar Añadir. 3. Ingresar los datos del usuario dejando uno de los campos obligatorios vacíos 4. Presionar botón de Guardar. nota: se realizaron las pruebas creando un usuario con rol REGISTROS MÉDICOS y otra prueba con un usuario con rol ADMIN. Los campos que se fueron dejando vacíos fueron nombre, rol y para el usuario con rol REGISTROS MÉDICOS, la Unidad Ejecutora.
Resultado esperado	Se despliega un mensaje de error indicando que uno de los campos obligatorios no está ingresado, no se da de alta al usuario en el sistema.

Funcionalidad	**Crear Usuario**
Objetivo de la prueba	Corroborar que no se permita dar de alta a un usuario ya existente en el sistema
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Usuarios y seleccionar Añadir. 3. Ingresar nombre de un usuario ya existente, rol y Unidad Ejecutora (si corresponde). 4. Presionar el botón Guardar Usuario. nota: se realizó la prueba con un usuario con rol REGISTROS

	MÉDICOS y otro con rol ADMIN.
Resultado esperado	Se despliega un mensaje de alerta que indica que el usuario con ese nombre de usuario ya existe en el sistema. No se da de alta al usuario en el sistema.

Funcionalidad	**Modificar Usuario**
Objetivo de la prueba	Modificar rol de usuario de ADMIN a REGISTROS MÉDICOS
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Usuarios. 3. Ingresar nombre de un usuario ADMIN ya existente y presionar el botón Buscar Usuario 4. Presionar botón Modificar del usuario correspondiente 5. Seleccionar como rol REGISTROS MÉDICOS. 6. Seleccionar una Unidad Ejecutora. 7. Guardar los cambios realizados
Resultado esperado	Cuando se selecciona el rol REGISTROS MÉDICOS se despliega el campo para seleccionar la Unidad Ejecutora. El rol del usuario y su Unidad Ejecutora se modificaron satisfactoriamente.

Funcionalidad	**Modificar Usuario**
Objetivo de la prueba	Modificar rol de usuario de REGISTROS MÉDICOS a ADMIN
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Usuarios. 3. Ingresar nombre de un usuario ADMIN ya existente y presionar el botón Buscar Usuario 4. Presionar botón Modificar del usuario correspondiente 5. Seleccionar como rol ADMIN. 6. Guardar los cambios realizados
Resultado esperado	Cuando se selecciona el rol ADMIN se oculta el campo para seleccionar la Unidad Ejecutora.

	El rol del usuario se modificó satisfactoriamente y no cuenta más con una Unidad Ejecutora asociada.

Caso	**Modificar Usuario**
Objetivo de la prueba	Modificar un usuario existente, cambiándolo a inactivo y verificar que tenga permisos para realizar acciones en el Sistema.
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Usuarios. 3. Ingresar nombre de un usuario ya existente y presionar el botón Buscar Usuario 4. Presionar botón Modificar del usuario correspondiente 5. Marcar al usuario como inactivo 4. Guardar el usuario.
Resultado esperado	Luego de modificar el usuario, al iniciar sesión el mismo al sistema, no le aparece ninguna opción para acceder a las funcionalidades del Sistema, solamente el nombre de usuario, la Unidad Ejecutora (si corresponde) y el botón de Salir.

Caso	**Ver Usuario**
Objetivo de la prueba	Corroborar que los datos de un usuario se desplieguen correctamente
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Usuarios. 3. Ingresar nombre de un usuario ya existente y presionar el botón Buscar Usuario. 4. Hacer doble click sobre el usuario correspondiente.
Resultado esperado	Se despliegan correctamente todos los datos del usuario.

Caso	**Registrar egreso**
Objetivo de la prueba	Verificar que se dé de alta a un egreso correctamente.
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Registrar Egreso.

	3. Ingresar todos los datos del egreso. 4. Guardar el egreso. nota: se realizó la misma prueba con un usuario ADMIN
Resultado esperado	El egreso fue dado de alta correctamente.

Caso	**Registrar egreso**
Objetivo de la prueba	Verificar que se de de alta a un egreso con solo los campos obligatorios cargados.
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Registrar Egreso. 3. Ingresar sólo los campos obligatorios 4. Guardar el egreso. nota: se realizó la misma prueba con un usuario ADMIN
Resultado esperado	El egreso fue dado de alta correctamente.

Caso	**Registrar egreso**
Objetivo de la prueba	Verificar que se cargue la Unidad Ejecutora correspondiente al usuario logueado
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Registrar Egreso. 3. Ingresar todos los datos del egreso. 4. Guardar el egreso.
Resultado esperado	El campo de la Unidad Ejecutora fue cargado automáticamente con la del usuario registrado. Este campo está deshabilitado y no se puede modificar.

	El egreso fue dado de alta correctamente.

Caso	**Registrar egreso**
Objetivo de la prueba	Verificar que se habilita el campo de la Unidad Ejecutora cuando el usuario es ADMIN
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Registrar Egreso. 3. Ingresar todos los datos del egreso. 4. Guardar el egreso.
Resultado esperado	Se habilitó el campo Unidad Ejecutora para que se seleccionara una. El egreso fue dado de alta correctamente.

Caso	**Registrar egreso**
Objetivo de la prueba	Verificar que no se permita dar de alta a un egreso que no tenga todos los campos obligatorios ingresados.
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Registrar Egreso. 3. Ingresar todos los datos obligatorios del egreso menos uno. 4. Presionar el botón de Guardar. nota: se realizó una prueba por cada campo obligatorio, dejándolo vacío cada vez. Las mismas se hicieron con un usuario con rol ADMIN también.
Resultado esperado	Se despliega un mensaje de error indicando que falta el campo obligatorio. No se da de alta al egreso.

Caso	**Registrar y Modificar egreso**
Objetivo de la prueba	Verificar que se realicen todos los chequeos correspondientes a los diagnósticos en el momento de dar de alta a un egreso o modificarlo.
Descripción	1. Iniciar sesión en el sistema.

	2. Ingresar a la sección Registrar Egreso. 3. Ingresar todos los datos obligatorios del egreso. Se realizaron los siguientes chequeos: - Si se ingresa un diagnóstico principal cuyo código comienza con la letra “S” o “T”, se debe ingresar una causa externa que comience con “V”, “W”, “X” o “Y”. - Si el código del diagnóstico principal está entre O80 y O84 o entre O03 y O06, debe existir un procedimiento de cualquier tipo - Si la persona es de sexo masculino no puede existir ningún diagnóstico que comience con la letra “O”.
Resultado esperado	Se despliega un mensaje de error si no se cumple con alguno de los chequeos.

Caso	**Registrar y Modificar egreso**
Objetivo de la prueba	Verificar que se realicen todos los chequeos correspondientes a las fechas de egreso e ingreso en el momento de dar de alta a un egreso o modificarlo.
Descripción	1. Iniciar sesión en el sistema. 2. Ingresar a la sección Registrar Egreso. 3. Ingresar todos los datos obligatorios del egreso. Se realizaron los siguientes chequeos: - La fecha de ingreso no puede ser mayor que la fecha de egreso. - La fecha de egreso no puede ser de más de tres meses de antigüedad con respecto a la fecha actual. - No puede existir otro egreso cuya fecha de egreso y cédula de identidad del paciente sean iguales al que se está dando de alta.
Resultado esperado	Se despliega un mensaje de error si no se cumple con alguno de los chequeos.

Caso	**Modificar egreso**
Objetivo de la prueba	Cargar los datos de un egreso guardado para poder modificarlos, teniendo en cuenta que ciertos campos no se pueden modificar.

Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Buscar Egreso. 3. Ingresar cédula de identidad del paciente asociado al egreso. 4. Hacer click en el botón “Modificar” del egreso correspondiente. 5. Modificar los datos del egreso. 6. Guardar el egreso modificado.
Resultado esperado	Tanto el documento del paciente como los datos personales del mismo que provienen del padrón están deshabilitados para que no se puedan modificar. Se modificaron los datos del egreso de forma satisfactoria.

Caso	**Modificar Egreso**
Objetivo de la prueba	Verificar que no se permita modificar egresos dados de alta por un usuario de una Unidad Ejecutora diferente a la del usuario logueado.
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Buscar Egreso. 3. Ingresar cédula de identidad de algún paciente que tenga egresos registrados por una Unidad Ejecutora distinta a la del usuario logueado.
Resultado esperado	Se muestran todos los egresos de ese paciente y el botón “Modificar” de los egresos de otra Unidad Ejecutora está deshabilitado.

Caso	**Ver Egreso**
Objetivo de la prueba	Verificar que se filtren bien los egresos por cédula de paciente y fecha
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Buscar Egreso. 3. Ingresar cédula de identidad y fecha de algún egreso que exista en el sistema. 4. Presionar el botón Buscar. nota: se realizó también la prueba con un usuario de rol ADMIN
Resultado esperado	Se muestra el egreso que cumple con los filtros.

Caso	**Ver egreso**
Objetivo de la prueba	Verificar que los datos de un egreso se muestran satisfactoriamente.
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Buscar Egreso. 3. Ingresar cédula de identidad del paciente asociado al egreso. 4. Presionar el botón Buscar. 5. Hacer doble click sobre el egreso correspondiente. nota: se realizó la misma prueba con un usuario ADMIN
Resultado esperado	Se despliegan todos los datos del egreso y los campos están todos deshabilitados.

Caso	**Auditoría de egresos**
Objetivo de la prueba	Verificar que los datos de auditoría de los egresos de un paciente se muestran satisfactoriamente.
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Auditoría de Egresos. 3. Ingresar cédula de identidad de un paciente. 4. Presionar el botón Buscar. nota: se realizó la misma prueba con un usuario ADMIN
Resultado esperado	Se despliegan todos los registros de las modificaciones realizadas a los egresos asociados al paciente de todas las Unidades Ejecutoras, ordenados por fecha de modificación.

Caso	**Auditoría de egresos**
Objetivo de la prueba	Verificar que los datos de auditoría se filtran correctamente
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Auditoría de Egresos. 3. Ingresar cédula de identidad de un paciente y fecha correspondiente a un egreso existente.

	4. Presionar el botón Buscar. nota: se realizó la misma prueba con un usuario ADMIN
Resultado esperado	Se despliegan todos los registros de las modificaciones realizadas a los egresos que cumplen con esos filtros.

Caso	**Generar Informe**
Objetivo de la prueba	Verificar que se genere un informe con los datos correspondientes.
Descripción	1. Iniciar sesión en el sistema con rol ADMIN. 2. Ingresar a la sección Informe. 3. Ingresar rango de fechas (“fecha desde” y “fecha hasta”) 4. Seleccionar una unidad ejecutora. 5. Presionar el botón “Generar Informe”.
Resultado esperado	Se genera un PDF que contenga el número de egresos de la unidad ejecutora seleccionada (dentro del rango de fechas ingresado) agrupados por médico.

Caso	**Generar Informe**
Objetivo de la prueba	Verificar que se genere un informe con los datos correspondientes.
Descripción	1. Iniciar sesión en el sistema con rol REGISTROS MÉDICOS. 2. Ingresar a la sección Informe. 3. Ingresar rango de fechas (“fecha desde” y “fecha hasta”). 4. Presionar el botón “Generar Informe”.
Resultado esperado	Se genera un PDF que contenga el número de egresos de la unidad ejecutora a la que pertenece el usuario (dentro del rango de fechas ingresado) agrupados por médico.

4.3.2 PRUEBAS NO FUNCIONALES

Para diseñar y correr las pruebas de stress se utilizó una herramienta desarrollada por Apache Jakarta, llamada JMeter. [14]

El objetivo de las pruebas fue comprobar si la aplicación de codificación de egresos soporta 160 hilos concurrentes, pues es el total de codificadores que existen entre todas las Unidades Ejecutoras.

Al momento de diseñar las pruebas de stress también se tomó en cuenta que los codificadores completan un egreso en la aplicación en 5 minutos como máximo y como máximo se registran 10.000 egresos por mes en todas las Unidades Ejecutoras. Todos estos datos fueron brindados por el responsable por parte de ASSE.

Se seleccionaron las transacciones Alta de Egresos, Modificación Egreso y Auditoría Egreso y para cada una se definió un guión y se automatizó su script correspondiente.

Alta de Egreso

Evento	**Descripción**
Inicio de sesión	Inicio de sesión del usuario
Click Alta Egreso	Se selecciona Alta de Egreso
Completar Datos	Se llenan los datos del Egreso
Click Guardar	Se da de alta al egreso
Fin de sesión	Finaliza la sesión de usuario

Modificar Egreso

Evento	**Descripción**
Inicio de sesión	Inicio de sesión del usuario
Click Ver/Modificar Egreso	Se selecciona Ver/Modificar Egreso del menú
Buscar Egreso	Se busca Egreso para modificar
Cargar Egreso	Se cargan los datos del egreso
Click Guardar	Se modifican los datos del egreso
Fin de sesión	Finaliza la sesión de usuario

Auditoría Egreso

Evento	**Descripción**
Inicio de sesión	Inicio de sesión del usuario
Click Auditoría de Egresos	Se selecciona Auditoría de Egresos
Buscar Egreso	Se filtra ingresando cédula de paciente
Fin de sesión	Finaliza la sesión de usuario

Puesto que para probar las transacciones Modificar Egreso y Auditoría Egresos es necesario contar con datos en el sistema, se ejecutó primero Alta Egreso con 160 hilos concurrentes. Así se dieron de alta 160 egresos que fueron utilizados en las otras 2 transacciones; los datos de éstos fueron cargados desde un archivo csv.

5 DATA WAREHOUSE

Este capítulo describe la implementación del requerimiento funcional, mencionado en el punto 3.2.1 de este documento. La información obtenida a partir del sistema de codificación de egresos debe ser consolidada en un data warehouse con un paquete de herramientas determinadas por ASSE. Por lo tanto se pasará por alto detalles propios de dicho paquete, dada la idoneidad de ASSE. Por otro lado, este conjunto de herramientas es fruto del análisis exhaustivo de un proyecto de grado previo el cual las describe con mayor profundidad. En el comienzo del capítulo se hace una introducción básica de conceptos y definición de data warehouse a los efectos de brindar al lector de cierto vocabulario utilizado. Una correcta presentación del tema, demanda de por sí sola de toda una documentación que escapa al alcance de este proyecto. Seguidamente se describe el requerimiento relevado y su correspondiente implementación.

5.1 INTRODUCCIÓN

Un data warehouse (DW) es una base de datos corporativa de apoyo a la toma de decisiones que se caracteriza por integrar datos crudos de una o más fuentes distintas, depurando y almacenando la información necesaria de forma organizada para luego procesarla, permitiendo su análisis desde múltiples perspectivas y con grandes velocidades de respuesta. Permite a los directivos que lo utilizan, tener una visión más completa e integral de los procesos dado que el resultado de su implementación es conocimiento acerca del funcionamiento de la organización.

La creación de un DW representa en la mayoría de las ocasiones uno de los primeros pasos, desde el punto de vista técnico, para implantar una solución completa y fiable de BI. Al no generar datos por sí mismos se dice que este tipo de sistemas son fuentes secundarias de información, alimentados desde fuentes de datos externas. Las áreas o componentes básicos que conforman un sistema de DW son:

- Sistemas de Datos Fuentes: Donde se encuentra la información relevante que va a ser utilizada y cargada en el DW.
- Área de almacenamiento temporal: Aquí se guardan los datos limpios, combinados y estandarizados dentro de un sistema temporal de almacenamiento y sobre el cual se realizan procesamientos secuenciales de trabajos.
- ETL: Procesos que permiten obtener datos de distintas fuentes y depurarlos por medio de transformaciones para finalmente cargarlos en el DW.
- Área de presentación de datos: Es donde la información es organizada, almacenada y habilitada para la consulta directa de los usuarios finales.
- Área de Herramientas de acceso a datos: Corresponde a las herramientas de acceso a los datos del área de presentación y que permiten realizar análisis analíticos sobre los mismos.

Los modelos de almacenamiento para procesamiento analítico online (OLAP) están diseñados y optimizados para guardar grandes volúmenes de datos. Esto se realiza de forma estructurada de manera de poder cumplir con la meta de funcionar en forma eficiente frente a los requerimientos del usuario por lo que están orientadas al procesamiento analítico. Para ello se utilizan estructuras multidimensionales denominadas Cubos OLAP.

Los cubos o hipercubos OLAP son estructuras que representan los datos como una matriz en la cual sus ejes corresponden a los criterios de análisis y en los cruces se encuentran los valores a analizar. Estos cubos constan de dimensiones y medidas. Las dimensiones están relacionadas con los criterios de análisis de los datos, son variables independientes, representan los ejes del cubo y están organizadas en jerarquías. Las medidas son los valores o indicadores a analizar, se corresponden a datos asociados a relaciones entre los objetos del problema, son variables dependientes y se encuentran en la intersección de las dimensiones.

Existe la posibilidad de moverse dentro de las jerarquías de las dimensiones y observar de esta forma diferentes visiones de las medidas. Se puede seleccionar alguna de las dimensiones que se pretende analizar para realizar operaciones de agregación o desagregación, así como también fijar valores sobre algunas de estas dimensiones.

A continuación se presentan algunas definiciones de conceptos que corresponden a los componentes básicos de un cubo.

- Medidas: Las medidas son atributos utilizados como unidades de medida sobre las entidades y que conforman los valores o indicadores a analizar. Son estandarizadas dentro de un mismo DW para que todas las fuentes de datos expresen sus valores de igual manera.

- Dimensiones: Las dimensiones son los criterios principales de análisis de los datos de un DW y se desprenden de las entidades que forman parte del origen de datos del problema. Con las dimensiones se conforman los ejes de los cubos de información a ser utilizados en los sistemas OLAP. Pueden tener jerarquías internas para organizar los valores de datos que representan.

- Hechos: Los hechos son criterios de análisis que contienen las medidas numéricas que serán utilizados por los analistas del negocio para apoyar el proceso de toma de decisiones. Contienen datos cuantitativos necesarios para el análisis.

5.2 REQUERIMIENTO FUNCIONAL

Como se mencionó en el Capítulo 1 la metodología para su desarrollo fue similar a la utilizada durante el análisis de requerimientos, realizándose varias reuniones con el responsable por parte de ASSE en las que se definió qué información sería almacenada en el data warehouse. De estas reuniones de relevamiento y muestra de prototipos, surgen los siguientes componentes para el cubo a implementar:

Medidas:

- Cantidad de Egresos

Dimensiones:

Dado el Egreso se quiere agrupar por:

- Número y descripción de unidad ejecutora.
- Caja profesional del médico.
- Sexo de paciente.
- Grupos de edad del paciente al momento del egreso, dos jerarquías.

Grupos de Edad de FONASA

Menores de 1 año	(edad simple = 0)
de 1 a 4	
de 5 a 14	
de 15 a 19	
de 20 a 44	
de 45 a 64	
de 65 a 74	
75 y más	(todos los mayores de 75 años
No Indicado	(otros valores errores o valores inexistentes)

Grupos de Edad CIE-10

Menores de 1 año	(edad simple = 0)
1 año	(edad simple = 1)
2 años	(edad simple = 2)
3 años	(edad simple = 3)
4 años	(edad simple = 4)
de 5 a 9	
de 10 a 14	
de 15 a 19	
de 20 a 24	
de 25 a 29	
de 30 a 34	
de 35 a 39	
de 40 a 44	
de 45 a 49	
de 50 a 54	
de 55 a 59	
de 60 a 64	
de 65 a 69	
de 70 a 74	
de 75 a 79	

80 y más	(todos los mayores de 80 años
No Indicado	(otros valores errores o valores inexistentes)

- Nivel de cuidados.
- Tipo de egreso.
- Departamento de residencia del usuario. En caso de no existir la información se considera como "Sin Dato".
- Nivel de instrucción.
- Estado ocupación.
- Diagnóstico principal según capítulos, categorías, etc.
- Causas externas. Si existe más de una causa externa, se toma la primera que se ingresó en el egreso. Si no existe, se considera como "No aplica".
- Tipo de ingreso.
- Días internación - estadía.

Agrupación de Días de Estadía
de 1 a 2 días
de 3 a 4 días
de 5 a 10 días
de 10 a 15 días
de 15 días a 1 mes
más de 1 mes

- Fecha del egreso.
- Se realizó algún procedimiento? Si o No.
- Códigos CIE9 (procedimientos), se considera el primero que se ingresó si existe.
- Diagnóstico secundario, igual que causa externa, se toma el primero si hay.

5.3 IMPLEMENTACIÓN

5.3.1 INTRODUCCIÓN

A los efectos de implementar la solución del DW, se cuenta con un paquete de herramientas provista por ASSE. Este paquete surge del análisis de un proyecto de grado anterior, realizado con el organismo. Se da una breve introducción a la herramienta, extraída de dicho informe de proyecto, como forma de contextualizar la herramienta.

Pentaho

La plataforma Open Source Pentaho Business Intelligence está basada en tecnología Java y con un ambiente de implementación también basado en Java lo que la hace una herramienta flexible y adaptable a varios ambientes. La plataforma posee módulos de reportes, análisis olap, cuadros de mando (Dashboards), extracción de datos (Data Mining), integración de datos (ETL), administración y seguridad. Posee una interfaz de usuario bastante amigable.

Características Generales:

- Versión Evaluada: Pentaho BI Suite Community Edition - 3.5.2 Estable , Junio 2010
- Licenciamiento: GPL2, LGPL, MPL (Mozilla Public Licence)
- Versión Comercial: Pentaho BI Suite Enterprise Edición (Mayor cantidad de funcionalidades)
- Componentes Principales: ETL, Job Designer, Conectores, Repositorio Visual, Análisis OLAP, Metadata, Data Mining, Reporting, Dashboards, BI Platform, Administration Server.

Aplicaciones a utilizar que lo componen:

- ETL: Pentaho Data Integration (Kettle)
- Creación de metadata de cubos (Schema Workbench)

5.3.2 MODELADO DEL DATA WAREHOUSE

Al modelar las dimensiones, la implementación física de las mismas es un punto muy importante. Las dimensiones pueden contar con múltiples atributos y con jerarquías de varios niveles entre los mismos, lo que hace que se deba definir si se normalizan o no las tablas de dimensión diseñadas. En algunos casos, la dimensión se puede representar en una única tabla donde reside toda la información al nivel más bajo, o también mediante un conjunto de tablas relacionadas que respeten la tercera forma normal. El primero de los casos corresponde a un esquema estrella (que consta de una tabla de hechos central y un conjunto de tablas de dimensión relacionadas al hecho). Al normalizar las dimensiones se dice que se transforma al modelo estrella en un copo de nieve, debido al gráfico que se desprende de su estructura como se observa en la figura 10.

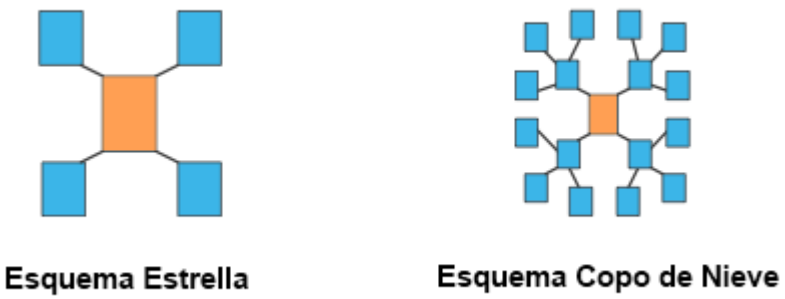


Figura 10: Esquema estrella y copo de nieve

El esquema estrella (star) es ideal por su simplicidad y velocidad para ser usado en análisis multidimensionales (OLAP en nuestro caso). Permite acceder tanto a datos agregados como de detalle.

El diseño de esquemas en estrella permite implementar la funcionalidad de una base de datos multidimensional utilizando una clásica base de datos relacional (más extendidas que las multidimensionales).

Otra razón para utilizar los esquemas en estrella es su simplicidad desde el punto de vista del usuario final. Las consultas no son complicadas, ya que las condiciones y las uniones (JOIN) necesarias sólo involucran a la tabla de hechos y a las de dimensiones, no haciendo falta que se encadenen uniones y condiciones a dos o más niveles como ocurriría en un esquema en copo de nieve(snowflake). En la mayoría de los casos son preferibles los de estrellas por su simplicidad respecto a los de copo de nieve por ser más fáciles de manejar.

Finalmente, es la opción con mejor rendimiento y velocidad pues permite indexar las dimensiones de forma individualizada sin que repercuta en el rendimiento de la base de datos en su conjunto.

Como se ve en el diagrama de tablas del DW (Figura 11), se puede apreciar el modelo estrella que fue el elegido por las razones anteriores.

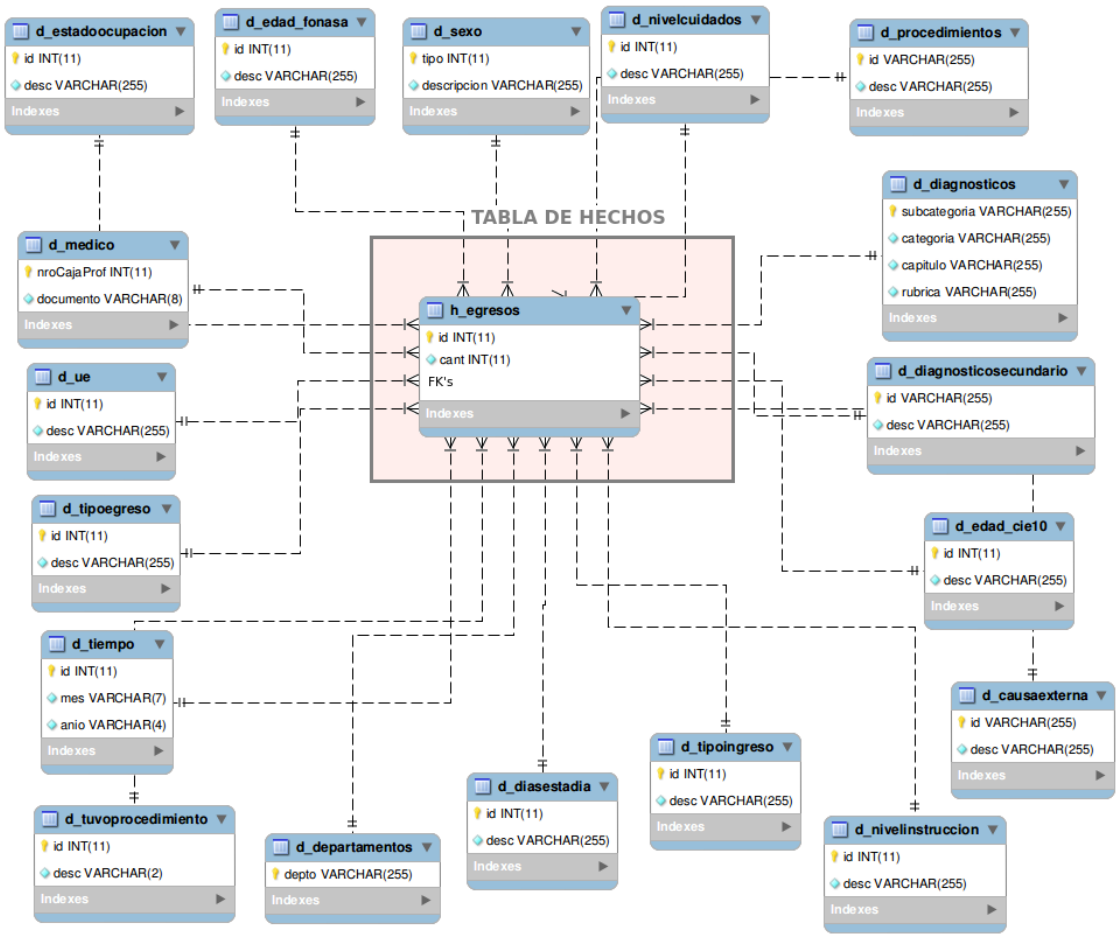


Figura 11: Tablas de la base de datos "dw_egresos"

Aclaraciones de la Figura 11: FK's en la tabla h_egresos de la Figura 11, representa al conjunto de campos que hacen de claves foráneas de todas las tablas con las que está relacionada, a los efectos de clarificar el diagrama. Estas no componen la clave de la tabla por una limitante de la base de datos, la que solo permite un máximo de 16 campos como clave compuesta y en nuestro caso contamos con 18 tablas (dimensiones). Por tal motivo se toma el criterio de solucionarlo con un campo "Id" autoincremental entero como clave de la tabla h_egresos.

Aplicando los conceptos introducidos en la sección anterior tenemos:

- Sistema de Datos Fuente: Base de datos "egresos". Ver Figura 8.
- Área de almacenamiento temporal: tabla h_egresos_tmp de la base de datos "dw_egresos". Ver Figura 12

Figura 12: Tabla de almacenamiento temporal en la base de datos "dw_egresos"

- Medidas: Cantidad de Egresos. Campo "cant" de la tabla h_egresos.
- Dimensiones:
  - Número y descripción de unidad ejecutora. Tabla d_ue
  - Caja profesional del médico. Tabla d_medico
  - Sexo de paciente. Tabla d_sexo
  - Grupos de edad del paciente al momento del egreso. Tablas d_edad_fonasa
  - y d_edad_cie10.
  - Nivel de cuidados. Tabla d_nivelcuidados.
  - Tipo de egreso. Tabla d_tipoegreso.
  - Departamento de residencia del usuario. Tabla d_departamentos.
  - Nivel de instrucción. Tabla d_nivelinstruccion

- Estado ocupación. Tabla d_estadoocupacion.
- Diagnóstico principal. Tabla d_diagnosticos. Los campos categoría, capítulo y rubrica, implementan el nivel de agrupación.
- Causas externas. Tabla d_causaexterna.
- Tipo de ingreso. Tabla d_tipoingreso
- Días internación - estadía. Tabla d_diasestadia.
- Fecha del egreso. Tabla d_tiempo. Solo se toma el mes y año, por los cuales se puede agrupar.
- Se realizó algún procedimiento. Tabla d_tuvoprocedimiento.
- Códigos CIE9 (procedimientos). Tabla d_procedimientos.
- Diagnóstico secundario. Tabla d_diagnosticosecuandario.

- Hechos: Tabla h_egresos

5.3.3 ETL

En esta sección se describe el procedimiento de carga del DW.

Como se mencionó en la introducción, ETL está compuesto por un único proceso que permite obtener datos de una fuente (base de datos egresos), depurarlos por medio de transformaciones para finalmente cargarlos en el DW (base de datos dw_egresos) . Este proceso fue realizado con la herramienta de desarrollo Spoon incluida en el paquete antes mencionado, Pentaho. Los conceptos que maneja dicha herramienta son los de pasos, transformaciones y trabajos. Por lo tanto, un trabajo es un conjunto de transformaciones y una transformación es un conjunto de pasos unidos por líneas de flujos.

A continuación se muestra las diferentes transformaciones que componen el trabajo o Job "Main "(Figura 16) y una descripción de sus pasos más importantes.

La primera transformación que compone el job Main es Inicializar (ver Figura 13). La finalidad de esta, es la carga de la dimensión d_tiempo con los últimos 3 meses/año según la fecha actual de ejecución. Esto es debido a que los usuarios cuentan con permisos de ingreso para egresos con una antigüedad menor a tres meses. Por lo tanto y ya que el proceso corre todos los días se está constantemente eliminando y volviendo a cargar dichos egresos.

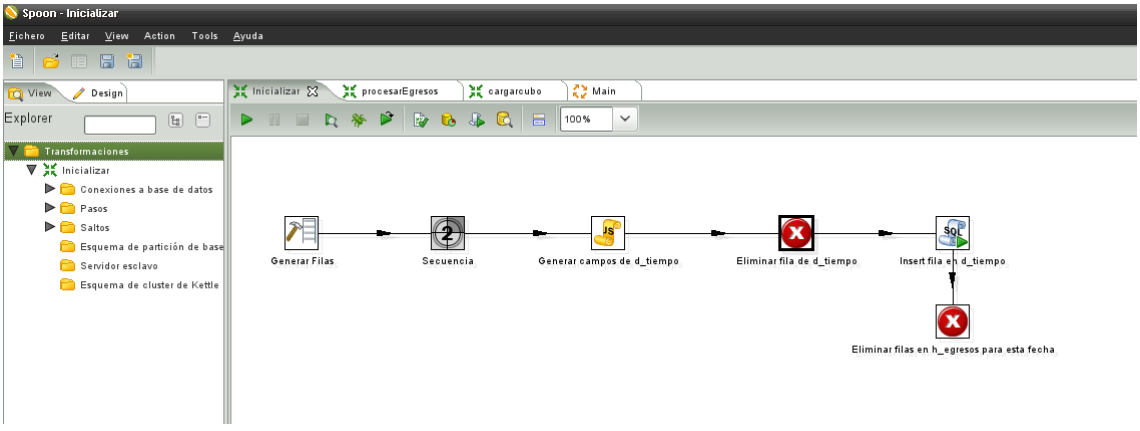


Figura 13: Transformación Inicializar

Seguidamente en el flujo del job Main, se procesa la transformación "procesarEgresos" (Ver Figura 14). Esta básicamente se encarga de la obtención de los datos de la tabla "egreso", la transformación de estos según las diferentes dimensiones y luego la carga de la tabla temporal "h_egresos_tmp" .

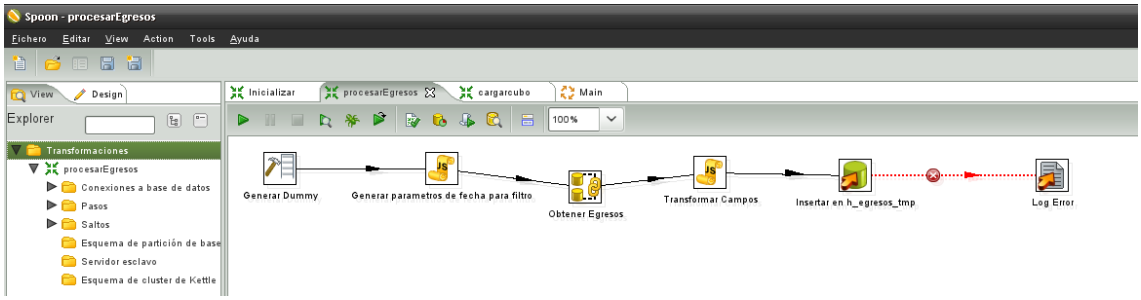


Figura 14: Transformación procesarEgresos

En el siguiente recuadro, se muestra el SQL que se ejecuta para obtener los datos de la tabla egreso. Los parámetros "?" son generados en el paso anterior según la premisa de los últimos tres meses con respecto a la fecha de hoy del sistema.

Step Obtener Egresos de la transformación procesarEgresos
SELECT A.nroUE, A.nroCajaProfMedico, A.sexoPac, A.fechaNacPac, A.fechaEgreso, A.nivelCuidados, A.tipoEgreso, A.deptoPac, A.nivelInstruccionPaciente, A.estadoOcupacionPaciente, A.tipoIngreso, A.diasInternacion, (YEAR(A.fechaIngreso)*100) + MONTH(A.fechaIngreso)AS fech, A.fueOperado, ifnull((select P.codigo from procedimiento P where P.idEgreso = A.id order by P.id Limit 0,1),0) as procedimiento, ifnull((select D.codigo from diagnostico D where D.idEgreso = A.id and D.tipoDiagnostico = 1 order by D.id Limit 0,1),0) as diagpri, ifnull((select D.codigo from diagnostico D where D.idEgreso = A.id and D.tipoDiagnostico = 2 order by D.id Limit 0,1),0) as diagsec, ifnull((select D.codigo from diagnostico D where D.idEgreso = A.id and D.tipoDiagnostico = 3 order by D.id Limit 0,1),0) as causaex FROM egreso A WHERE A.fechaIngreso >= ? AND A.fechaIngreso <= ?

En el siguiente recuadro se muestra el código que implementa todas las transformaciones de los datos.

Step Transformar Campos de la transformación procesarEgresos

var sexo = "";
var edad = fechaNacPac;
var edadFonasa = 9;
var edadCie10 = 0;
var diasinter = 0;

```
var tuvoproc = 2;

if(sexoPac == 'M' )
        sexo = 1;
else if(sexoPac == 'F')
        sexo = 2;
else
        sexo = 3;

if (nroCajaProfMedico == null) nroCajaProfMedico = 0;

if (edad != null){
edad = trunc(dateDiff(fechaNacPac,fechaEgreso,"hh")/24/365.25);
if (edad == 0) edadFonasa = 1;
else if ((edad >= 1) && (edad <= 4)) edadFonasa = 2;
else if ((edad >= 5) && (edad <= 14)) edadFonasa = 3;
else if ((edad >= 15) && (edad <= 19)) edadFonasa = 4;
else if ((edad >= 20) && (edad <= 44)) edadFonasa = 5;
else if ((edad >= 45) && (edad <= 64)) edadFonasa = 6;
else if ((edad >= 65) && (edad <= 74)) edadFonasa = 7;
else if (edad >= 75) edadFonasa = 8;

if (edad == 0) edadCie10 = 1;
else if (edad == 1) edadCie10 = 2;
else if (edad == 2) edadCie10 = 3;
else if (edad == 3) edadCie10 = 4;
else if (edad == 4) edadCie10 = 5;
else if ((edad >= 5) && (edad <= 9)) edadCie10 = 6;
else if ((edad >= 10) && (edad <= 14)) edadCie10 = 7;
else if ((edad >= 15) && (edad <= 19)) edadCie10 = 8;
else if ((edad >= 20) && (edad <= 24)) edadCie10 = 9;
else if ((edad >= 25) && (edad <= 29)) edadCie10 = 10;
else if ((edad >= 30) && (edad <= 34)) edadCie10 = 11;
else if ((edad >= 35) && (edad <= 39)) edadCie10 = 12;
else if ((edad >= 40) && (edad <= 44)) edadCie10 = 13;
else if ((edad >= 45) && (edad <= 49)) edadCie10 = 14;
else if ((edad >= 50) && (edad <= 54)) edadCie10 = 15;
else if ((edad >= 55) && (edad <= 59)) edadCie10 = 16;
else if ((edad >= 60) && (edad <= 64)) edadCie10 = 17;
else if ((edad >= 65) && (edad <= 69)) edadCie10 = 18;
else if ((edad >= 70) && (edad <= 74)) edadCie10 = 19;
else if ((edad >= 75) && (edad <= 79)) edadCie10 = 20;
else if (edad >= 80)  edadCie10 = 21;
}

if ((deptoPac == null)||(deptoPac == "")) deptoPac = "SIN DATO";

if (nivelInstruccionPaciente == null) nivelInstruccionPaciente = 5;

if (estadoOcupacionPaciente == null) estadoOcupacionPaciente = 7;

if (diasInternacion <= 2) diasinter = 1;
else if ((diasInternacion >= 3) && (diasInternacion <= 4)) diasinter = 2;
```

```
else if ((diasInternacion >= 5) && (diasInternacion <= 10)) diasinter = 3;
else if ((diasInternacion >= 11) && (diasInternacion <= 15)) diasinter = 4;
else if ((diasInternacion >= 16) && (diasInternacion <= 30)) diasinter = 5;
else if (diasInternacion >= 31) diasinter = 6;

if (fueOperado == 1) tuvoproc = 1;
```

Seguidamente en el flujo del job Main, se procesa la transformación "cargarcubo" (Ver Figura 15). Esta transformacion agrupa y cuenta los datos de egresos transformados en la tabla temporal h_egresos_tmp e inserta estos en la tabla h_egresos que representa nuestro cubo.

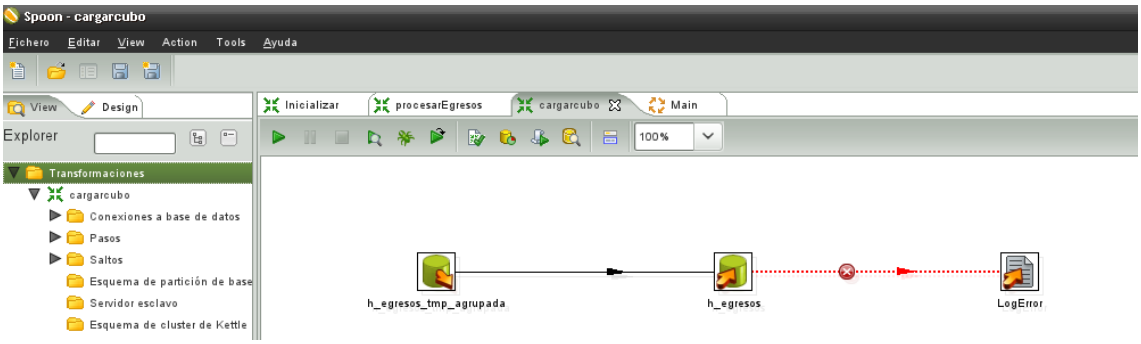


Figura 15: Transformación cargarcubo

En el siguiente recuadro, se muestra el SQL que se ejecuta para obtener los datos de la tabla egreso. Los parámetros "?" son generados en el paso anterior según la premisa de los últimos tres meses con respecto a la fecha de hoy del sistema.

Step h_egresos_tmp_agrupada de la transformación cargarcubo

```
SELECT  nroUE, nroCajaProf, sexo, diagnostico, edad_fonasa, edad_cie10, nivelcuidados
, tipoegreso, depto, nivelinstruccion, estadoocupacion, tipoingreso, diasestadia, fecha,
causaexterna, diagnosticosecundario, tuvoprocedimiento, procedimiento, count(*) as cant
FROM h_egresos_tmp group by nroUE, nroCajaProf, sexo, diagnostico, edad_fonasa
, edad_cie10, nivelcuidados, tipoegreso, depto, nivelinstruccion, estadoocupacion
, tipoingreso, diasestadia, fecha, causaexterna, diagnosticosecundario, tuvoprocedimiento
, procedimiento
```

La siguiente figura muestra todo el job con el correspondiente flujo de llamadas a las transformaciones anteriormente comentadas.

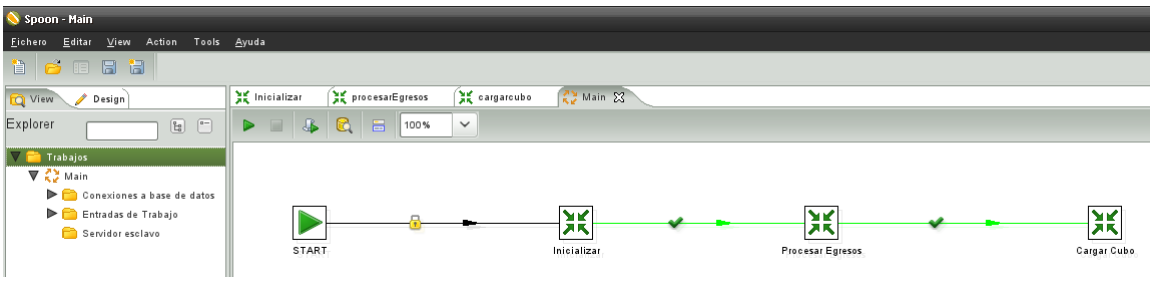


Figura 16: Trabajo(JOB)Main

Nota: Como se ve en la figura 14, no se hace validación de valores incorrectos o inexistentes en las dimensiones. Se tomó está decisión a los efectos de tomar esta tecnología como herramienta de reportería y manejando estos datos sin evolución en el tiempo inmediato. Ver sección 6.2 correspondientes a trabajos futuros.

6 CONCLUSIONES Y TRABAJO FUTURO

6.1 CONCLUSIONES Y RESULTADOS OBTENIDOS

El proyecto se enmarca en las actividades de investigación y desarrollo llevadas adelante por ASSE. En dicho esquema, se pretende apoyar las reformas sanitarias a nivel nacional con sistemas de información confiables y de valor agregado. Los objetivos que se definieron para el mismo fueron:

- Investigar y elegir un framework web open source que facilitara la realización de formularios web para la recolección de datos obtenidos en distintas Unidades Ejecutoras. Debería cumplir ser multiplataforma, tener una estética atractiva y moderna, poder utilizar autenticación con LDAP, generar formularios web de forma ágil, prototipado rápido y vitalidad adecuada.

- Desarrollar una herramienta web sobre el framework elegido que ayudara a:
    - Facilitar la recolección de los datos en las diferentes Unidades Ejecutoras.
    - Que la calidad de los datos sea la mejor posible.
    - Facilitar la consolidación de la información enviada por distintas Unidades Ejecutoras.
    - Facilitar el procesamiento de los datos recolectados.

A continuación se detalla la forma que en que se cumplió cada objetivo y las dificultades encontradas.

Investigar y elegir un framework web open source que facilite la realización de formularios web para la recolección de datos obtenidos en distintas Unidades Ejecutoras.

La elección de la herramienta se realizó considerando los requisitos establecidos por ASSE: debía ser software libre, multiplataforma, de estética atractiva y moderna, debía poder consumir web services, contar con autenticación con LDAP y era deseable que la herramienta pudiera ser utilizada por usuarios no expertos. Además, la herramienta debía permitir la generación de formularios de forma ágil para poder realizar prototipado rápido y debía contar con una buena documentación y una comunidad sustentable.

Tras hacer un análisis del estado del arte se llegó a la conclusión de que WaveMaker sería el framework más adecuado para desarrollar este proyecto porque a priori cumpliría con las características enumeradas.

Luego del desarrollo del Sistema de Codificación de Egresos se comprobó que efectivamente WaveMaker cumplió con las expectativas esperadas.

El entorno de desarrollo drag-and-drop de esta herramienta hizo posible la generación de formularios de forma ágil, lo cual permitió aplicar la metodología de prototipado rápido para poder validar los requerimientos y permitir la visualización de la aplicación en etapas tempranas.

WaveMaker nos permitió definir fácilmente, entre otras cosas, acciones en los botones, tablas, validaciones de los campos, visibilidad de componentes según el rol del usuario, páginas y navegación entre las mismas. Estos aspectos ya están resueltos por el framework lo que generó un ahorro sustantivo de tiempo durante la etapa de implementación. Sin embargo, WaveMaker los resolvía mediante código javascript lo cual

hacía la tarea de detectar la causa de un error un poco tediosa. Para la detección de las causas se debió recurrir a otras herramientas como ser complementos de navegadores web.

A pesar de que WaveMaker cuenta con una extensa documentación para resolver algunos problemas la misma no era suficiente y se debió recurrir a foros. Cabe destacar que la herramienta posee una gran comunidad activa tanto en inglés como en español.

Desarrollar una herramienta web sobre el framework elegido que ayude a:

- Facilitar la recolección de los datos en las diferentes Unidades Ejecutoras.

Para el cumplimiento de este objetivo se desarrolló una aplicación llamada Sistema de Codificación de Egresos que permite informatizar la recolección de los datos correspondiente a los egresos de las diferentes Unidades Ejecutoras.

Con este sistema los usuarios de las distintas Unidades Ejecutoras pueden registrar diariamente los egresos los cuales son almacenados en una base de datos centralizada, lo que permite que el responsable de ASSE acceda a éstos sin necesidad de reunir y consolidad las planillas que solía recibir vía e-mail.

- Que la calidad de los datos sea la mejor posible.

En el proceso de desarrollo del producto se subsanaron diversas carencias detectadas en la metodología que hasta entonces aplicaba ASSE, mejorando significativamente la calidad de los datos. Esto se logró, entre otras mejoras, al sustituir por un formulario la planilla de cálculo donde se cargaban los datos de los egresos. En el nuevo formulario ahora es posible realizar validaciones que permiten detectar problemas de inconsistencias y duplicaciones. Hasta el momento se destinaban recursos humanos que revisaban manualmente cada egreso para detectar inconsistencias.

También es posible cargar los datos de los pacientes obteniendo la información desde el padrón de usuarios de ASSE sin necesidad de ingresarlos y así evitar los frecuentes errores que se producían al introducirlos manualmente.

- Facilitar la consolidación de la información enviada por distintas Unidades Ejecutoras y el procesamiento de los datos recolectados.

El equipo de Gestión de la Información de ASSE obtenía datos estadísticos a partir de la consolidación de todas las planillas recibidas, realizando dicha consolidación de forma prácticamente manual y luego se utilizaba la herramienta SPSS2 para su análisis. Esta forma de trabajo consumía mucho tiempo y requería de conocimientos técnicos específicos.

La integración del Data Warehouse automatizó esta consolidación y permitió la visualización de la información mediante reportes y vistas de análisis.

A continuación en la figura 17, se muestra una vista de análisis extraída con la herramienta Pentaho usada en el DW mencionado en el Capítulo 5. Los datos obtenidos corresponden a la información almacenada al 13/8/2013 (los primeros egresos fueron registrados a fines de abril). Simplemente muestra la cantidad de egresos registrados por la dimensión del cubo Unidad Ejecutora. Cabe destacar que la puesta en producción en las

distintas Unidades Ejecutoras se hizo de forma paulatina; situación ésta que queda particularmente destacada en la figura 17, donde pueden resultar llamativas las diferencias entre las cantidades registradas.

UE	Cantidad de Egresos
4 - CHPR - H. Mujer	92
5 - Hospital Maciel	1
6 - Hospital Pasteur	877
9 - I.N.O.T.	1
10 - I.N.R.U.	107
12 - Hospital Saint Bois	799
16 - Cen. Dep. Canelones	43
18 - Cen. Dep. Colonia	74
21 - Cen. Dep. Florida	38
22 - Cen. Dep. Lavalleja	46
23 - Cen. Dep. Maldonado	411
24 - Cen. Dep. Paysandú	900
28 - Cen. Dep. Salto	1.273
31 - Cen. Dep. Tacuarembó	84
62 - Cen. Aux. Las Piedras	21

Figura 17: Cantidad de Egresos en producción al 13/8/2013

UE	Mes	Cantidad de Egresos
28 - Cen. Dep. Salto	04/2013	42
	05/2013	97
	06/2013	663
	07/2013	471

Figura 18: Egresos del Centro Departamental de Salto

La figura 18 muestra la cantidad de egresos registrados por el Centro Departamental de Salto discriminados por mes.

Del mismo modo que en los ejemplos anteriores, la herramienta de vista de análisis de Pentaho facilita los posibles cruzamientos que se deseen a partir de las 18 dimensiones cargadas en el cubo. También permite exportar una vista de análisis a una planilla de cálculo para su posterior procesamiento. En la Tabla 3 se muestra la exportación a planilla de cálculo de la dimensión correspondiente al diagnóstico principal de un egreso. Los datos

obtenidos corresponden a la información almacenada al 13/8/2013 con las mismas 15 Unidades Ejecutoras de la Figura 17. En esta Tabla se observa la cantidad total de egresos registrados en este período y según la clasificación internacional de enfermedades (CIE10) las afecciones principales registradas para dichos egresos hospitalarios.

Capitulo Diagnóstico Principal	Cantidad de Egresos
TODOS	4.964
242 - Otras complicaciones del embarazo y del parto	262
169 - Neumonía	259
133 - Catarata y otros trastornos del cristalino.	193
175 - Bronquitis, enfisema y otras EPOC	164
281 - Otros traumatismos de regiones específicadas, no clasificadas en otra parte.	157
179 - Otras efermedades del sistema respiratorio	150
270 - Otros síntomas, signos y hallazgos anormales clínicos y de laboratorio, NCOP.	147
151 - Insuficiencia cardíaca	135
195 - Colelitiasis y colecistitis.	134
243 - Parto único espontáneo	133
239 - Otra atención materna relacionada con el feto y con la cavidad amniótica y con posibles problemas del parto	119
007 - Tuberculosis respiratoria.	101
217 - Otras enfermedades del sistema urinario	89
267 - Dolor abdominal y pélvico.	89
170 - Bronquitis aguda y bronquilitis aguda	85
198 - Infecciones de la piel y del téjido subcutáneo.	77
192 - Otras enf.de los intestinos y del peritoneo.	73
...	...

Tabla 3: Ejemplo de exportación de vista de análisis con Pentaho

A propósito de los objetivos enumerados, es necesario mencionar la satisfacción demostrada tanto por el responsable por parte de ASSE como por los usuarios finales que asistieron a las distintas presentaciones del producto, que nos manifestaron su predisposición a utilizar un sistema que a todas luces aparentaba ser más eficiente que el engorroso método anterior.

Cabe destacar que a pesar de la dimensión y complejidad de una institución pública como ASSE, el proyecto logró de todas formas dejar un sistema en producción que facilitó el proceso de trabajo en un área específica, así como le pudo brindar a la organización elementos que le ayuden a la micro y macro gestión de sus Unidades Ejecutoras.

Según el plan de implantación previsto, se estima que para fines de Noviembre de 2013, todo Egreso Hospitalario de las Unidades Ejecutoras que informan a ASSE, va a ser registrado en el Sistema de Codificación de Egresos.

6.2 TRABAJOS FUTUROS

A continuación se presentan posibles líneas de trabajos a futuro interesantes que no formaron parte del alcance del proyecto:

Mejoras del modelo de datos del SCE: Como se mencionó en la sección 3.5, por cuestiones de simplicidad para los datos del paciente (guardados en el egreso) que se obtienen del Padrón de Usuarios de ASSE se definieron los mismos tipos de datos que devuelve este sistema. Una posible mejora es optimizar los tipos de datos. Por ejemplo para los atributos correspondientes al domicilio de un paciente se podrían utilizar tipos de datos geográficos para luego ser georeferenciados.

Mantenimiento de dimensiones en el Data Warehouse: En la implementación del Data Warehouse no se tuvieron en cuenta los posibles cambios sobre las dimensiones. El ejemplo más claro es en la dimensión *Unidad Ejecutora*, ya que en el futuro pueden agregarse nuevas, fusionarse dos unidades, etc. Por este motivo una posible mejora sería modificar el proceso ETL (utilizar versionado sobre la dimensión) para que el mismo sea más robusto ante posibles modificaciones.

Creación de reportes utilizando las herramientas de Pentaho: Para mejorar la gestión de los egresos hospitalarios**,** en el futuro se pueden crear reportes utilizando la herramienta ofrecida por Pentaho llamada Pentaho Report Design que permite diseñar reportes sobre los cubos del DW.

Mantenimiento de las tablas de la base de datos "*codigos*": Estaba fuera del alcance del proyecto el mantenimiento de estas tablas. Sería de gran utilidad contar con una aplicación para facilitar las altas, bajas y modificaciones de entidades como Unidades Ejecutoras, Médicos, etc.

Actualización WaveMaker: La versión de WaveMaker que se utilizó para el desarrollo del SCE fue la 6.4.5. Al momento del cierre del proyecto ya se contaba con nuevas versiones de este framework. Una línea de trabajo a futuro sería investigar las mejoras de las nuevas versiones para ver los posibles beneficios sobre el sistema desarrollado

6.3 GESTIÓN DEL PROYECTO

Para este proyecto fue definida una metodología de prototipado rápido que permitió alcanzar los objetivos planteados.

En primer lugar se estudió el estado del arte de herramientas/IDES web y se analizaron los requerimientos. Estas tareas se realizaron en paralelo, como se puede ver en la Tabla 4. El estado del arte comprendió la investigación de las herramientas existentes para cumplir con los objetivos mencionados anteriormente. En particular se estudiaron herramientas RAD similares a una que se tomó como punto de comparación y que contaran con interfaz web. El relevamiento de requerimientos se realizó con el responsable del proyecto por parte de ASSE mediante reuniones quincenales en las cuales se determinaron las necesidades de este organismo y a partir de éstas se especificaron los requerimientos.

Teniendo en cuenta los resultados obtenidos en ambas tareas se seleccionó la herramienta a ser utilizada para la elaboración del producto.

Luego de una etapa de aprendizaje de éste framework se comenzó a desarrollar el prototipo, labor ésta que insumió seis meses de trabajo. Es necesario destacar que se buscó que la herramienta de referencia permitiera el prototipado rápido que permitiera realizar validaciones en etapas tempranas. Se decidió implementar en primer instancia la funcionalidad principal, pues la misma tenía gran impacto en el diseño de la interfaz gráfica y en la arquitectura del sistema. De forma incremental se fueron desarrollando y validando todas las funcionalidades hasta llegar al producto final.

En esta etapa de construcción del prototipo del Sistema, también se avanzó de forma paralela en la elaboración de un repositorio de datos común, así como en desarrollar los servicios web necesarios para poder consumir el mismo. Los detalles de esta tarea fueron descritos con más profundidad en los Capítulos 3 y 4.

Sobre el final de la etapa de desarrollo se hicieron pruebas funcionales y no funcionales en el ambiente de testing de ASSE. Éstas fueron detalladas en el Capítulo 4.

Además de las validaciones que se hicieron con el responsable por parte de ASSE, presentamos el producto a usuarios finales de las distintas Unidades Ejecutoras, participando al menos una persona de cada UE. A raíz de estas presentaciones surgió la necesidad de agregar una funcionalidad y realizar cambios menores al prototipo. Por este motivo, el desarrollo del producto está divido en dos instancias como se puede observar en la Tabla 4.

Cumpliendo con la metodología de trabajo establecida por ASSE para la puesta en producción, todo sistema que se implanta debe pasar necesariamente por tres ambientes:

1) Testing.

2) Pre-producción

3) Producción.

La primera etapa involucró exclusivamente técnicos de ASSE , mientras que, para la segunda y tercera etapa la responsabilidad recayó en técnicos de una empresa tercerizada. La participación de terceros provocó que la tarea de implantación llevara más tiempo de lo programado.

Durante la implantación del sistema se comenzó con la integración del Data Warehouse. La metodología para su desarrollo fue similar a la utilizada durante el análisis de requerimientos, realizándose varias reuniones con el responsable por parte de ASSE en las que se definió qué información sería almacenada en el Data Warehouse.

Una vez finalizado el producto ASSE seleccionó cuatro Unidades Ejecutoras de Montevideo, Maldonado, Salto y Paysandú que comenzaron a trabajar en un plan piloto. Tras comprobar la efectividad del sistema se fueron sumando paulatinamente más Unidades Ejecutoras con la intención de llegar a su total implantación en todo el país.

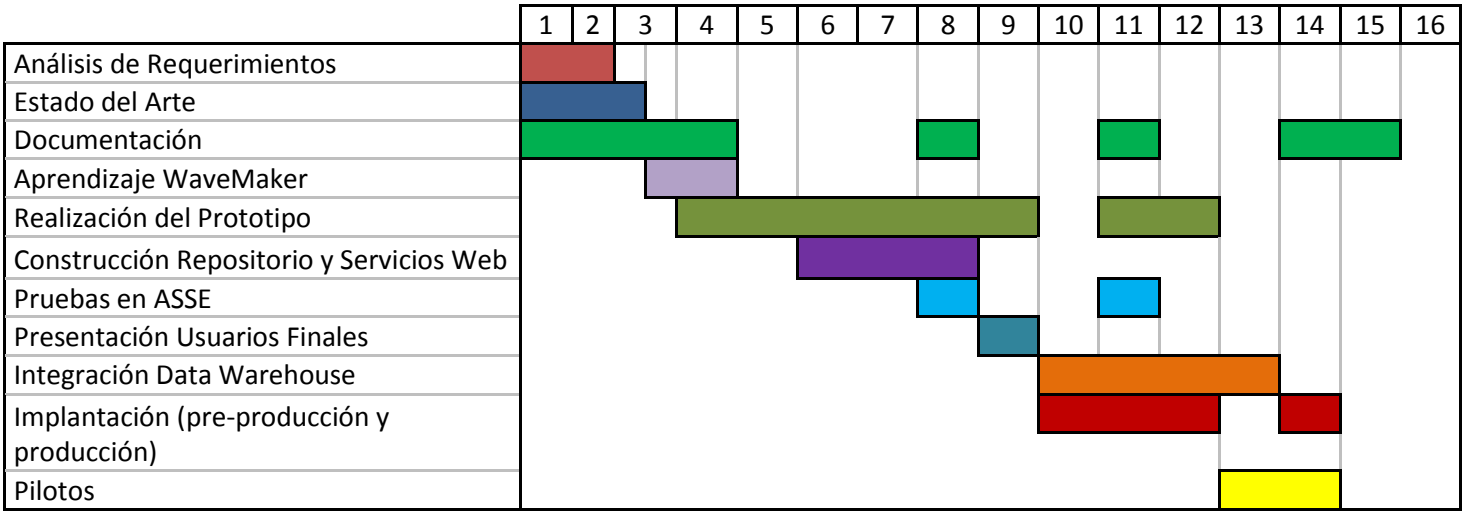


Tabla 4: Diagrama de Gantt del Proyecto

GLOSARIO

ABM: Altas, Bajas y Modificaciones

AJAX: Asynchronous JavaScript And XML (JavaScript asíncrono y XML)

APEX: Oracle Application Express

API: Application Programming Interface (Interfaz de programación de aplicaciones)

ASSE: Administración de los Servicios de Salud del Estado

BI: Business Intelligence

CIE9: Clasificación internacional de enfermedades, novena versión

CIE10: Clasificación internacional de enfermedades, décima versión

CRUD: Create, Read, Update and Delete (Crear, Obtener, Actualizar y Borrar)

DW: Data Warehouse

ETL: Extraction, Transformation, Load (Extracción, Transformación y Carga)

IDE: Integrated Development Environment (Entorno de Desarrollo Integrado)

LDAP: Lightweight Directory Access Protocol (Protocolo Ligero de Acceso a Directorios)

MVC: Modelo Vista Controlador

POJO: Plain Old Java Object

RAD: Rapid Application Development (desarrollo rápido de aplicaciones)

SCE: Sistema de Codificación de Egresos

SOAP: Simple Object Access Protocol

SPSS 2: Statistical Package for the Social Sciences

SVN: Subversion

UE: Unidad Ejecutora

WSDL: Web Services Description Language (Lenguaje de Descripción de Servicios Web)

REFERENCIAS

[1] «IBM - SPSS Software,» . http://www-01.ibm.com/software/analytics/spss/.

[2] «Parlamento del Uruguay,» . http://www.parlamento.gub.uy/repartidos/AccesoRepartidos.asp?Url=/repartidos/camara/d2012120211-01.htm.

[3] «World Wide Web Consortium,» . http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#whatis. [Último acceso: 27 06 2013].

[4] Esteban Revello, Germain Venzal, Nicolás Gerolami, «Implantación de Data Warehouse Open Free,» 2011. . http://www.fing.edu.uy/~asabigue/prgrado/2010dw.pdf.

[5] «Diseño y Construcción de DW, Fing, UdeLaR,» . http://www.fing.edu.uy/inco/cursos/disDW. [Último acceso: 27 06 2013].

[6] «JSON,» . www.json.org. [Último acceso: 27 06 2013].

[7] «The Dojo Foundation,» . www.dojotoolkit.org. [Último acceso: 27 06 2013].

[8] «Spring Framework,» . http://dev.wavemaker.com/wiki/bin/Spring. [Último acceso: 27 06 2013].

[9] «WaveMaker Community,» . http://dev.wavemaker.com/forums/?q=node/1916. [Último acceso: 27 06 2013].

[10] «Apache Tomcat,» . http://tomcat.apache.org/. [Último acceso: 27 06 2013].

[11] «Source Forge,» . http://sourceforge.net/projects/wavemakervisual/files/6.4.5%20Downloads/. [Último acceso: 27 06 2013].

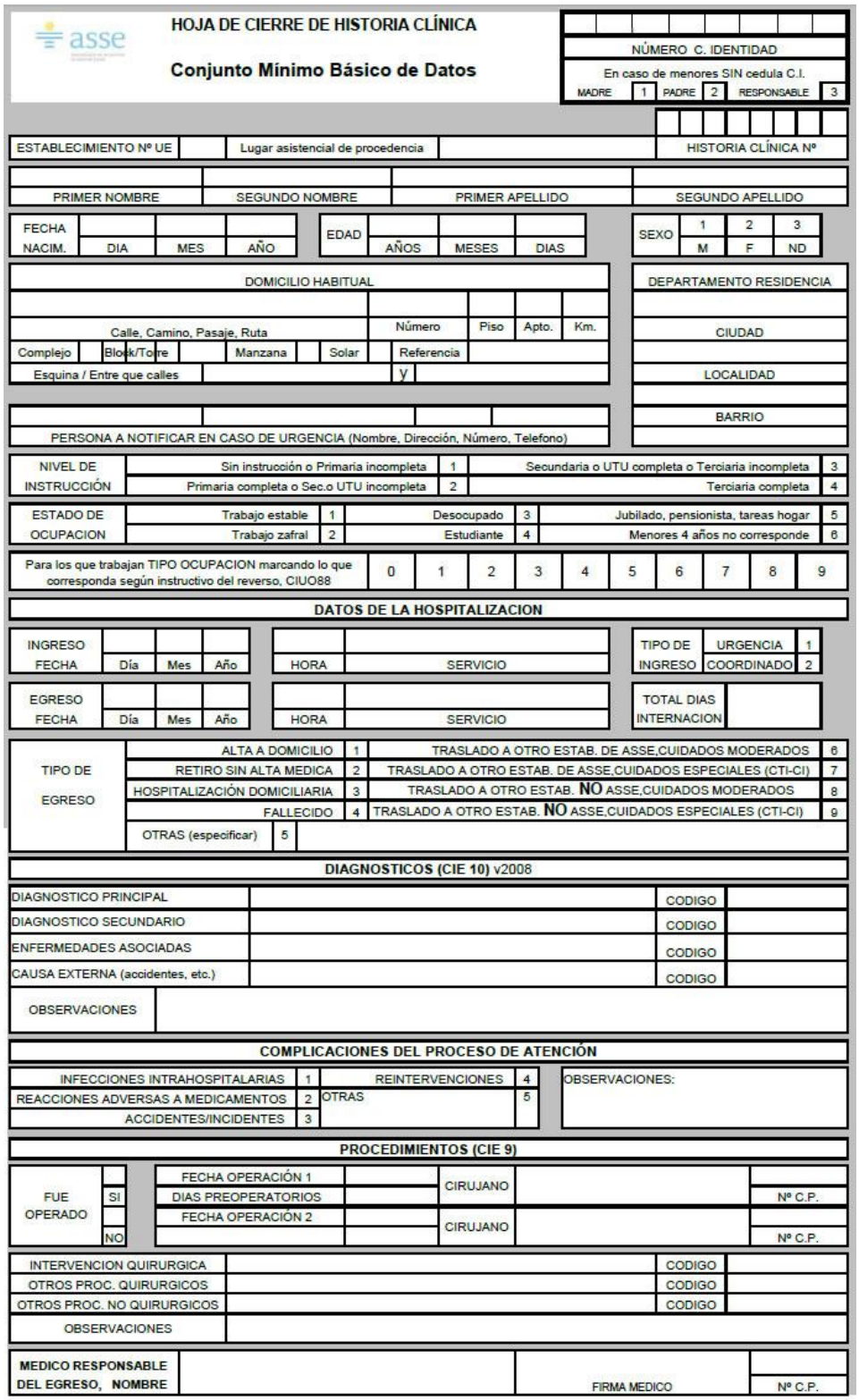
[12] «Jaspersoft Community,» . http://community.jaspersoft.com/. [Último acceso: 27 06 2013].

[13] «Jaspersoft Community - iReport,» . http://community.jaspersoft.com/wiki/ireport-501-release-notes. [Último acceso: 27 06 2013].

[14] «Apache JMeter,» . http://jmeter.apache.org/. [Último acceso: 27 06 2013].

ANEXO 1: HOJA DE CIERRE DE HISTORIA CLÍNICA

En este anexo se presenta la hoja de cierre de historia clínica, la misma es completa por los médicos al momento del egreso de un paciente y luego codificada por los encargados de registros médicos de cada Unidad Ejecutora.



HOJA DE CIERRE DE HISTORIA CLÍNICA

Conjunto Mínimo Básico de Datos

NÚMERO C. IDENTIDAD

En caso de menores SIN cedula C.I.

MADRE 1 | PADRE 2 | RESPONSABLE 3

ESTABLECIMIENTO Nº UE | Lugar asistencial de procedencia | HISTORIA CLÍNICA Nº

PRIMER NOMBRE | SEGUNDO NOMBRE | PRIMER APELLIDO | SEGUNDO APELLIDO

FECHA NACIM. DIA MES AÑO | EDAD AÑOS MESES DIAS | SEXO 1 M 2 F 3 ND

DOMICILIO HABITUAL

Calle, Camino, Pasaje, Ruta | Número | Piso | Apto. | Km.

Complejo | Block/Torre | Manzana | Solar | Referencia

Esquina / Entre que calles | y

PERSONA A NOTIFICAR EN CASO DE URGENCIA (Nombre, Dirección, Número, Telefono)

DEPARTAMENTO RESIDENCIA | CIUDAD | LOCALIDAD | BARRIO

NIVEL DE INSTRUCCIÓN: Sin instrucción o Primaria incompleta 1 | Primaria completa o Sec.o UTU incompleta 2 | Secundaria o UTU completa o Terciaria incompleta 3 | Terciaria completa 4

ESTADO DE OCUPACION: Trabajo estable 1 | Trabajo zafral 2 | Desocupado 3 | Estudiante 4 | Jubilado, pensionista, tareas hogar 5 | Menores 4 años no corresponde 6

Para los que trabajan TIPO OCUPACION marcando lo que corresponda según instructivo del reverso, CIUO88: 0 1 2 3 4 5 6 7 8 9

DATOS DE LA HOSPITALIZACION

INGRESO FECHA Día Mes Año | HORA | SERVICIO | TIPO DE INGRESO: URGENCIA 1, COORDINADO 2

EGRESO FECHA Día Mes Año | HORA | SERVICIO | TOTAL DIAS INTERNACION

TIPO DE EGRESO:

- ALTA A DOMICILIO 1
- RETIRO SIN ALTA MEDICA 2
- HOSPITALIZACIÓN DOMICILIARIA 3
- FALLECIDO 4
- OTRAS (especificar) 5
- TRASLADO A OTRO ESTAB. DE ASSE,CUIDADOS MODERADOS 6
- TRASLADO A OTRO ESTAB. DE ASSE,CUIDADOS ESPECIALES (CTI-CI) 7
- TRASLADO A OTRO ESTAB. **NO** ASSE,CUIDADOS MODERADOS 8
- TRASLADO A OTRO ESTAB. **NO** ASSE,CUIDADOS ESPECIALES (CTI-CI) 9

DIAGNOSTICOS (CIE 10) v2008

DIAGNOSTICO PRINCIPAL		CODIGO	
DIAGNOSTICO SECUNDARIO		CODIGO	
ENFERMEDADES ASOCIADAS		CODIGO	
CAUSA EXTERNA (accidentes, etc.)		CODIGO	
OBSERVACIONES			

COMPLICACIONES DEL PROCESO DE ATENCIÓN

INFECCIONES INTRAHOSPITALARIAS	1	REINTERVENCIONES	4	OBSERVACIONES:
REACCIONES ADVERSAS A MEDICAMENTOS	2	OTRAS	5	
ACCIDENTES/INCIDENTES	3			

PROCEDIMIENTOS (CIE 9)

FUE OPERADO: SI / NO

FECHA OPERACIÓN 1		CIRUJANO		Nº C.P.
DIAS PREOPERATORIOS				
FECHA OPERACIÓN 2		CIRUJANO		Nº C.P.

INTERVENCION QUIRURGICA		CODIGO	
OTROS PROC. QUIRURGICOS		CODIGO	
OTROS PROC. NO QUIRURGICOS		CODIGO	
OBSERVACIONES			

MEDICO RESPONSABLE DEL EGRESO, NOMBRE | FIRMA MEDICO | Nº C.P.

ANEXO 2: FUNCIONALIDADES DEL SISTEMA DE CODIFICACIÓN DE EGRESOS

En este anexo se presentan las funcionalidades del Sistema de Codificación de Egresos.

Nombre funcionalidad	Alta de Usuario
Sinopsis	Se da de alta a un usuario nuevo en el sistema asignándole rol y unidad ejecutora.
Actores	Administrador del sistema
Pre-Condiciones	
PRE1	El usuario debe haber iniciado sesión
PRE2	El usuario a crear debe existir en el LDAP del correo de ASSE.
Post-Condiciones	
POS1	Existe un nuevo usuario en el sistema.
Descripción	
Se ingresa a la funcionalidad Usuarios y se selecciona añadir. Se despliegan los campos para ingresar nombre, rol, en caso de que corresponda unidad ejecutora y observaciones, todos los campos son obligatorios salvo observaciones. En el campo nombre se debe de ingresar un usuario válido del correo de ASSE, en caso contrario se creará el usuario pero el mismo no va a poder iniciar sesión en el sistema. Los roles posibles son son ADMIN y REGISTROS MÉDICOS, la unidad Ejecutora se completa solo para los usuarios con rol REGISTROS MÉDICOS. Se brinda la opción de Guardar o Cancelar, en caso de cancelar vuelve a la pantalla principal de la funcionalidad Usuarios sin crear al usuario y en caso de Guardar se crea el usuario con los datos ingresados. La contraseña del usuario será la misma que la de su correo.	

Nombre funcionalidad	Ver/Modificar usuario
Sinopsis	Se ven/modifican datos asociados a un usuario.

Actores	Administrador del sistema.
Pre-Condiciones	
PRE1	El usuario debe haber iniciado sesión
PRE2	Debe existir un usuario en el sistema
Post-Condiciones	
POS1	Se ven/modifican los datos de un usuario.
Descripción	
Se ingresa a la funcionalidad Usuarios, se ingresa el nombre de usuario que se quiere buscar y se selecciona el botón Buscar. El campo para el ingreso del nombre no es obligatorio, en caso de presionar Buscar dejando el campo vacío se muestran en la tabla todos los usuarios que existen en el sistema y en caso contrario se muestran los usuarios que coincidan con la búsqueda. Se puede buscar un usuario ingresando parte del comienzo del nombre. Se despliega en la tabla los usuarios que coinciden con la búsqueda. En caso de querer consultar los datos del usuario se debe hacer doble click sobre la fila correspondiente, luego se muestra un diálogo con el Nombre de Usuario, Rol, Usuario Activo, Observaciones y en caso de que el Rol no sea Admin se muestra la Unidad Ejecutora. En caso de querer modificar los datos se debe presionar el botón Modificar que se encuentra en la fila, luego se muestra un diálogo con el Nombre de Usuario, Rol, Usuario Activo, Observaciones y en caso de que el Rol no sea Admin se muestra la Unidad Ejecutora. Se permite modificar todos los datos excepto el nombre de usuario. En caso de poner en el campo Usuario activo el valor No, el usuario quedará inactivo por lo que no podrá realizar acciones en el sistema. Se brindan las opciones de Cancelar o Guardar, en caso de cancelar la modificación queda sin efecto y en caso de Guardar se modifican los datos del usuario.	

Nombre funcionalidad	Inicio de sesión de usuario
Sinopsis	Un usuario inicia sesión en el sistema.
Actores	Administrador del sistema / Registros médicos
Pre-Condiciones	

PRE1	El usuario debe existir en el sistema.
Post-Condiciones	
POS1	El usuario accede al sistema.
Descripción	
El usuario inicia sesión ingresando usuario y contraseña del LDAP del correo de ASSE. En caso de que el usuario tenga el Rol REGISTROS MEDICOS se le brindan las opciones de Registrar Egreso, Ver / Modificar Egreso y Auditoría de Egresos, en caso de que el rol sea ADMIN a parte de las opciones mencionadas anteriormente también se le brinda la opción de Usuarios.	

Nombre funcionalidad	Registrar Egreso
Sinopsis	Se da de alta un egreso en el sistema con los datos correspondientes.
Actores	Administrador del sistema / Registros médicos
Pre-Condiciones	
PRE1	El usuario debe haber iniciado sesión
Post-Condiciones	
POS1	Existe un nuevo egreso en el sistema.
Descripción	
Se ingresa a la funcionalidad Registrar Egreso, se despliegan los siguientes campos para su ingreso: - Establecimiento N° UE: solo lo debe completar los usuarios con rol ADMIN, para los usuarios con rol REGISTROS MÉDICOS se completa automáticamente de la Unidad Ejecutora que tiene asignada el usuario. El campo es obligatorio. - Fecha de Ingreso Hospitalario: Debe ser menor o igual a la fecha de ingreso del nivel de cuidado. - Fecha de Egreso Hospitalario: Debe ser mayor que la fecha de ingreso hospitalario y mayor a la fecha de egreso del nivel de cuidado. - Lugar asistencial de procedencia: Se da la posibilidad de seleccionar lugar interno a ASSE, externo a ASSE u otros, en caso de seleccionar interno a ASSE o externo a ASSE se muestra una lista con	

los que están cargados en el sistema para cada caso, en caso de seleccionar Otros se muestra un campo para el ingreso de texto. El campo Lugar asistencial de procedencia no es obligatorio.

- Documento: Se debe ingresar el documento del paciente y seleccionar buscar, en caso de que no se encuentre un paciente que coincida con la búsqueda se despliega un mensaje indicando lo sucedido. Los pacientes se van a buscar al padrón de usuarios de ASSE. El campo es obligatorio.

- Los campos Primer nombre, Segundo nombre, Primer apellido, Segundo apellido, Fecha nacimiento, Sexo, Depto residencia, Localidad, Domicilio, Numero, Apto, Entre y Estado del padrón, se completan automáticamente a partir del paciente buscado trayendo los datos del padrón de usuarios de ASSE.

- Edad: Se calcula automáticamente a partir de la fecha de egreso y la fecha de nacimiento del paciente luego de que estos campos sean completados.

- Nivel de instrucción: Se despliega una lista con los posibles valores que puede tomar el campo. No es obligatorio completar este dato.

- Estado de ocupación: Se despliega una lista con los posibles valores que puede tomar el campo. No es obligatorio la completar este campo.

- Para los que trabajan TIPO OCUPACIÓN según CIU088: Se despliega una lista con los posibles valores que puede tomar el campo. No es obligatorio la completar este dato.

- Fecha de ingreso: La fecha debe de ser anterior a la fecha actual y posterior a 3 meses para atrás. No se pueden crear egresos con más de 3 meses de antigüedad. Debe ser mayor a la fecha de ingreso hospitalario. El campo es obligatorio.

- Fecha de egreso: La fecha debe de ser anterior a la fecha actual y posterior a la fecha de ingreso. Debe ser menor a la fecha de egreso hospitalario. El campo es obligatorio.

- Servicio Ingreso: Se da la posibilidad de seleccionar un servicio según la clasificación del SINADI. Primero se debe seleccionar si es Básica, Médica o Quirúrgica, y luego según lo elegido se despliega una lista con todas los posibles. El campo no es obligatorio.

- Servicio egreso: Se da la posibilidad de seleccionar un servicio según la clasificación del SINADI. Primero se debe seleccionar si es Básica, Médica o Quirúrgica, y luego según lo elegido se despliega una lista con todas los posibles. El campo es obligatorio.

- Tipo de Ingreso: Se da una lista con los posibles valores para ingresar. El campo es obligatorio.

- Tipo de Egreso: Se da una lista con los posibles valores para ingresar. El campo es obligatorio.

- Nivel Cuidados: Se da una lista con los posibles valores para ingresar. El campo es obligatorio.

- Total días internación: Se calcula automáticamente a partir de la fecha de egreso y la fecha de ingreso. Se redondea para arriba, por ejemplo si el total de horas entre ambas fechas es 25 entonces la cantidad de días de internación son 2.

- Diagnósticos: Se brinda la posibilidad de ingresar el código CIE 10 correspondiente al diagnóstico o buscarlo a partir de por lo menos los dos primeros caracteres del código. Luego de ingresar el código se carga automáticamente su descripción. El diagnóstico principal es obligatorio y único, los diagnósticos secundarios o causas externas no son obligatorios y se pueden ingresar varios. Se chequea que:

- No puede estar dos veces el mimo diagnóstico.

- En caso de ingresar un diagnóstico principal que comience con S o T es obligatorio que exista una

causa externa que comience con V, W, X o Y.

- Si el diagnóstico principal está entre O80 y O84 o entre O03 y O06, debe existir algún procedimiento de cualquier tipo.

- Si la persona es de sexo M no pueden existir diagnosticos (prin., sec., ce) que comiencen con O.

- Complicaciones del proceso de atención: Se despliegan todos los posibles valores dejando seleccionar varios o ninguno.

- Procedimientos: Se debe seleccionar si el paciente tuvo algún procedimiento o no (obligatorio), en caso de que haya tenido alguno se debe ingresar por lo menos un procedimiento CIE9. Se brinda la posibilidad de ingresar los procedimientos a partir del código CIE 9 o buscarlo a partir de por lo menos los dos primeros caracteres del código. Luego de ingresar el código se carga automáticamente su descripción. Para cada código ingresado se debe detallar que tipo de procedimiento es, si una intervención quirúrgica, otros procedimientos quirúrgicos o otros procedimientos no quirúrgicos.

- Médico responsable del egreso: Se busca al médico a partir del número de caja profesional. Se carga automáticamente los datos del médico buscado, nombres, apellidos y cédula de identidad. No es obligatorio ingresar un médico responsable.

- Observaciones del codificador: es un campo texto para que se ingresen todos los comentarios que se quiera. No es obligatorio.

Se brinda la posibilidad de Cancelar o Guardar, en caso de Cancelar vuelve al menú principal perdiendo todos los datos que fueron ingresados, en caso de Guardar se despliega un mensaje de confirmación que al aceptar guarda el egreso con los datos ingresados quedando en la misma pantalla con todos los campos vacíos para ingresar un nuevo egreso.

Todos los campos que son obligatorios están marcados con un * de color rojo al lado del nombre del campo.

Los valores posibles para los campos son los que están al dorso de la hoja de cierre de historia clínica.

No se puede guardar un egreso si no cumple todos los egresos o si falta completar datos en algún campo obligatorio, en caso que se quiera guardar se muestra un mensaje indicando lo sucedido.

No se pueden dar de alta egresos que tengan más de 3 meses de antigüedad ni egresos con fecha de egreso posterior a la fecha actual.

Nombre funcionalidad	Ver/Modificar Egreso
Sinopsis	Se ven o modifican los datos de un egreso existente en el sistema
Actores	Administrador del sistema / Registros médicos

Pre-Condiciones	
PRE1	El usuario debe haber iniciado sesión
PRE2	Debe existir un egreso en el sistema.
Post-Condiciones	
POS1	Se ven/modifican los datos de un egreso

Descripción

Se ingresa a la funcionalidad Ver/Modificar Egreso. Se brinda la posibilidad de buscar el egreso por documento del paciente y fecha de egreso o solo por documento del paciente.

Luego de haber ingresado los datos del egreso y seleccionar buscar se despliega en una tabla todos los egresos que coincidan con la búsqueda ingresada. Para cada egreso se muestran los datos del paciente, documento, nombre y apellido, unidad ejecutora del egreso, fecha de egreso, nivel de cuidado, servicio del egreso y un botón para modificar. Un usuario con rol Registros médicos puede ver todos los egresos que hay en el sistema pero solo puede modificar los que fueron creados en la misma Unidad Ejecutora que tiene asignada el usuario y que no tengan más de tres meses de antigüedad. Un usuario con rol Administrador puede ver y modificar todos los egresos.

En caso de querer ver los datos de un egreso se debe hacer doble click sobre la fila correspondiente, luego se muestra la misma pantalla de alta egreso pero con todos los datos cargados y los campos deshabilitados para no poder modificar los mismos. Se brinda la posibilidad de Volver donde se vuelve a la pantalla anterior.

En caso de querer modificar se debe presionar el botón modificar que se encuentra en la fila correspondiente al egreso, luego se muestra la misma pantalla de alta egreso con datos del egreso seleccionado ya cargados. Se permite modificar todos los datos salvo el documento del paciente. Los chequeos de los campos son los mismos que en la funcionalidad Registrar Egreso. Se brinda la opción de Cancelar o Guardar, en caso de Cancelar vuelve a la pantalla anterior sin modificar los datos y en caso de Guardar vuelve a la pantalla anterior guardando las modificaciones.

Nombre funcionalidad	Auditoría de Egreso
Sinopsis	Se auditan las modificaciones que tuvo un egreso.
Actores	Administrador del sistema / Registros médicos

Pre-Condiciones	
PRE1	El usuario debe haber iniciado sesión
PRE2	Debe existir un egreso en el sistema.
Post-Condiciones	
POS1	Se ven las modificaciones que tuvo un egreso.
Descripción	
Se ingresa a la funcionalidad Auditoría de Egreso. Se brinda la posibilidad de buscar el egreso por documento del paciente y fecha de egreso o solo por documento del paciente. Luego de haber ingresado los datos del egreso y seleccionar buscar se despliega en una tabla todos los egresos que coincidan con la búsqueda ingresada, para cada egreso se muestran todas las modificaciones que tuvo incluyendo la creación que sería la primer modificación que aparece en la tabla. Las modificaciones se muestran ordenadas por fecha. Para cada modificación se muestra Fecha Modificación, Usuario (usuario que realizó la modificación), Fecha Egreso, Paciente (paciente del egreso) y Unidad Ejecutora (Unida Ejecutora del egreso). Para un mismo egreso los tres últimos datos coinciden.	

Nombre funcionalidad	Generar Informe
Sinopsis	Se genera un informe con la cantidad de egresos por Cédula de Identidad de médico para una unidad ejecutora entre un rango de fechas.
Actores	Administrador del sistema / Registros médicos
Pre-Condiciones	
PRE1	El usuario debe haber iniciado sesión
Post-Condiciones	

POS1	Se genera un PDF con el informe
Descripción	
Se ingresa a la funcionalidad Informe. Se selecciona el rango de fechas que se desea. Un usuario con rol administrador debe seleccionar una Unidad Ejecutora. En caso de que el rol sea Registros Médicos el informe contendrá información sobre los egresos de su Unidad Ejecutora.	

ANEXO 3: TABLAS DE LA BASE DE DATOS

- **Tabla de Egresos**

Nombre: Egresos
Descripción: Almacena la información relativa a los egresos de los pacientes; es decir del alta del paciente (a domicilio, contra voluntad médica, fuga), el fallecimiento, o el traslado a otro Hospital o Institución Médica para completar su internación.

Los datos del paciente se obtienen del padrón de usuarios de ASSE y se guardan junto con el egreso y no serán modificados a través de la aplicación. Por razones de simplicidad se mantuvieron los mismos tipos de datos para los campos obtenidos del padrón.

ATRIBUTO	DESCRIPCIÓN
Id	Identificador del Egreso (primary key) Tipo: int(11)
Ci	Cédula de Identidad del paciente asociado al Egreso Tipo: varchar(255) NOT NULL
fechaIngresoHospitalario	Fecha de Ingreso Hospitalario del paciente Tipo: Date
fechaEgresoHospitalario	Fecha de Egreso Hospitalario del paciente Tipo: Date
fechaIngreso	Fecha de Ingreso del paciente en el momento que comienza a ocupar una cama de dotación del hospital. Tipo: datetime NOT NULL
servicioIngreso	Identificador del servicio del ingreso. Foreign key a la tabla 'tipoServicio' Tipo: int(11)
tipoIngreso	Identificador del tipo de ingreso. Foreign key a la tabla 'tipoIngreso'. Tipo: int(11) NOT NULL
fechaEgreso	Fecha de Egreso del paciente Tipo: datetime NOT NULL
servicioEgreso	Identificador del servicio del ingreso. Foreign key a la tabla 'tipoServicio'

	Tipo: int(11) NOT NULL
tipoEgreso	Identificador del Tipo de Egreso. Foreign key a la tabla 'tipoEgreso' Tipo: int(11) NOT NULL
nroUE	Identificador de la Unidad del cual egresa el paciente. Tipo: int(11) NOT NULL
NombreUE	Nombre de la Unidad del cual egresa el paciente. Tipo: varchar(255) NOT NULL
lugarProcedencia	Es el Hospital, Centro Auxiliar, Centro de Salud, Mutualista o Institución, al que está adscrito el paciente. Tipo: varchar(255)
nivelInstruccionPaciente	Identificador del tipo de nivel de Instrucción del paciente asociado al egreso. Foreign key a la tabla nivel 'nivelInstruccion'. Tipo: int(11)
estadoOcupacionPaciente	Identificador del tipo de estado de ocupación del paciente asociado al egreso. Tipo: int(11)
tipoOcupacionPaciente	Identificador del tipo de ocupación del paciente asociado al egreso. Foreign key a la tabla 'tipoOcupacion'. Tipo: int(11)
fueOperado	Indica si el paciente fue operado o no. Tipo: tinyint(1) Vale 0 en caso de que no haya sido operado y 1 si lo fue. NOT NULL
nroCajaProfMedico	Nº de Caja Profesional del médico responsable del llenado de la hoja de cierre de historia clínica. Tipo: int(11)
ciMedico	Cédula de Identidad del médico responsable del llenado de la hoja de cierre de historia clínica. Tipo: varchar(8)

nivelCuidados	Identificador del nivel de cuidados. Foreign key a la tabla 'nivelCuidados' Tipo: int(11) NOT NULL
primerNombrePac	Primer nombre del paciente asociado al egreso. Tipo: varchar(255)
segundoNombrePac	Segundo nombre del paciente asociado al egreso. Tipo: varchar(255)
primerApellidoPac	Primer apellido del paciente asociado al egreso. Tipo: varchar(255)
segundoApellidoPac	Segundo apellido del paciente asociado al egreso. Tipo: varchar(255)
fechaNacPac	Fecha de nacimiento del paciente asociado al egreso. Tipo: date
sexoPac	Sexo del paciente asociado al egreso. Tipo: varchar(255) Valores posibles: F, M ND
deptoPac	Departamento de residencia del paciente asociado al egreso. Tipo: varchar(255)
localidadPac	Localidad de residencia del paciente asociado al egreso. Tipo: varchar(255)
domicilioPac	Domicilio de residencia del paciente asociado al egreso. Tipo: varchar(255)
nroPuertaPac	Número de puerta de la residencia del paciente asociado al egreso. Tipo: varchar(255)
aptoPac	Número de apartamento de residencia del paciente asociado al egreso. Tipo: varchar(255)
entreCalle1	Esquina 1 del domicilio del paciente asociado al egreso Tipo: varchar(255)
entreCalle2	Esquina 2 del domicilio del paciente asociado al egreso Tipo: varchar(255)

diasInternacion	Número de días de internación del paciente. Tipo: int(11) NOT NULL
estadoPadron	Estado del paciente en el padrón de ASSE Tipo: varchar(255)
fechaAlta	Fecha de creación del Egreso en el sistema Tipo: datetime NOT NULL
observaciones	Observaciones del egreso Tipo: Text

- **Tabla de Tipo de Egreso**

Nombre: tipoEgreso
Descripción: Almacena los distintos tipos de egresos.

ATRIBUTO	DESCRIPCIÓN
id	Identificador del Tipo de Egreso (primary key). Tipo: int(11) NOT NULL
tipo	Descripción del tipo de egreso. Tipo: varchar(255) NOT NULL

- **Tabla de Tipo de Ingreso**

Nombre: tipoEgreso
Descripción: Almacena los distintos tipos de ingresos.

ATRIBUTO	DESCRIPCIÓN
id	Identificador del Tipo de Ingreso. (primary key) Tipo: int(11) NOT NULL

Descripción	Descripción del tipo de ingreso. Tipo: varchar(255) NOT NULL

- **Tabla de Clasificación de Servicios**

Nombre: áreaservicio
Descripción: Almacena las distintas clasificaciones de los servicios.

ATRIBUTO	DESCRIPCIÓN
id	Identificador de la clasificación de los servicios.(primary key) Tipo: int(11) NOT NULL
Nombre	Nombre de la clasificación de los servicios. Tipo: varchar(255) NOT NULL

- **Tabla de Tipo de Servicio**

Nombre: tiposervicio
Descripción: Almacena los distintos tipos de servicios.

ATRIBUTO	DESCRIPCIÓN
id	Identificador del Tipo de Ingreso. (primary key) Tipo: int(11) NOT NULL
idAreaServicio	Identificador de la clasificación a la cual pertenece el tipo de servicio. Foreign key a la tabla 'areaservicio'. Tipo: int(11) NOT NULL
Nombre	Nombre del tipo de servicio. Tipo: varchar(255) NOT NULL

- **Tabla de Tipo de Ocupación**

Nombre: tipoOcupacion
Descripción: Almacena los tipos de ocupación posibles para un paciente (clasificación de ocupación según CIUO88).

ATRIBUTO	DESCRIPCIÓN
Id	Identificador del Tipo de Ocupación. (primary key) Tipo: int(11) NOT NULL
Desc	Descripción del tipo de ocupación. Tipo: varchar(255) NOT NULL

- **Tabla de Estado de Ocupación**

Nombre: estadoOcupacion
Descripción: Almacena los estados de ocupación posibles para un paciente.

ATRIBUTO	DESCRIPCIÓN
id	Identificador del Estado de Ocupación. (primary key) Tipo: int(11) NOT NULL
descripción	Descripción del estado de ocupación. Tipo: varchar(255) NOT NULL

- **Tabla de Nivel de Instrucción de un paciente**

Nombre: nivelInstruccion
Descripción: Almacena los niveles de instrucción posibles para un paciente.

ATRIBUTO	DESCRIPCIÓN
id	Identificador del nivel de instrucción. (primary key) Tipo: int(11) NOT NULL
descripción	Descripción del nivel de instrucción Tipo: varchar(255) NOT NULL

- **Tabla de Nivel de Cuidados de un egreso**

Nombre: nivelCuidados
Descripción: Almacena los tipos de nivel de cuidados de un egreso.

ATRIBUTO	DESCRIPCIÓN
id	Identificador del nivel de cuidados. (primary key) Tipo: int(11) NOT NULL
descripción	Descripción del nivel de cuidados Tipo: varchar(255) NOT NULL

- **Tabla de Complicaciones**

Nombre: complicaciones
Descripción: Almacena las posibles complicaciones asociadas al tratamiento recibido por el paciente.

ATRIBUTO	DESCRIPCIÓN
id	Identificador de la complicación. (primary key) Tipo: int(11) NOT NULL
descripción	Descripción de la complicación Tipo: varchar(255) NOT NULL

- **Tabla de Complicaciones de un egreso**

Nombre: complicacionesEgreso
Descripción: Almacena las complicaciones asociadas a un egreso

ATRIBUTO	DESCRIPCIÓN
id	Identificador de la tupla Egreso-Complicación. (primary key) Tipo: int(11) NOT NULL

idEgreso	Id del egreso. Foreign key a la tabla 'egreso'. Tipo: int(11) NOT NULL
idComplicacion	Id de la complicación. Foreign key a la tabla 'complicacion' Tipo: int(11) NOT NULL

- **Tabla de tipos de procedimiento**

Nombre: tipoProcedimiento
Descripción: Almacena los posibles tipos de procedimientos que se realizan al paciente en el momento en que se realiza la intervención quirúrgica principal o dentro del episodio de hospitalización.

ATRIBUTO	DESCRIPCIÓN
id	Identificador del tipo de procedimiento (primary key). Tipo: int(11) NOT NULL
descripción	Descripción del tipo de procedimiento Tipo: varchar(255) NOT NULL

- **Tabla de Procedimientos**

Nombre: procedimiento
Descripción: Almacena los procedimientos asociados al egreso, que se realizaron al paciente en el momento de la intervención quirúrgica principal o dentro del episodio de hospitalización.

ATRIBUTO	DESCRIPCIÓN
id	Identificador de la tupla Egreso-Procedimiento. (primary key) Tipo: int(11) NOT NULL
idEgreso	Id del egreso . Foreign key a la tabla 'egreso' Tipo: int(11) NOT NULL
tipoProcedimiento	id del tipo de procedimiento realizado al paciente. Foreign key a la tabla 'tipoProcedimiento'

	Tipo: int(11) NOT NULL Valores posibles: INTERVENCIÓN QUIRÚRGICA, OTROS PROC. QUIRÚRGICOS, OTROS PROC. NO QUIRÚRGICOS
código	Código CIE 9 correspondiente al procedimiento Tipo: varchar(255) NOT NULL

- **Tabla de tipos de diagnósticos**

Nombre: tipoDiagnostico
Descripción: Almacena los tipos de diagnóstico o afecciones posibles asociadas a pacientes.

ATRIBUTO	DESCRIPCIÓN
id	Identificador del tipo de diagnóstico. (primary key) Tipo: int(11) NOT NULL
nombre	Nombre del diagnóstico Tipo: varchar(255) NOT NULL

- **Tabla de Diagnósticos**

Nombre: diagnostico
Descripción: Almacena los diagnósticos o afecciones asociadas al paciente.

ATRIBUTO	DESCRIPCIÓN
id	Identificador de la tupla Egreso-Procedimiento (primary key) Tipo: int(11) NOT NULL
idEgreso	Id del egreso. Foreign key a la tabla 'egreso' Tipo: int(11) NOT NULL
tipoDiagnostico	id del tipo de diagnóstico. Foreign key a la tabla 'tipodiagnostico' Tipo: int(11)

	NOT NULL Valores posibles: DIAGNÓSTICO PRINCIPAL, DIAGNÓSTICO SECUNDARIO, CAUSA EXTERNA
código	Código CIE 10 correspondiente al diagnóstico Tipo: varchar(255) NOT NULL

- **Tabla de Usuarios**

Nombre: usuarios
Descripción: Almacena los datos de los usuarios del sistema

ATRIBUTO	DESCRIPCIÓN
id	Identificador del usuario (primary key) Tipo: int(11) NOT NULL
nombre	Nombre del usuario (único) Tipo: varchar(255) NOT NULL
rol	Rol del usuario Tipo: varchar(255) Valores posibles: ADMIN, REGISTROS MÉDICOS
unidadEjecutora	Identificador de la Unidad Ejecutora a la que pertenece el usuario. Tipo: int(11)
activo	indica si el usuario está activo o no Tipo: tinyint NOT NULL Valores posibles: 1 (activo), 0 (inactivo)
Observaciones	Observaciones del usuario Tipo: varchar(255)

- **Tabla de Auditoría de egresos**

Nombre: auditadoEgreso
Descripción: Almacena registros de las modificaciones realizadas a un egreso.

ATRIBUTO	DESCRIPCIÓN
id	Identificador de la tupla AuditadoEgreso-Egreso. (primary key) Tipo: int(11) NOT NULL
idEgreso	Id del egreso. Foreign key a la tabla 'egreso' Tipo: int(11) NOT NULL
usuario	nombre del usuario que creo o editó el egreso Tipo: int(11) NOT NULL
fecha	fecha de modificación del egreso Tipo: datetime NOT NULL

ANEXO 4: TABLA DE LA BASE DE DATOS CÓDIGOS

- **Tabla de Pacientes**

Nombre: paciente
Descripción: Almacena la información relativa a los pacientes.

ATRIBUTO	DESCRIPCIÓN
Id	Identificador del Egreso (primary key) Tipo: int(11)
documento	Cédula de Identidad del paciente Tipo: varchar(8) NOT NULL
nombre	Primer nombre del paciente Tipo: varchar(250)
SegundoNombre	Segundo nombre del paciente Tipo: varchar(255)
apellido	Primer apellido del paciente Tipo: varchar(250)
segundoApellido	Segundo apellido del paciente asociado al egreso. Tipo: varchar(255)
FechaNac	Fecha de nacimiento del paciente Tipo: date
Sexo	Sexo del paciente Tipo: varchar(255) Valores posibles: F, M ND
domdepartamento	Departamento de residencia del paciente Tipo: varchar(255)
domlocalidad	Localidad de residencia del paciente Tipo: varchar(255)
domcalle	Domicilio de residencia del paciente Tipo: varchar(255)
domnumero	Número de puerta de la residencia del paciente Tipo: varchar(20)

dompiso	Número de piso de la residencia del paciente Tipo: varchar(20)
domapto	Número de apartamento de residencia del paciente Tipo: varchar(20)
domkm	km de la residencia del paciente Tipo: varchar(20)
domcalle1	Esquina 1 del domicilio del paciente Tipo: varchar(255)
domcalle2	Esquina 2 del domicilio del paciente Tipo: varchar(255)
domcomplejo	Complejo de la residencia del paciente Tipo: varchar(20)
domblock	Block de la residencia del paciente Tipo: varchar(20)
dommanzana	Manzana de la residencia del paciente Tipo: varchar(20)
domsolar	Solar de la residencia del paciente Tipo: varchar(20)
domreferencia	Referencia para la residencia del paciente Tipo: varchar(255)
dombarrio	Barrio de residencia del paciente Tipo: varchar(255)
domciudad	Ciudad de residencia del paciente Tipo: varchar(255)
nivelInstruccionPaciente	Identificador del tipo de nivel de Instrucción del paciente. Tipo: int(11)
estadoOcupacionPaciente	Identificador del tipo de estado de ocupación del paciente. Tipo: int(11)
tipoOcupacionPaciente	Identificador del tipo de ocupación del paciente. Tipo: int(11)

- **Tabla de Médicos**

Nombre: Médicos
Descripción: Almacena la información relativa a los médicos.

nroCajaProf	Número de Caja Profesional del Médico (primary key) Tipo: int(11) NOT NULL
documento	Cédula de identidad del médico Tipo: varchar(8) NOT NULL
nombre	Nombre del médico Tipo: varchar(250) NOT NULL
apellido	Apellido del médico Tipo: varchar(250) NOT NULL

- **Tabla de Unidades Ejecutoras**

Nombre: unidades_ejecutoras
Descripción: Almacena la información relativa a las unidades ejecutoras y a las unidades asistenciales

id	Identificador de la unidad ejecutora o unidad asistencial (primary key) Tipo: int(11) NOT NULL
nombre	Nombre de la unidad ejecutora o unidad asistencial Tipo: varchar(255) NOT NULL
departamento	Nombre del departamento donde se encuentra la unidad ejecutora o unidad asistencial. Tipo: varchar(50)
asistencial	Indica si es una unidad asistencial (1) o si es una unidad ejecutora.

	Tipo: tinyint(1) NOT NULL

- **Tabla de Prestadores de salud externos a ASSE**

Nombre: externos
Descripción: Almacena la información relativa a los prestadores de salud externos asse

nombre	Identificador del prestador de salud (primary key) Tipo: varchar(100) NOT NULL
publico	Indica si el prestador de salud es público (1) o no (0). Tipo: tinyint(1) NOT NULL

- **Tabla de Códigos CIE9**

Nombre: cie9
Descripción: Almacena los códigos cie9, correspondientes a los procedimientos.

cod	Identificador del código (primary key) Tipo: varchar(255) NOT NULL
descrip	Descripción del procedimiento Tipo: varchar(255) NOT NULL

- **Tabla de Códigos CIE10**

Nombre: cie10
Descripción: Almacena los códigos cie10, correspondientes a los diagnósticos.

cod	Identificador del código (primary key) Tipo: varchar(255) NOT NULL

descrip	Descripción del diagnóstico Tipo: varchar(255) NOT NULL

ANEXO 5: DETALLE TÉCNICO DE LAS INTERFACES

Dado el tipo de integración (Web Server) se hace la especificación de la integración al "Repositorio 1" por medio del WSDL que se muestra a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions targetNamespace="http://DefaultNamespace"
xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://DefaultNamespace"
xmlns:intf="http://DefaultNamespace" xmlns:tns1="http://wspackage"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
<wsdl:types>
    <schema elementFormDefault="qualified" targetNamespace="http://DefaultNamespace"
xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://wspackage"/>
   <element name="getCie9">
      <complexType>
      <sequence>
        <element name="Codigo" type="xsd:string"/>
      </sequence>
      </complexType>
   </element>
   <element name="getCie9Response">
      <complexType>
      <sequence>
        <element name="getCie9Return" type="tns1:RegCie"/>
      </sequence>
      </complexType>
   </element>
   <element name="getListaCie9">
      <complexType>
      <sequence>
        <element name="CantReg" type="xsd:int"/>
        <element name="Filtro" type="xsd:string"/>
      </sequence>
      </complexType>
   </element>
   <element name="getListaCie9Response">
      <complexType>
      <sequence>
    <element maxOccurs="unbounded" name="getListaCie9Return" type="tns1:RegCie"/>
      </sequence>
      </complexType>
   </element>
   <element name="getCie10">
      <complexType>
      <sequence>
        <element name="Codigo" type="xsd:string"/>
      </sequence>
      </complexType>
```

```
</element>
<element name=”getCie10Response”>
    <complexType>
    <sequence>
      <element name=”getCie10Return” type=”tns1:RegCie”/>
    </sequence>
    </complexType>
</element>
<element name=”getListaCie10”>
    <complexType>
    <sequence>
      <element name=”CantReg” type=”xsd:int”/>
      <element name=”Filtro” type=”xsd:string”/>
    </sequence>
    </complexType>
</element>
<element name=”getListaCie10Response”>
    <complexType>
    <sequence>
 <element maxOccurs=”unbounded” name=”getListaCie10Return” type=”tns1:RegCie”/>
    </sequence>
    </complexType>
</element>
<element name=”getMedico”>
    <complexType>
    <sequence>
      <element name=”nroCajaProf” type=”xsd:int”/>
    </sequence>
    </complexType>
</element>
<element name=”getMedicoResponse”>
    <complexType>
    <sequence>
      <element name=”getMedicoReturn” type=”tns1:Medico”/>
    </sequence>
    </complexType>
</element>
<element name=”getUE”>
    <complexType>
    <sequence>
      <element name=”Id” type=”xsd:int”/>
    </sequence>
    </complexType>
</element>
<element name=”getUEResponse”>
    <complexType>
    <sequence>
      <element name=”getUEReturn” type=”tns1:UnEjecutora”/>
    </sequence>
    </complexType>
</element>
<element name=”getListaUE”>
    <complexType>
    <sequence>
      <element name=”asist” type=”xsd:string”/>
    </sequence>
    </complexType>
</element>
<element name=”getListaUEResponse”>
```

```
        <complexType>
        <sequence>
     <element maxOccurs="unbounded" name="getListaUEReturn" type="tns1:UnEjecutora"/>
        </sequence>
        </complexType>
    </element>
    <element name="getListaExternos">
        <complexType/>
    </element>
    <element name="getListaExternosResponse">
        <complexType>
        <sequence>
     <element maxOccurs="unbounded" name="getListaExternosReturn" type="xsd:string"/>
        </sequence>
        </complexType>
    </element>
    </schema>
     <schema elementFormDefault="qualified" targetNamespace="http://wspackage"
xmlns="http://www.w3.org/2001/XMLSchema">
      <complexType name="RegCie">
           <sequence>
           <element name="cod" nillable="true" type="xsd:string"/>
           <element name="descr" nillable="true" type="xsd:string"/>
           </sequence>
    </complexType>
    <complexType name="Medico">
           <sequence>
           <element name="apellido" nillable="true" type="xsd:string"/>
           <element name="documento" nillable="true" type="xsd:string"/>
           <element name="nombre" nillable="true" type="xsd:string"/>
           <element name="nroCajaProf" type="xsd:int"/>
           </sequence>
    </complexType>
    <complexType name="UnEjecutora">
        <sequence>
        <element name="id" type="xsd:int"/>
          <element name="nombre" nillable="true" type="xsd:string"/>
          </sequence>
    </complexType>
    </schema>
</wsdl:types>

    <wsdl:message name="getListaUEResponse">

    <wsdl:part element="impl:getListaUEResponse" name="parameters">

    </wsdl:part>

    </wsdl:message>

    <wsdl:message name="getCie9Response">

    <wsdl:part element="impl:getCie9Response" name="parameters">

    </wsdl:part>

    </wsdl:message>

    <wsdl:message name="getListaCie9Response">

    <wsdl:part element="impl:getListaCie9Response" name="parameters">

    </wsdl:part>

    </wsdl:message>
```

<wsdl:message name=”getListaCie10Request”>
<wsdl:part element=”impl:getListaCie10” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getListaCie10Response”>
<wsdl:part element=”impl:getListaCie10Response” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getListaUERequest”>
<wsdl:part element=”impl:getListaUE” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getCie10Response”>
<wsdl:part element=”impl:getCie10Response” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getListaExternosResponse”>
<wsdl:part element=”impl:getListaExternosResponse” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getUERequest”>
<wsdl:part element=”impl:getUE” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getCie10Request”>
<wsdl:part element=”impl:getCie10” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getMedicoRequest”>
<wsdl:part element=”impl:getMedico” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getCie9Request”>
<wsdl:part element=”impl:getCie9” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getMedicoResponse”>
<wsdl:part element=”impl:getMedicoResponse” name=”parameters”>
</wsdl:part>

```
</wsdl:message>
<wsdl:message name=”getListaCie9Request”>
<wsdl:part element=”impl:getListaCie9” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getUEResponse”>
<wsdl:part element=”impl:getUEResponse” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:message name=”getListaExternosRequest”>
<wsdl:part element=”impl:getListaExternos” name=”parameters”>
</wsdl:part>
</wsdl:message>
<wsdl:portType name=”WsCod”>
<wsdl:operation name=”getCie9”>
<wsdl:input message=”impl:getCie9Request” name=”getCie9Request”>
</wsdl:input>
 <wsdl:output message=”impl:getCie9Response” name=”getCie9Response”>
      </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name=”getListaCie9”>
 <wsdl:input message=”impl:getListaCie9Request” name=”getListaCie9Request”>
      </wsdl:input>
 <wsdl:output message=”impl:getListaCie9Response” name=”getListaCie9Response”>
      </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name=”getCie10”>
      <wsdl:input message=”impl:getCie10Request” name=”getCie10Request”>
      </wsdl:input>
 <wsdl:output message=”impl:getCie10Response” name=”getCie10Response”>
      </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name=”getListaCie10”>
 <wsdl:input message=”impl:getListaCie10Request” name=”getListaCie10Request”>
      </wsdl:input>
 <wsdl:output message=”impl:getListaCie10Response” name=”getListaCie10Response”>
      </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name=”getMedico”>
 <wsdl:input message=”impl:getMedicoRequest” name=”getMedicoRequest”>
      </wsdl:input>
 <wsdl:output message=”impl:getMedicoResponse” name=”getMedicoResponse”>
      </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name=”getUE”>
      <wsdl:input message=”impl:getUERequest” name=”getUERequest”>
      </wsdl:input>
      <wsdl:output message=”impl:getUEResponse” name=”getUEResponse”>
      </wsdl:output>
      </wsdl:operation>
```

```
        <wsdl:operation name=”getListaUE”>
    <wsdl:input message=”impl:getListaUERequest” name=”getListaUERequest”>
        </wsdl:input>
    <wsdl:output message=”impl:getListaUEResponse” name=”getListaUEResponse”>
        </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name=”getListaExternos”>
    <wsdl:input message=”impl:getListaExternosRequest” name=”getListaExternosRequest”>
        </wsdl:input>
    <wsdl:output message=”impl:getListaExternosResponse”
name=”getListaExternosResponse”>
        </wsdl:output>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name=”WsCodSoapBinding” type=”impl:WsCod”>
    <wsdlsoap:binding style=”document” transport=”http://schemas.xmlsoap.org/soap/http”/>
        <wsdl:operation name=”getCie9”>
        <wsdlsoap:operation soapAction=””/>
        <wsdl:input name=”getCie9Request”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:input>
        <wsdl:output name=”getCie9Response”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name=”getListaCie9”>
        <wsdlsoap:operation soapAction=””/>
        <wsdl:input name=”getListaCie9Request”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:input>
        <wsdl:output name=”getListaCie9Response”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name=”getCie10”>
        <wsdlsoap:operation soapAction=””/>
        <wsdl:input name=”getCie10Request”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:input>
        <wsdl:output name=”getCie10Response”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name=”getListaCie10”>
        <wsdlsoap:operation soapAction=””/>
        <wsdl:input name=”getListaCie10Request”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:input>
        <wsdl:output name=”getListaCie10Response”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name=”getMedico”>
        <wsdlsoap:operation soapAction=””/>
        <wsdl:input name=”getMedicoRequest”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:input>
        <wsdl:output name=”getMedicoResponse”>
        <wsdlsoap:body use=”literal”/>
```

```
        </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name=”getUE”>
        <wsdlsoap:operation soapAction=””/>
        <wsdl:input name=”getUERequest”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:input>
        <wsdl:output name=”getUEResponse”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name=”getListaUE”>
        <wsdlsoap:operation soapAction=””/>
        <wsdl:input name=”getListaUERequest”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:input>
        <wsdl:output name=”getListaUEResponse”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name=”getListaExternos”>
        <wsdlsoap:operation soapAction=””/>
        <wsdl:input name=”getListaExternosRequest”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:input>
        <wsdl:output name=”getListaExternosResponse”>
        <wsdlsoap:body use=”literal”/>
        </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name=”WsCodService”>
        <wsdl:port binding=”impl:WsCodSoapBinding” name=”WsCod”>
    <wsdlsoap:address location=”http://localhost:8080/WSCodigos/services/WsCod”/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

La configuración del servicio, se encuentra detallada en el Anexo 6 procedimiento de instalación.

ANEXO 6: PROCEDIMIENTO DE INSTALACIÓN

El paquete de instalación cuenta con los siguientes archivos:

- Egresos.rar (proyecto de WaveMaker)
- WSCodigos.war
- egresosDB.properties
- egresos.sql
- codigos.sql

Primer paso:

Lo siguiente es una imagen de la configuración en WaveMaker de la validación de usuarios por medio de una base de datos de usuario, en forma de directorio y utilizando el protocolo de comunicación LDAP.

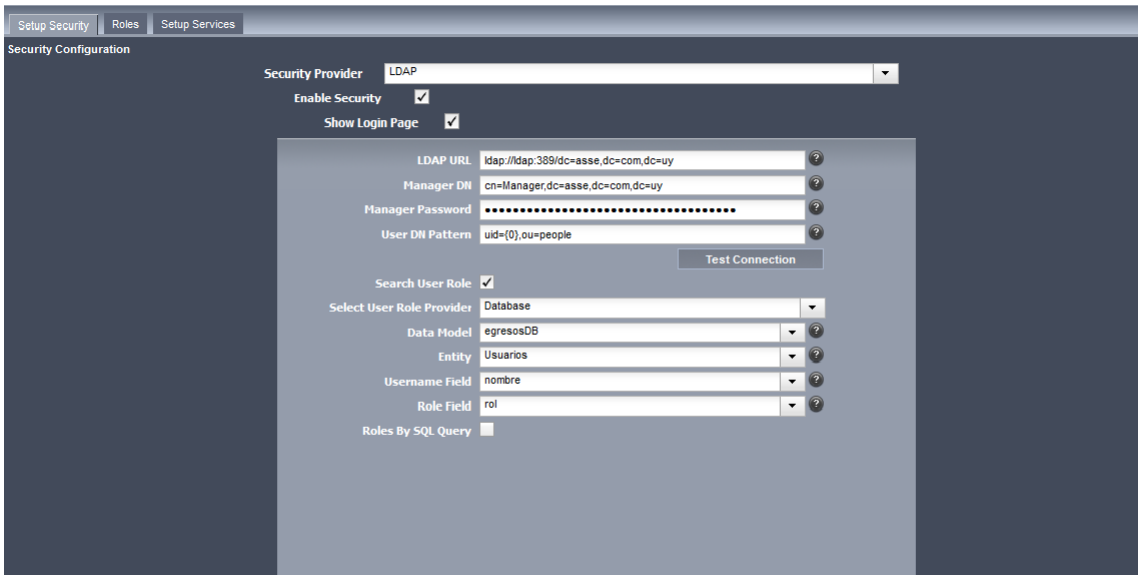


Figura 17: Configuración WaveMaker para LDAP de ASSE

Security Provider : Para nuestro caso LDAP.

Enable Security: Habilitar seguridad al iniciar el programa.

Show Login Page: Mostrar página de autenticación.

LDAP URL: Ubicación del directorio.

Manager DN : Especificación del usuario manager.

Manager Password: Clave del usuario manager.

User DN Pattern: Patrón de usuario DN.

Search User Role: Este flag y el resto de la sección de la página, es la que define el origen para la obtención de los roles que tiene el usuario que ha iniciado la sesión. En este caso, los roles son mantenidos por la propia aplicación ya que ASSE no los define a nivel de directorio.

Segundo Paso:

Desde WaveMaker realizar el deploy del SCE. Se generará el archivo *Egresos.war*.

Tercer Paso:

Realizar las siguientes modificaciones dentro del archivo *Egresos.war*:

- En *WEB_INF/classes/WEB_Personas/parametros.properties* configurar la URL del Padrón de Usuarios de ASSE:

  http://wsafiliados:8080/WSPersonas/servlet/aserver_personas

- En *WEB_INF/classes/DefaultNamespace/parametros.properties* configurar la URL para el WS de codigos
  - http://wscodigos:8080/WSCodigos/services/WsCod
- Borrar de *WEB-INF/classes* el archivo *egresosDB.properties.*
- En *WEB-INF/classes/egresosDB.spring.xml* modificar la ruta al archivo *egresosDB.properties,* sustituyendo el texto "classpath egresosDB.properties" por
  - file:///${CATALINA_HOME}/conf/egresosDB.properties

    Nota: se debe tener definida la variable de entorno CATALINA_HOME a la ruta del servidor Tomcat.

Cuarto Paso:

Copiar el archivo *Egresos.war* al directorio ${CATALINA_HOME}/*webapps* del servidor Tomcat.

Copiar el archivo *WSCodigos.war* al directorio ${CATALINA_HOME}/webapps

Quinto Paso:

Copiar archivo *egresosDB.properties* a: ${CATALINA_HOME}/conf/egresosDB.properties.

Sexto Paso:

Editar en el archivo ${CATALINA_HOME}/conf/egresosDB.properties los atributos

egresosDB.connectionUrl con la URL a la base de datos *egresos.*

Séptimo Paso:

Agregar al archivo *${CATALINA_HOME}/conf/context.xml:*

```
<Resource
    name="jdbc/dbcodigos" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    url="jdbc:mysql://SERVER:3306/codigos"
    driverClassName="com.mysql.jdbc.Driver"
```

username="USER" password="PASSWORD"/>

donde SERVER es la URL a la base de datos *codigos.*

USER usuario a la base de datos *codigos*

PASSWORD password para la base de datos *codigos*

Octavo Paso:

Reiniciar Tomcat.

Noveno Paso:

Crear bases de datos codigos y egresos y ejecutar los scripts correspondientes *codigos.sql* y *egresos.sql*. Crear usuario en mysql 'egresos2012' con password 'egresos2012' y ejecutar las siguientes consultas de asignación de permisos:

```
grant all on egresos.* to egresos2012@'IP_SERVIDOR_BD' identified by
'egresos2012';

grant all on codigos.* to egresos2012@'IP_SERVIDOR_BD' identified by
'egresos2012';

flush privileges;
```

donde IP_SERVIDOR_BD es la ip del servidor de bd donde se ejecutaron los scripts anteriores.