Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sistema de Alerta Precoz de la Sepsis Grave y Shock Séptico

Proyecto Final de Carrera

Ingeniería Técnica en Informática de Sistemas

Autor: Alberto Domínguez Ricarte - Sergio Alejandro Pastor Pastor

Director Universidad Politécnica: Jon Ander Gómez Adrián

Director Hospital Peset Aleixandre: José Miguel Puig Saqués

12/08/2013

Resumen

Durante la realización de nuestro proyecto de final de Carrera en el Hospital Universitario Doctor Peset Aleixandre de Valencia, hemos desarrollado una aplicación de diagnóstico automático de Sepsis Grave y prevención ante Shock Séptico. Dicha aplicación recoge datos de forma mecánica y analiza los posibles eventos de Sepsis, y en caso positivo, emite una alerta móvil avisando al responsable del nuevo evento, de forma que el paciente afectado recibe la atención y la medicación necesaria en la mayor brevedad de tiempo posible (algo vital como veremos más adelante). Los profesionales del hospital tendrán a su disposición un sitio web donde poder ver las nuevas alertas y las que aún están vigentes, pudiendo confirmar o descartar un caso de Sepsis, añadir comentarios, observar los valores anómalos, etc…

Trabajo Realizado:

La mayoría del trabajo ha sido realizado a partes iguales, y ambos hemos tocado el mismo código, debido a que hemos utilizado la programación en parejas típica de las metodologías ágiles, y en concreto, de *eXtreme Programming*.

Sin embargo, cada uno se ha centrado en un aspecto del proyecto a la hora de idear, diseñar o testear dicho apartado.

Si bien está explicado en las Historias de Usuario, Tareas y Pruebas de *eXtreme Programming* quien ha realizado dichas funciones, lo expondremos aquí para dejar constancia de ello:

- Sergio Alejandro Pastor Pastor:
  - Dar formato a los valores obtenido en las consultas a Orion.
  - Inserción de los valores umbrales y datos a comparar en nuetra BD.
  - Desarrollo de algoritmos de detección, análisis, alertas y borrado.
  - Creación del archivo de configuración.
  - Consulta e Inserción en el Histórico de Alertas.
  - Encargado de la explotación de datos con fines estadísticos.
  - Realización de las pruebas que le corresponden.

- Alberto Domínguez Ricarte:
  - Creación de consultas a Orion para la obtención de datos.
  - Creación de la base de datos propia de nuestra aplicación.
  - Inserción de datos en nuestra Base de Datos propia.
  - Extracción y Encapsulado de datos de nuestra BD.
  - Desarrollo de las páginas web de Alertas e Históricos.
  - Algoritmo de autentificación.
  - Realización de las pruebas que le corresponden.

Tabla de contenidos

1. Introducción 7
2. El problema de la Sepsis 8
3. SepSOS 10
   - ¿En que Consiste Sepsos y como funciona? 10
   - Diagrama de funcionamiento-Modulos 11
   - Criterios de Pruebas-Valores para el lanzamiento de una Alerta por Sepsis 12
4. Diseñando SepSOS – Metodologias Ágiles 14
   - ¿Qué es una metodología? 14
5. Fase de Análisis 16
   - Historias de Usuario 16
   - Bocetos de la Web 25
   - Bocetos Alerta 27
   - Diagrama UML 28
6. Fase de Desarrollo 29
   - Tareas de Ingeniería 29
   - Base de Datos de la Aplicación SepSOS 45
   - Implementación y Código 51
   - Proyecto DANA 52
   - Proyecto Página Web SepSOS 56
   - Proyecto SepSOS 65
7. Fase de Pruebas 93
   - Casos de Prueba de Aceptación 93
   - Web de Pruebas 113
8. Ejemplo de Funcionamiento 115
9. Instalación del Servicio 119
10. Mejoras Implementadas 120
11. Posibles Mejoras a Implementar 121
12. Repercusiones de SepSOS 122
13. Bibliografía 127
14. Opinión Personal 128
15. Agradecimientos 129

1. Introducción

¿Qué es la Sepsis?

SEPSIS (del griego *septos*, que significa podredumbre), es la respuesta sistémica del huésped a la infección que tiene una finalidad eminentemente defensiva. Se conoce como sepsis al síndrome de respuesta inflamatoria sistémica (SRIS) provocado por una infección, generalmente grave. Esta reacción del organismo se desarrolla como respuesta a gérmenes patógenos pero no se debe a la presencia de los microorganismos en sí, sino a la acción del sistema inmune liberando sustancias pro-inflamatorias que ponen en marcha el SRIS.

Definiciones de términos relacionados con la Sepsis:

- SRIS: Síndrome de Respuesta Inflamatoria Sistémica. Conjunto de Signos y Sintomas comunes en la respuesta frente a agresiones diversas que suscitan inflamación (no necesariamente infecciones).
- Sepsis Grave: Sepsis con datos de hipoperfusión tisular (hipotensión arterial, acidosis, oliguria e incluso obnubilación). En la actualidad es considerada como la causa de muerte más común en unidades de cuidados intensivos (UCI) no coronarias.
- Shock Séptico: Sepsis grave con hipotensión arterial que persiste a pesar de la infusión de volumen (se trata de un tipo de shock distributivo).
- Fallo multiorgánico: Fracaso de uno o varios órganos (generalmente varios), en el seno de una enfermedad aguda grave como el shock séptico. Una de las consecuencias del shock séptico es el fallo multiorgánico.

Causas más comunes de la Sepsis:

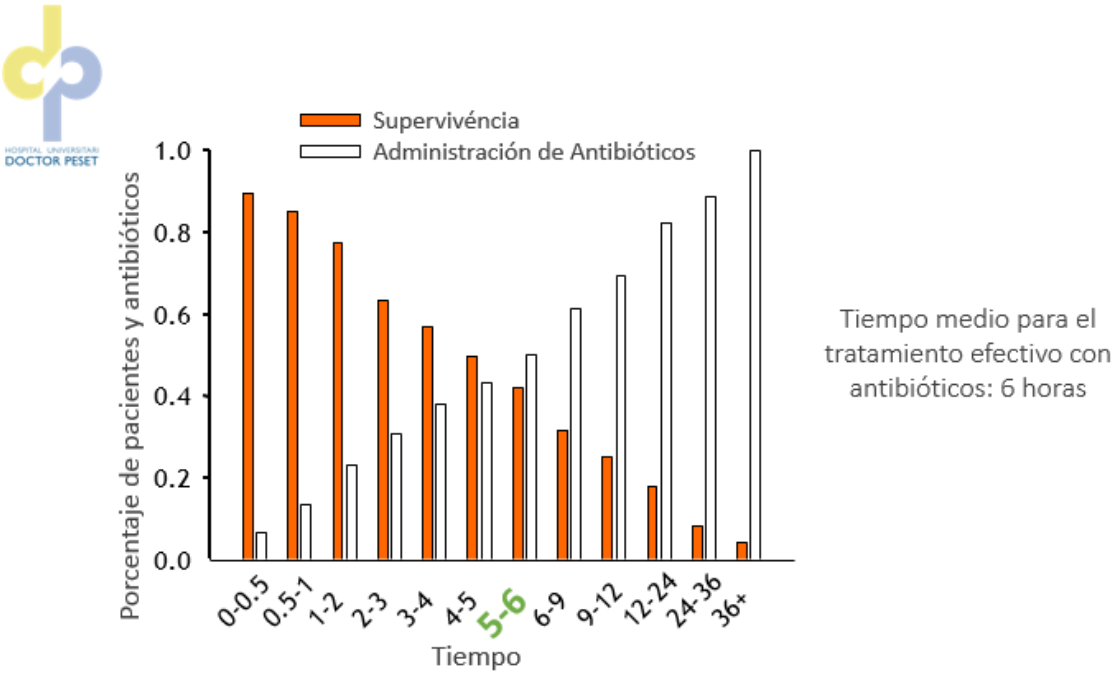
Durante la Sepsis se desencadenan una cascada no controlada de eventos inflamatorios en el organismo. La diabetes mellitus, las enfermedades linfoproliferativas como la leucemia, la cirrosis hepática, las quemaduras extensas y ciertos fármacos neutropénicos son los principales factores predisponentes para la aparición de Sepsis por *bacterias Gram negativas*. Por su parte, la Sepsis por bacterias Gram positivas se ve favorecida por catéteres intravenosos o sondas vesicales, prótesis o el uso de drogas intravenosas.

2. El problema de la Sepsis

Estudios nacionales indican que cada 90 minutos, un paciente en estado avanzado de SEPSIS ingresa en la UCI, con edades entre 50~80 años y con una mortalidad entorno al 33%.

De esos pacientes, solo el 64% recibe los antibióticos antes del punto crítico de la enfermedad (que es de 6 horas desde su aparición). De todos los pacientes que reciben antibióticos, solo el 51% recibe los antibióticos correctos a su afección.

El retraso en el diagnóstico de la enfermedad y en el tratamiento correcto implica más Disfunción Orgánica y mayor mortalidad debido a su rápida evolución.



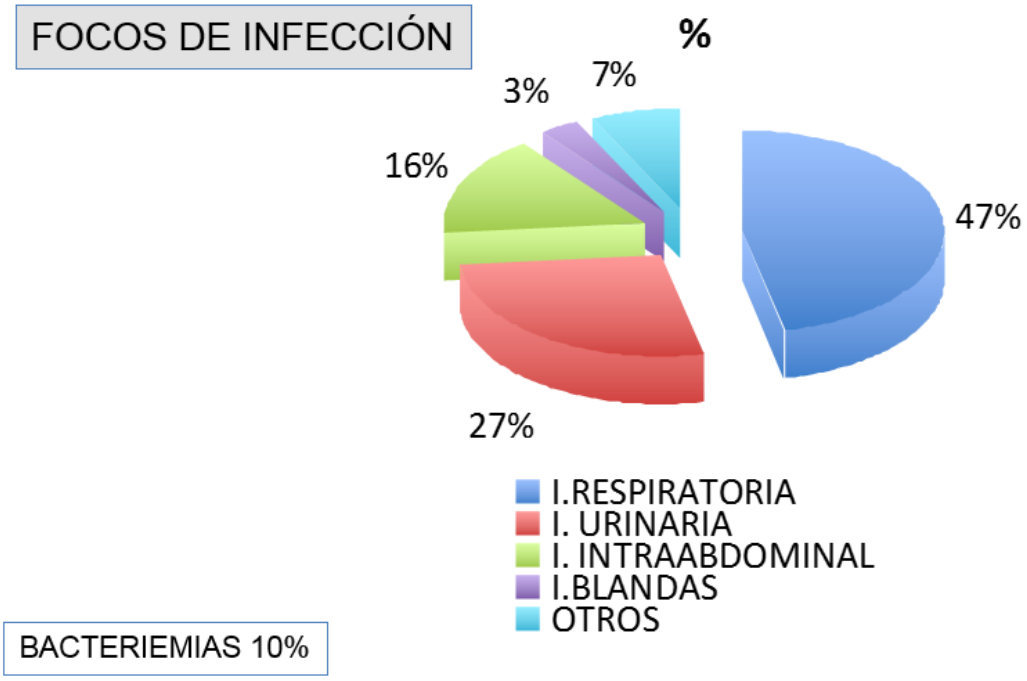
Detectar la enfermedad en su fase más temprana implica no solo un mayor número de vidas salvadas (descenso de la mortalidad en un 20%), si no también un ahorro en antibióticos y un menor tiempo de estancia en el hospital (En el caso de la UCI, un día de ingreso equivale a 1200€).

Recursos que podrían ser destinados a otros pacientes u otros propósitos.

Causas más comunes de la Sepsis:

Como se puede observar en el diagrama, las causas de un Shock Séptico pueden ser diversas.

El diagrama muestra los porcentajes en una relación Sepsis-Causa, siendo la infección respiratoria la mayor causante de Sepsis, seguida de la infección urinaria, la intraabdominal, otras multiples causas y las infecciones blandas.



3. SepSOS

Ante la necesidad de una detección precoz de la enfermedad y de un ahorro de recursos hospitalarios nace *SepSOS*. SepSOS es un proyecto de la Unidad de Informática del Doctor Peset Aleixandre y de la Unidad de Cuidados Intensivos, realizado en colaboración con la Universidad Politécnica de Valencia.

¿En qué consiste SepSOS y cómo funciona?

Mediante la automatización de un análisis de resultados clínicos, se pueden detectar tempranamente valores anómalos correspondientes a un caso de *Sepsis* y agilizar el diagnóstico de ésta.

Ante una detección, se procederá a avisar al profesional responsable mediante una alerta web en su ordenador, y al servicio correspondiente mediante un mensaje de texto.

SepSOS se encuentra separado en módulos, cada uno con una función específica:

- Módulo de Extracción de Datos: Es el encargado de coger de los diversos sistemas de información los datos que se precisen. Debe ser parametrizable ya que los criterios de extracción aplicados pueden cambiar con el tiempo, o incluso se pueden extender a otros tipos de alarmas para otras colecciones de datos.

- Módulo de Detección: Se encarga de comprobar que pruebas dan positivo para la afección indicada. En el caso de que se den los criterios de un posible caso de sepsis, se procede a lanzar la alerta.

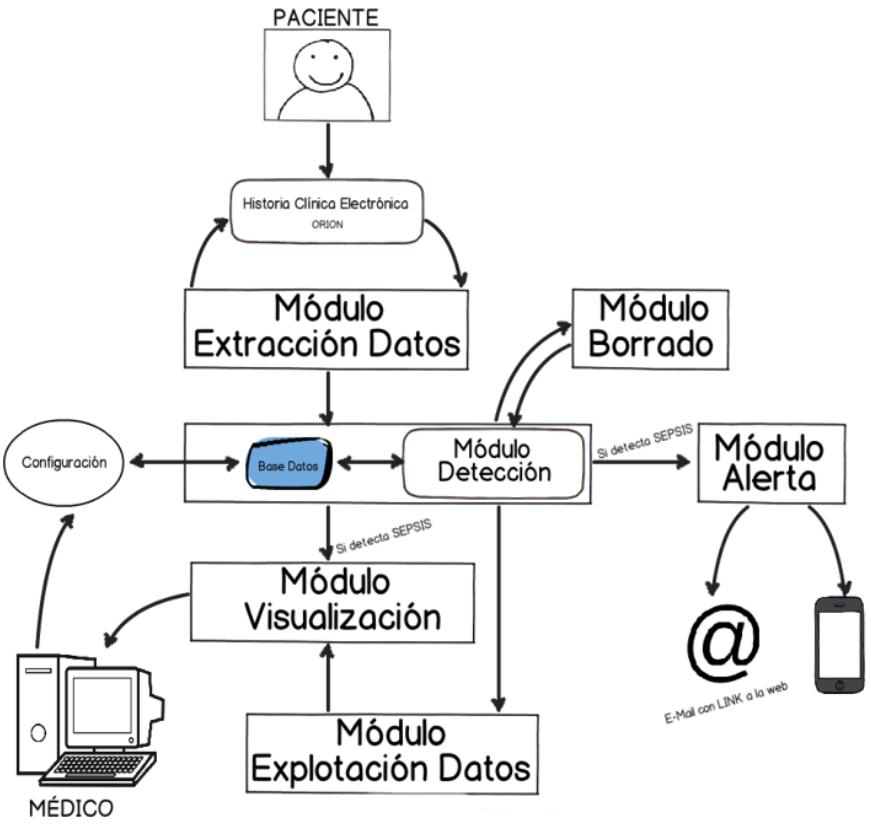
- Módulo de Presentación/Visualización: Se encarga de presentar al médico profesional el estado actual de los pacientes del hospital que tienen un posible caso de Sepsis (esto es, las nuevas alertas y las alertas que se están tratando). Puede ser filtrado por alertas atendidas, sin atender o todas, además de las que están marcadas para borrar (ya sea porque el paciente se ha curado o porque ha dado un falso positivo y no era un caso de Sepsis).

- Módulo de Alertas: Se encarga de gestionar el resultado del módulo anterior e informará al profesional responsable de la alerta por Sepsis. Dicho módulo de alertas dispondrá de alertas por correo electrónico y por servicio de mensajería a teléfono móvil corporativo de forma automatizada, además de por una página web dedicada al servicio de alertas por Sepsis.

- Módulo de borrado: Se encarga del borrado de pruebas una vez el médico ha marcado para borrar la alerta, o si para esas pruebas, no se ha lanzado una alerta. Una vez el paciente ha sido marcado como curado en Orion Clinics (Aplicación utilizada en el hospital Peset Aleixandre para gestionar todo el volumen de datos producidos por el hospital), y tras pasar 48 horas, las pruebas procederán a borrarse y la alerta por Sepsis (si la ha habido) pasará a guardarse en el histórico de alertas por Sepsis.

Con esta estructura se pretende garantizar que el control de la Sepsis tenga una infraestructura robusta que garantice el objetivo que se pretende.

Diagrama de Funcionamiento – Módulos:



Criterios de Pruebas-Valores para el lanzamiento de una Alerta por Sepsis:

Prueba	Comparación	Valor Comparación	Unidad Medida	Edad Mínima	Edad Máxima	¿Guardar Valor Previo?	Peso	Vigencia Horas
Bilirrubina Total	>=	4	mg/dL		16 años	☐	1	
Bilirrubina Total	>=	2	mg/dL	16 años	150 años	☐	1	
Creatinina	>=	0,8	mg/dL		2 años	☑	1	
Creatinina	>=	1,2	mg/dL	2 años	6 años	☑	1	
Creatinina	>=	1,6	mg/dL	6 años	13 años	☑	1	
Creatinina	>=	2	mg/dL	13 años	18 años	☑	1	
Creatinina	>=	1,5	mg/dL	18 años	150 años	☑	1	
Frec. Cardiaca	>	180	lat/min		2 años	☑	100	
Frec. Cardiaca	>	140	lat/min	2 años	6 años	☑	100	
Frec. Cardiaca	>	130	lat/min	6 años	13 años	☑	100	
Frec. Cardiaca	>	110	lat/min	13 años	18 años	☑	100	
Frec. Cardiaca	>	90	lat/min	18 años	150 años	☑	100	
Frec. Respiratoria	>	50	resp/min		7 días	☑	100	
Frec. Respiratoria	>	40	resp/min	8 días	30 días	☑	100	
Frec. Respiratoria	>	35	resp/min	31 días	2 años	☑	100	
Frec. Respiratoria	>	30	resp/min	2 años	6 años	☑	100	
Frec. Respiratoria	>	20	resp/min	6 años	150 años	☑	100	
INR	>	1,5	INR		150 años	☐	1	
Lactato	>=	2,5	mmol/L		150 años	☑	1000	24
Lactato Gasometría	>=	2,5	mmol/L		150 años	☑	1000	24
Leucocitos	>	30	x10e9/L		7 días	☐	100	
Leucocitos	>	19,5	x10e9/L	8 días	30 días	☐	100	
Leucocitos	>	17,5	x10e9/L	31 días	2 años	☐	100	
Leucocitos	>	15,5	x10e9/L	2 años	6 años	☐	100	
Leucocitos	>	13,5	x10e9/L	6 años	13 años	☐	100	
Leucocitos	>	11	x10e9/L	13 años	18 años	☐	100	
Leucocitos	>	12	x10e9/L	18 años	150 años	☐	100	
Leucocitos	<	5	x10e9/L		2 años	☐	100	
Leucocitos	<	6	x10e9/L	2 años	6 años	☐	100	
Leucocitos	>	15,5	x10e9/L	2 años	6 años	☐	100	
Leucocitos	>	13,5	x10e9/L	6 años	13 años	☐	100	
Leucocitos	>	11	x10e9/L	13 años	18 años	☐	100	
Leucocitos	>	12	x10e9/L	18 años	150 años	☐	100	
Leucocitos	<	5	x10e9/L		2 años	☐	100	
Leucocitos	<	6	x10e9/L	2 años	6 años	☐	100	
Leucocitos	<	4,5	x10e9/L	6 años	18 años	☐	100	
Leucocitos	<	4	x10e9/L	18 años	150 años	☐	100	
pCO2	<	32	mmHg		150 años	☑	1	
PCR	>=	50	mg/L		16 años	☐	0	
PCR	>=	20	mg/L	16 años	150 años	☐	0	
PCT	>=	5	ng/mL		150 años	☑	1000	24
Plaquetas	<	100	x10e9/L		150 años	☐	1	
pO2	<	60	mmHg		150 años	☑	1	
Saturación O2	<	90	%		150 años	☐	1	
Temperatura	>	38	ºC		150 años	☑	100	
Temperatura	<	36	ºC		150 años	☑	100	
Tensión Arterial Sistólica	<	60	mmHg		7 días	☑	1	
Tensión Arterial Sistólica	<	70	mmHg	8 días	30 días	☑	1	
Tensión Arterial Sistólica	<	75	mmHg	31 días	6 años	☑	1	
Tensión Arterial Sistólica	<	85	mmHg	6 años	18 años	☑	1	
Tensión Arterial Sistólica	<	90	mmHg	18 años	150 años	☑	1	
TTPS	>=	60	s		150 años	☐	1	

En la tabla anterior se pueden observar las pruebas consultadas para posibles casos de Sepsis. Todas estas pruebas tienen su relevancia a la hora de lanzar una alerta, dicha relevancia se mide por la columna “Peso”. A mayor “Peso”, más importancia tiene la prueba.

Para que se lance una Alerta de Sepsis, se tienen que cumplir las siguientes condiciones o criterios:

- ✓ Que se produzca un Aviso Directo: Cuando una prueba con un *Peso* de 1000 da positivo, se considera un aviso directo y se lanza una Alerta por Sepsis. Es la alerta más grave que existe, y se muestra en la web de alertas como Alerta GRAVE.

- ✓ Que se produzca un aviso por SRIS: Para generar una Alerta por *SRIS* se necesitan mínimo dos pruebas positivas con un *Peso* de 100 cada una, lo cual genera un aviso de SRIS. A partir de esas dos pruebas positivas, cada prueba con un *Peso* de 100 o un *Peso* de 1 sumarán un punto a la gravedad del *SRIS*.

  Ejemplo:
  -Tres pruebas con peso 100 = SRIS + 1
  -Tres pruebas con peso 100 y una prueba con peso 1 positivas = SRIS + 2
  -Dos pruebas con peso 100 y dos pruebas con peso 1 positivas = SRIS + 2

  SRIS + 2 es más grave que SRIS + 1, pero ninguna de las dos es superior en cuanto gravedad al Aviso Directo visto anteriormente (es más, no importa la gravedad del *SRIS*, nunca serán consideradas más grave que un Aviso Directo).

La gravedad de la enfermedad es clave a la hora de su tratamiento, siendo una afección que evoluciona rápidamente desde su aparición. Una persona con una Sepsis catalogada como grave puede llegar a pasar muchos días en la UCI. Nuestro objetivo es que no necesite pasar por la Unidad de Cuidados Intensivos, atajando la enfermedad y curándole cuanto antes. Y en el caso de que no se pueda evitar su ingreso en la UCI, queremos que esté la mayor brevedad de tiempo posible, de esta forma conseguimos ahorrar recursos hospitalarios (1200€ cuesta al hospital estar un día completo en la UCI), tener más camas disponibles para otros enfermos en dicha unidad y una pronta recuperación del afectado por Sepsis.

4. DiseñandoSepSOS: Metodologías Ágiles

Debido a que somos alumnos de Informática de Sistemas con especialización en Ingeniería del Software, nos hemos decantado por utilizar metodologías agiles para la realización de la aplicación SepSOS.

¿Qué es una Metodología?

La metodología (del griego μέθοδος de μετά metá 'más allá, después, con', οδως odós 'camino' y λογος logos 'razón, estudio'), hace referencia al conjunto de procedimientos racionales utilizados para alcanzar una gama de objetivos que rigen en una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos. Alternativamente puede definirse la metodología como el estudio o elección de un método pertinente para un determinado objetivo.

- **Metodologías Agiles:**
  Son métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requerimientos y soluciones evolucionan mediante la colaboración de grupos auto-organizados y multidisciplinarios. Existen muchos métodos de desarrollo ágil, la mayoría minimiza riesgos desarrollando software en lapsos cortos. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, sino que la meta es tener un prototipo funcional al final de cada iteración. Al terminar la iteración el equipo vuelve a evaluar las prioridades del proyecto x.

Debido a que somos un grupo pequeño de personas, pero aprovechando la oportunidad que teníamos de trabajar junto al equipo de informática del Hospital Doctor Peset Aleixandre, decidimos trabajar con la metodología ágil ***eXtreme Programming*** o ***XP***.

- ***XP*** es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck. Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Se consideran los cambios de requisitos sobre la marcha como un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Hay que ser capaz de

adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto, siendo una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

- Está pensado para equipos de programadores pequeños (de hasta 10 personas), fomentando la programación en parejas.

- La comunicación con el cliente debe ser continua, a poder ser, una reunión diaria.

- Desarrollo incremental, con mejoras una detrás de otra.

- Pruebas unitarias, comprobando que lo que se realiza, funciona, obteniendo, al final de la iteración, un prototipo funcional.

- Refactorización del código, solucionando los errores antes de añadir una nueva funcionalidad.

- Código simple, cuanto más sencillo es algo, más posibilidad de que funcione. Además cada programador tiene a su disposición cualquier parte del código.

Durante nuestra estancia en el hospital realizando el proyecto, José Miguel Puig (Director del Departamento de Informática y Director del Proyecto Final de Carrera del Hospital) y David Estellés (Jefe del Centro de Atención de Usuarios y Responsable de la Unidad Informática de Sepsis) realizaron el rol de Clientes.

Al comienzo del proyecto, nos reunimos con ellos como Analistas, captando los requisitos que tenía la aplicación. Mediante Historias de Usuario, pudimos capturar las funcionalidades del programa, darles una prioridad...

Aclaradas las necesidades del proyecto, pasamos a realizar el Rol de Programadores, realizando una programación en parejas tal y como lo establece *XP*. Además, gracias a la voluntad y disponibilidad de José Miguel y David, pudimos reunirnos a diario para resolver dudas y añadir/retirar funcionalidades. Es de agradecer que un cliente se tome las molestias de atendernos y estar implicados en el desarrollo de una aplicación.

Cada vez que finalizábamos un módulo/iteración/funcionalidad, comprobábamos que los resultados eran los esperados aplicando las pruebas que previamente habíamos acordado con el cliente.

5. Fase de Análisis

A continuación se pueden observar las historias de usuario utilizadas durante la fase de análisis del proyecto final de carrera, pero antes, explicaremos que es una historia de usuario.

Historia de Usuario (HU):

Es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en *XP* para la especificación de requisitos (acompañadas de las discusiones con los usuarios y las pruebas de validación). Cada historia de usuario debe ser limitada, esta debería poderse escribir sobre una nota adhesiva pequeña. Dentro de la metodología XP las historias de usuario deben ser escritas por los clientes.

Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos. Además permiten responder rápidamente a los requisitos cambiantes.

A continuación, nuestras historias de usuario redactadas con nuestros clientes:

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Obtener Datos Pruebas
Usuario: Sistema	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alto	
Descripción: **Lectura de Orion Clinics (Base de Datos Informix) para obtener los datos de unas pruebas médicas respectivos a un paciente.**	
Observaciones: **El Sistema debe estar preparado para albergar datos de más afecciones a parte de la Sepsis. Pasamos de Informix a SQLServer**	

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Obtener Datos Pacientes
Usuario: Sistema	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alto	
Descripción: **Lectura de Orion Clinics (Base de Datos Informix) para obtener los datos personales respectivos a un paciente.**	
Observaciones: **El Sistema debe estar preparado para albergar datos de más afecciones a parte de la Sepsis. Pasamos de Informix a SQLServer**	

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Almacenar Datos Pruebas
Usuario: Sistema	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alta	
Descripción: **Guardar los datos de las pruebas extraídos en la base de datos propia de Sepsos.**	
Observaciones: **Las pruebas pueden ser del tipo “Triaje” y “Laboratorio”.**	

Historia de Usuario	
Número: 4	**Nombre Historia de Usuario:** Almacenar Datos Paciente
Usuario: Sistema	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alto	
Descripción: Guardar los datos de los pacientes extraídos en la base de datos propia de Sepsos	
Observaciones: Información con un alto nivel de privacidad. Máxima precaución y seguridad.	

Historia de Usuario	
Número: 5	**Nombre Historia de Usuario:** Precarga Valores de Comparación de Datos.
Usuario: Sistema	**Iteración Asignada:** 2
Prioridad en Negocio: Alto	
Riesgo en Desarrollo: Bajo	
Descripción: Añadir a la base de Datos de “SepSOS” los valores umbrales de resultados de pruebas para contrastar con los obtenidos de los pacientes.	
Observaciones: Dichos valores se utilizarán para dar un diagnóstico positivo o negativo de la enfermedad.	

Historia de Usuario	
Número: 6	**Nombre Historia de Usuario:** Precarga de Datos Hospitalarios
Usuario: Sistema	**Iteración Asignada:** 2
Prioridad en Negocio: Medio	
Riesgo en Desarrollo: Bajo	
Descripción: **El Sistema "SepSOS" debe conocer los datos sobre médicos, plantas, enfermedades, teléfonos, servicios…**	
Observaciones:	

Historia de Usuario	
Número: 7	**Nombre Historia de Usuario:** Análisis de Datos para el diagnóstico.
Usuario: Sistema	**Iteración Asignada:** 2
Prioridad en Negocio: Alto	
Riesgo en Desarrollo: Alto	
Descripción: **Implementación del algoritmo de diagnóstico para detectar casos positivos de Sepsis.**	
Observaciones: **El algoritmo está basado en el peso de las pruebas.**	

Historia de Usuario	
Número: 8	**Nombre Historia de Usuario:** Aviso Móvil
Usuario: Personal Médico	**Iteración Asignada:** 2
Prioridad en Negocio: Medio	
Riesgo en Desarrollo: Medio	
Descripción: **Aviso al móvil del Médico Responsable o del Servicio Responsable del Paciente, indicando un posible caso de Sepsis.**	
Observaciones: **La Frecuencia del aviso o a quien irá dirigido depende de la urgencia del caso y de que el aviso haya sido o no verificado.** **En el aviso aparecerán datos del paciente como "Cama", "Nº Historia Clínica", etc…**	

Historia de Usuario	
Número: 9	**Nombre Historia de Usuario:** Aviso al Correo Electrónico
Usuario: Personal Médico	**Iteración Asignada:** 3
Prioridad en Negocio: Baja	
Riesgo en Desarrollo: Bajo	
Descripción: **Aviso al correo electrónico del Médico Responsable de un posible caso de Sepsis.**	
Observaciones: **El aviso en el correo electrónico puede contener un link a la web "SepSOS"**	

Historia de Usuario	
Número: 10	**Nombre Historia de Usuario:** Visualización en la WEB
Usuario: Personal Médico - Informático	**Iteración Asignada:** 3
Prioridad en Negocio: Medio	
Riesgo en Desarrollo: Medio	
Descripción: La web contendrá todos los pacientes que hayan dado positivo en Sepsis, pudiendo ver sus datos en detalle. Cuando un médico reciba una alerta, deberá pulsar el botón de "Alerta Vista", confirmando que ya es consciente de que hay un posible caso de Sepsis.	
Observaciones: El profesional tendrá la opción de escribir comentarios, confirmar o no si era un caso de Sepsis, y la posibilidad de archivar la alerta si ya ha finalizado el trato con el paciente.	

Historia de Usuario	
Número: 11	**Nombre Historia de Usuario:** Configuración
Usuario: Personal Informático	**Iteración Asignada:** 4
Prioridad en Negocio: Bajo	
Riesgo en Desarrollo: Medio	
Descripción: La aplicación contendrá un apartado de configuración, mediante el cual, un profesional con privilegios para tal tarea se encargará de cambiar valores umbrales, frecuencias de avisos, acceso de personal, etc…	
Observaciones: Dicha sección solo podrá ser visualizada por el responsable de dicha labor.	

Historia de Usuario	
Número: 12	**Nombre Historia de Usuario:** LOG
Usuario: Personal Informático	**Iteración Asignada:** 4
Prioridad en Negocio: Bajo	
Riesgo en Desarrollo: Bajo	
Descripción: Guardaremos las acciones realizadas por los usuarios de la WEB de SepSOS.	
Observaciones:	

Historia de Usuario	
Número: 13	**Nombre Historia de Usuario:** Histórico de Alertas
Usuario: Personal Médico	**Iteración Asignada:** 5
Prioridad en Negocio: Bajo	
Riesgo en Desarrollo: Bajo	
Descripción: El profesional podrá marcar para archivar casos positivos de sepsis que quedarán almacenados en un histórico.	
Observaciones:	

Historia de Usuario	
Número: 14	**Nombre Historia de Usuario:** Estadísticas y búsquedas
Usuario: Personal Médico - Informático	**Iteración Asignada:** 5
Prioridad en Negocio: Bajo	
Riesgo en Desarrollo: Bajo	
Descripción: Análisis estadísticos de los parámetros de cada alerta, afección y paciente para un futuro estudio sobre la enfermedad y sobre la aplicación SepSOS.	
Observaciones:	

Historia de Usuario	
Número: 15	**Nombre Historia de Usuario:** Gestión de Datos.
Usuario: Personal Médico – Informático - Sistema	**Iteración Asignada:** 4
Prioridad en Negocio: Bajo	
Riesgo en Desarrollo: Bajo	
Descripción: Se define una ventana activa para las pruebas. Pasado esa ventana, el dato no es vigente y debe ser desechado.	
Observaciones: 48 horas después de que los datos sean desechados y Orion Clinic marque como que el paciente está curado, la alerta se archivará en el histórico.	

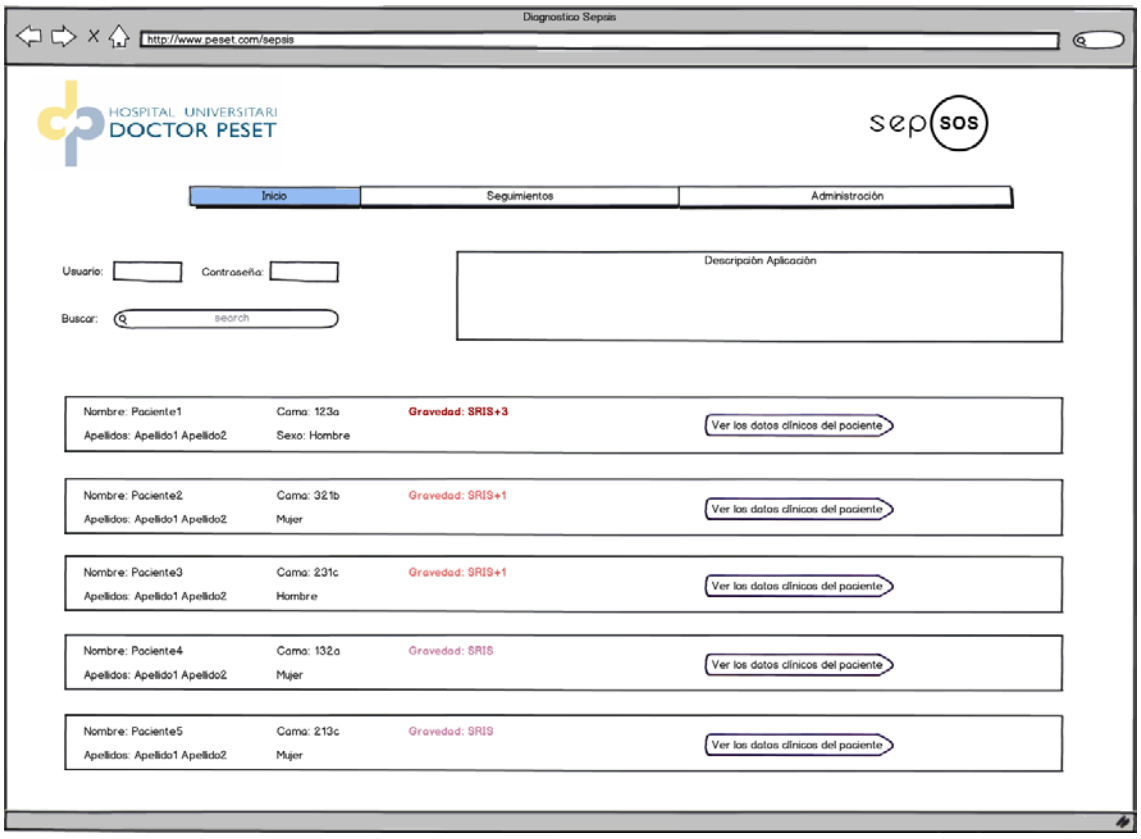
Historia de Usuario	
Número: 16	**Nombre Historia de Usuario:** Login
Usuario: Personal Médico - Informático	**Iteración Asignada:** 5
Prioridad en Negocio: Bajo	
Riesgo en Desarrollo: Bajo	
Descripción: **Autentificación de usuarios para el acceso a la aplicación/web.**	
Observaciones: **Se tratará siempre de profesionales del hospital.**	

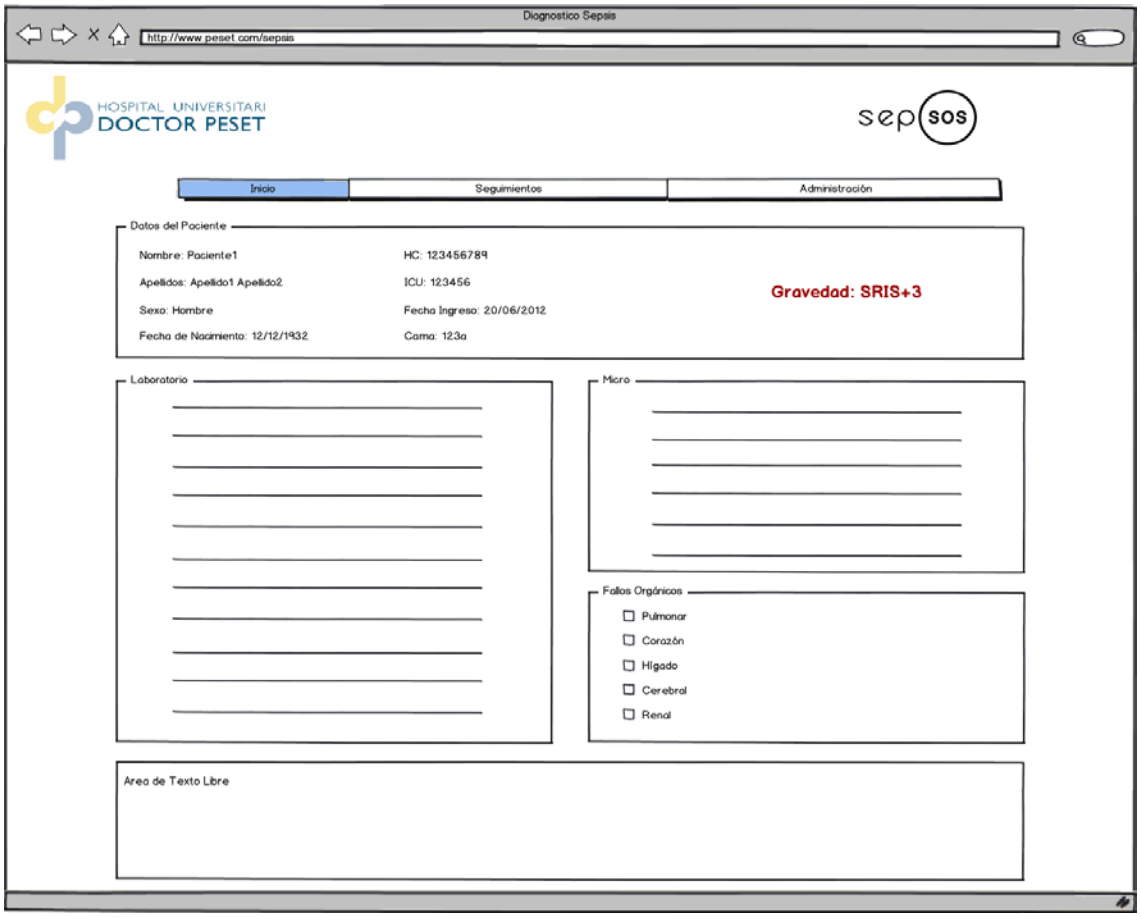
Estos son los requisitos acordados con el Hospital a la hora de realizar la aplicación SepSOS. Cabe destacar que las historias de usuario están diseñadas para ser utilizadas en reuniones, ser modificadas, ralladas, etc...
Nosotros las hemos pasado a limpio para poder incluirlas en el PFC.

Una vez el equipo de programadores recibe las historias de usuario, las divide en Tareas de Ingeniería para comenzar con el desarrollo de la aplicación.

Además de Historias de Usuario, pueden realizarse bocetos de la interfaz o cualquier otra documentación que sirva para captar mejor los requisitos de la aplicación. A continuación puede visualizarse la evolución que ha ido sufriendo la web mediante bocetos realizados con *Balsamiq Mockups*.

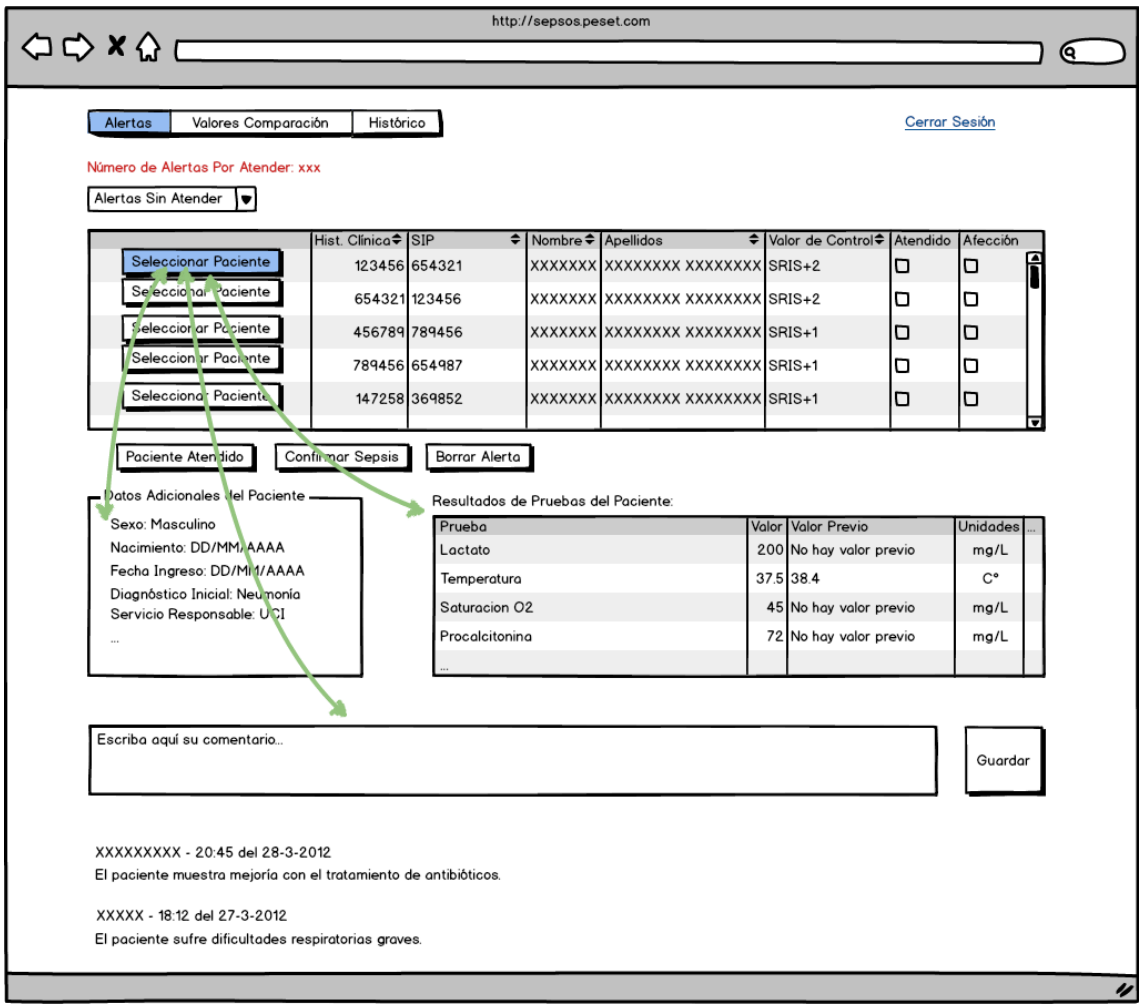
Bocetos de la WEB:





En un primer momento, la web estaba pensada para trabajar mediante múltiples ventanas. Al hacer click en la alerta de un paciente determinado (Primera Imagen), podíamos observar en una ventana diferente todos los datos del paciente (Segunda Imagen), como son sus datos personales, sus pruebas de laboratorio y de microbiología, fallos orgánicos, etc... y un Área de Texto libre para que el médico introduzca los comentarios que les parezca oportuno.

Más tarde, y tras varias reuniones con los médicos, responsables del Hospital y la unidad de Informática, decidimos rediseñar la web introduciendo relaciones Maestro-Esclavo, con toda la información contenida en una sola ventana:



Como se puede observar, al Seleccionar un Paciente, aparecen abajo los datos personales del paciente de menor relevancia, los resultados de las pruebas en una tabla, el Área de Texto Libre para los comentarios y los comentarios. Obviamente, si no hay ningún paciente seleccionado, dichas áreas y botones no se mostrarán. Además, tenemos una barra de navegación superior desde donde podemos acceder al histórico (muy similar a la ventana de alertas) y a una tabla con todos los valores de comparación que utilizamos para detectar la Sepsis (Tabla de la Página 11).

Bocetos Alertas:

Primera idea sobre la estructura que iban a tener los SMS_Alerta que iban a recibir los responsables de cada servicio.

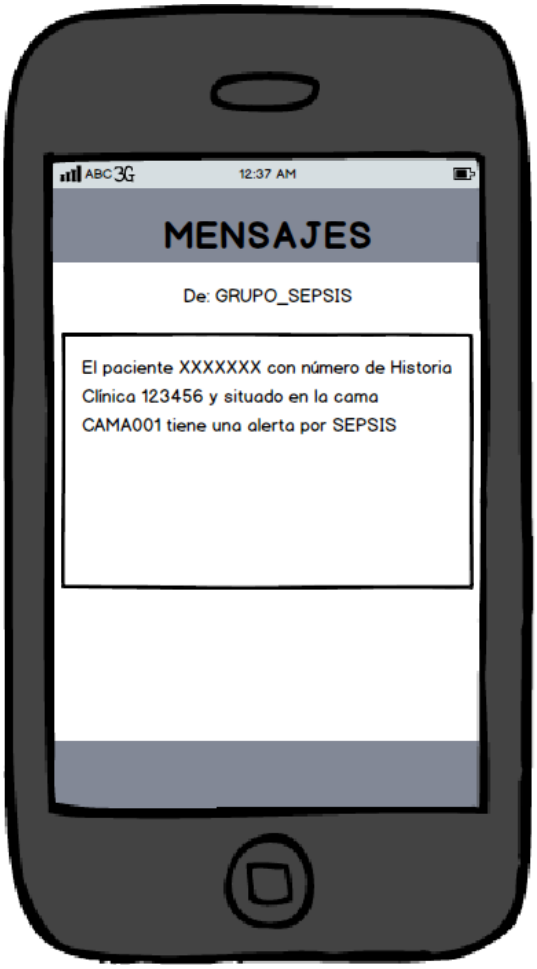
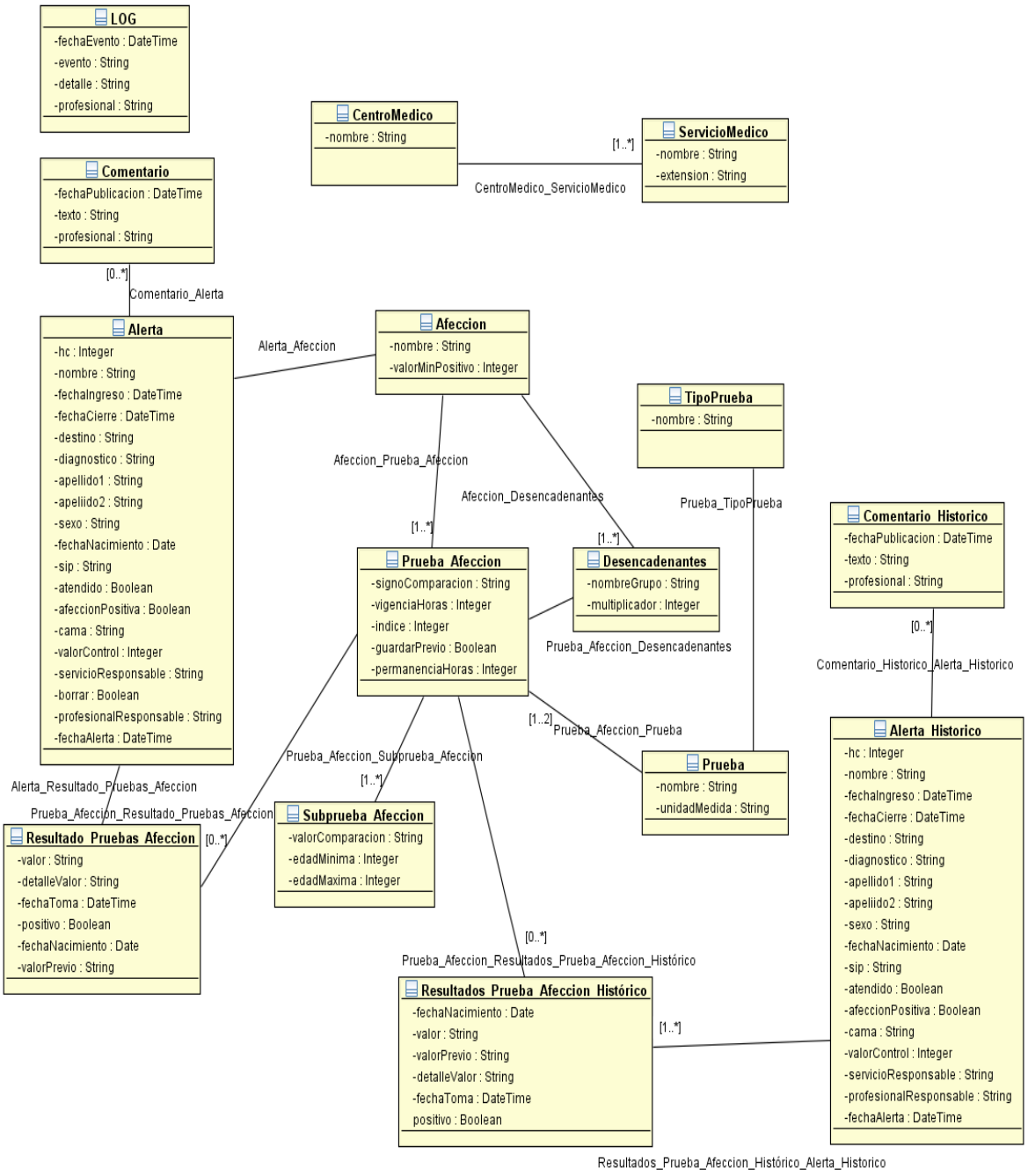


Diagrama UML:

A continuación se puede observar un diagrama en el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Dicho diagrama ha sido sometido a muchos cambios durante la fase de análisis, y nos ha servido para saber cómo organizar nuestra Base de Datos.



Gracias a que vamos a hacer uso de una programación orientada a objetos, el Diagrama UML es el complemento perfecto que nos va a facilitar el diseño de la aplicación.

6. Fase de Desarrollo

Las Historias de Usuario previamente mostradas pasan a ser fraccionadas en Tareas de Ingeniería.

Tareas de Ingeniería:

Las Tareas de Ingeniería son las funcionalidades a implementar extraídas de las Historias de Usuario. Cada tarea será asignada a un programador del equipo de desarrolladores. Cada tarea está relacionada con las historias de usuario a las que pertenece y tiene una fecha de finalización aproximada (nosotros hemos obviado dicha fecha debido a que no siempre podíamos estar físicamente en el Hospital trabajando).

A continuación, las Tareas de Ingeniería:

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 1, 2
Nombre Tarea: Establecer Conexión con Orion Clinic	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Añadir a Visual Studio los drivers y componentes necesarios para que pueda establecer una conexión vía ODBC con la base de datos de Orión la cual es de tipo IBM Informix.** **A continuación se parametrizará la conexión para apuntar a la réplica de solo lectura, de este modo conseguiremos trabajar con una BD menos sobrecargada mejorando los tiempo de respuesta en las consultas y aumentando la seguridad del sistema evitando que nadie pueda manipular los datos hospitalarios.** **La ruta de conexión debe estar en un fichero de configuración para su reutilización y fácil editado alterando lo menos posible la aplicación.**	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Escribir las consultas que devuelvan resultados de pruebas hospitalarias.	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Identificar dentro de Orión donde están los datos que necesitamos para saber que pruebas, de que tipo, cuando y su resultado se le han realizado a un paciente determinado. Escribir una consulta apuntando a las tablas y columnas de interés que nos devuelva la información. La consulta debe preguntar solo por resultados entre intervalos de tiempo, ya sea de hace 15 días para atrás (precarga del programa) o de hace 15 minutos (consulta rutinaria).	

	Tarea de Ingeniería
Número Tarea: 3	**Número Historia de Usuario:** 1, 2
Nombre Tarea: Encapsulado de la obtención de datos de Orión	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: Todos los métodos responsables de conectar y obtener datos de Orión tanto de pruebas como de datos de pacientes deben aparecer en una única biblioteca de acceso a datos con el fin de facilitar su uso frente a un cambio de origen de datos, de manera que el cambio no afecte a la integridad de sepSOS mas allá de esta capa.	

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 2
Nombre Tarea: Escribir las consultas que devuelvan datos de pacientes	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: **Identificar dentro de Orión donde están los datos personales y de localización tanto dentro del hospital (cama, servicio) y fuera de él (domicilio, en caso de que la alerta se produzca tras el alta) y quien es el médico responsable, incluido al servicio al que pertenece (para dar la alerta por sepsis) de un paciente determinado.**	

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 3, 4
Nombre Tarea: Interpretado de las consultas a Orión	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Debido a que la base de datos de SepSOS (sqlSERVER 2008) no es del mismo tipo que la de Orion Clinic (IBM Informix) y que los datos recibidos necesitan ser procesados previa inserción en nuestra BD por problemas de tipo de datos, aglutinación de ampos, etc… se procederá a un recorrido de los "dataTable" devueltos para hacer las correcciones oportunas.**	

	Tarea de Ingeniería
Número Tarea: 6	**Número Historia de Usuario:** 3
Nombre Tarea: Redactar Query de Inserción de datos de Pruebas.	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Una vez tenemos los datos de resultados de pruebas procesados y listos para la inserción procederemos a recorrer los objetos en el que se hallen y de forma iterativa ejecutaremos una Query por fila que irá haciendo inserciones y actualizaciones sobre nuestra base de datos.	

	Tarea de Ingeniería
Número Tarea: 7	**Número Historia de Usuario:** 3, 4
Nombre Tarea: Creación de la Base de Datos de SepSOS	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Crear las tablas necesarias con sus respectivos atributos para el almacenamiento de los datos de nuestra aplicación. Seguir la estructura del diagrama de clases (transformado en entidad-relación)	

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** 4
Nombre Tarea: Redactar Query de Inserción de datos de pacientes.	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Una vez tenemos en los datos de pacientes procesados y listos para inserción procederemos a recorrer los objetos en el que se hallen y de forma iterativa ejecutaremos una Query por fila que irá haciendo inserciones y actualizaciones sobre nuestra BD.	

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** 5
Nombre Tarea: Inserción de Valores de Comparación de Pruebas	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: Inserción en nuestra base de datos de las pruebas tanto de triaje como de resultados de laboratorio, sus respectivos valores y signos de comparación así como a quienes son aplicables (niños de diferentes edades y adultos)	

	Tarea de Ingeniería
Número Tarea: 10	**Número Historia de Usuario:** 6
Nombre Tarea: Insertado de valores de comparación de pruebas	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: Inserción en nuestra base de los datos información perteneciente al hospital como sus diferentes servicios, números de teléfonos corporativos, etc…	

	Tarea de Ingeniería
Número Tarea: 11	**Número Historia de Usuario:** 7, 10
Nombre Tarea: Redactar Query de recuperación de pruebas de SepSOS	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Escribir una consulta que nos devuelva resultados de pruebas tanto de laboratorio como de triaje de nuestra base de datos.	

Tarea de Ingeniería	
Número Tarea: 12	**Número Historia de Usuario:** 7,10
Nombre Tarea: Redactar Query de recuperación de pacientes de SepSOS	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Escribir una consulta que nos devuelva datos de pacientes tanto hospitalizados como en urgencias procedentes de nuestra base de datos.	

Tarea de Ingeniería	
Número Tarea: 13	**Número Historia de Usuario:** 8, 9, 10
Nombre Tarea: Redactar Querys de recuperación de datos hospitalarios de SepSOS	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Escribir una consulta que nos devuelva datos relativos a información sobre los médicos, servicios hospitalarios, teléfonos… procedentes de nuestra base de datos.	

	Tarea de Ingeniería
Número Tarea: 14	**Número Historia de Usuario:** 7 - 15
Nombre Tarea: Encapsulado de obtención de datos de sepSOS	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Todos los métodos relacionados con el acceso a la base de datos de SepSOS deberán estar encapsulados en una librería para facilitar su uso desde cualquier punto de la aplicación	

	Tarea de Ingeniería
Número Tarea: 15	**Número Historia de Usuario:** 7
Nombre Tarea: Escritura Algoritmo de Análisis de Datos	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: Redacción del algoritmo que se encargará de: -Recuperar datos de pruebas de triaje y laboratorio. -Comparar los resultados con los espectros de valores que indican Sepsis. -En caso afirmativo, traernos los datos del paciente. -En caso afirmativo, informar al algoritmo de envío de alertas que debe mandarse. -Al terminar, guardar todos los datos.	

Tarea de Ingeniería	
Número Tarea: 16	**Número Historia de Usuario:** 8,9
Nombre Tarea: Escritura del Algoritmo de Envío de Alertas	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Algoritmo de control que usará a los algoritmos de envío de alerta móvil y de correo electrónico.**	

Tarea de Ingeniería	
Número Tarea: 17	**Número Historia de Usuario:** 8
Nombre Tarea: Algoritmo de envío de alerta móvil	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Escritura de algoritmo que hará uso del recurso del hospital de enviar mensajes de texto a móviles corporativos para enviar alertas relacionadas con la Sepsis de forma autónoma, controlando si fuera necesario el reenvío.**	

	Tarea de Ingeniería
Número Tarea: 18	**Número Historia de Usuario:** 9
Nombre Tarea: Algoritmo de envío de correos electrónicos	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Escritura de algoritmo que hará uso del recurso del hospital para enviar correos electrónicos a médicos para enviar alertas relacionadas con la Sepsis de forma autónoma, controlando si fuera necesario el reenvío.**	

	Tarea de Ingeniería
Número Tarea: 19	**Número Historia de Usuario:** 10,14
Nombre Tarea: Página maestra de la web de SepSOS	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: **Diseño y programación de la página maestra en ASP.NET de la web de SepSOS, se trata del marco estático de funcionalidades no variables tales como la navegación, ingreso y presentación (logos y aspectos visuales) que contendrá al resto de páginas de la aplicación.**	

Tarea de Ingeniería	
Número Tarea: 20	**Número Historia de Usuario:** 10
Nombre Tarea: Pagina web de visualizado de pruebas y pacientes	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: Diseño y programación de una página web en la que apreciaremos los pacientes que han dado positivo en una alerta. De estos se podrá hacer una vista en detalle donde apreciaremos cuales han sido los resultados de las pruebas y de estas cuales han sido las desencadenantes de la alerta. Además, por cada paciente se podrá marcar que la alerta ha sido vista por un médico, desecharla o confirmar como caso de Sepsis.	

Tarea de Ingeniería	
Número Tarea: 21	**Número Historia de Usuario:** 11
Nombre Tarea: Archivo de configuración de SepSOS	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: Creación de un archivo de configuración para la aplicación SepSOS, con las siguientes modificaciones posibles: -Inserción, modificación y borrado de pruebas. -Inicio, reinicio, detención y pausa de la aplicación. -Modificación de variables globales como ventana de permanencia de datos o valor umbral de la prueba. -Forzado manual de acciones.	

Tarea de Ingeniería	
Número Tarea: 22	**Número Historia de Usuario:** 12
Nombre Tarea: Uso del LOG	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Añadir a cada acción de la web que desencadene navegabilidad una inserción en la base de datos de SepSOS (en la tabla de LOG) indicando quien es el desencadenante de tal acción, cuando la ha realizado y de que se trata.**	

Tarea de Ingeniería	
Número Tarea: 23	**Número Historia de Usuario:** 13
Nombre Tarea: Inserción en el Histórico de Alertas	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Ante una alerta confirmada, automáticamente se debe insertar en tablas de estructura indenticas a la de datos de pacientes y resultados de pruebas las correspondientes a esa alerta para su posterior consulta una vez finalizada la alerta**	

Tarea de Ingeniería	
Número Tarea: 24	**Número Historia de Usuario:** 13
Nombre Tarea: Consulta de Históricos	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Escribir una consulta a las tablas relacionadas con pacientes y pruebas archivadas pasa su consulta.**	

Tarea de Ingeniería	
Número Tarea: 25	**Número Historia de Usuario:** 13
Nombre Tarea: Página Web de Históricos	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: **Diseño y programación de una web con estructura semejante a la de visualización de alertas solo que esta contendrá datos correspondientes a alertas archivadas.**	

	Tarea de Ingeniería
Número Tarea: 26	**Número Historia de Usuario:** 14
Nombre Tarea: Escribir Querys para la explotación estadística de datos	
Tipo de Tarea : Opcional - Mejora	
Programador Responsable: Sergio	
Descripción: Redactar consultas a las tablas de datos de pruebas y pacientes con parámetros de entrada variables para que puedan ser llamadas desde la web de estadísticas y muestren resultados con los filtros deseados.	

	Tarea de Ingeniería
Número Tarea: 27	**Número Historia de Usuario:** 14
Nombre Tarea: Página web de estadísticas	
Tipo de Tarea : Opcional - Mejora	
Programador Responsable: Alberto	
Descripción: Diseño y programación de una página web donde el usuario podrá aplicar filtros a las consultas a la base de datos y obtener sus resultados.	

Tarea de Ingeniería	
Número Tarea: 28	**Número Historia de Usuario:** 15
Nombre Tarea: Algoritmo de Borrado	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Escritura del algoritmo responsable de mantener los datos válido dentro de la ventana definida, es decir, borrado de los que no estén dentro de dichos márgenes.**	

Tarea de Ingeniería	
Número Tarea: 29	**Número Historia de Usuario:** 17
Nombre Tarea: Escritura del algoritmo de automatización	
Tipo de Tarea : Desarrollo	
Programador Responsable: Sergio	
Descripción: **Escritura de una tarea programada que se encargue de lanzar los algoritmos de consulta de pruebas cada cierto tiempo.** **A continuación se lanzará el algoritmo de análisis y al finalizar, el de borrado.**	

Tarea de Ingeniería	
Número Tarea: 30	**Número Historia de Usuario:** 16
Nombre Tarea: Algoritmo de autenticación	
Tipo de Tarea : Desarrollo	
Programador Responsable: Alberto	
Descripción: **Escritura de un método que ante los datos de usuario y contraseña introducidos en los campos de *Login*, compruebe contra la GPO del Active Directory del dominio del hospital PESET que el usuario está habilitado para usar SepSOS.**	

Base de Datos de la Aplicación SEPSOS:

La base de datos que utilizamos en nuestra aplicación es una Base de Datos *SQLServer 2008* que interactúa con la base de datos del Hospital del tipo IBM Informix que está incluido dentro de *Orion Clinic*, a través de nuestros Módulos Detección y Extracción.

Nos hemos decidido a utilizar una base de datos propia y SQLServer 2008 por muchos motivos:

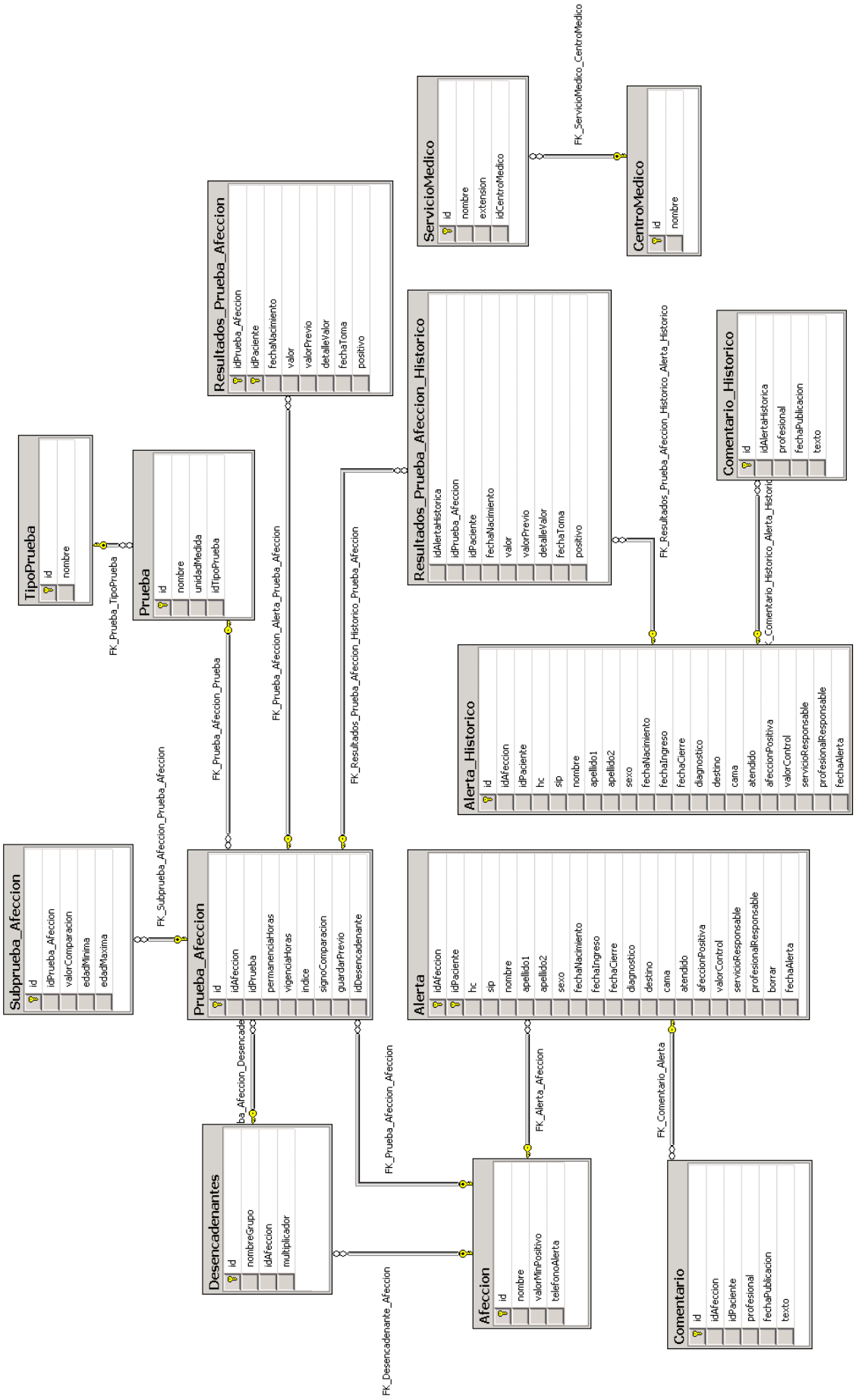
- Las licencias y recursos que posee el Hospital, que nos permite trabajar con software exclusivo y potente de forma gratuita.

- La base de *Orion Clinic* es enorme, muy pesada y con actualizaciones constantes. Además es un proyecto ajeno a nosotros sobre el cual no teníamos derecho.

- Tener una base de datos propia aporta escalabilidad, si, por ejemplo, se deseara portar la aplicación a otro hospital que no trabajara con *Orion Clinic*.

- El lenguaje de programación (C#), el sistema operativo del servidor (Windows Server 2003) y el entorno de programación (Visual Studio) utilizado nos aporta ese extra de integración con una base de datos *SQLServer 2008*.

A continuación se puede observar la estructura de la base de datos que ha derivado del diagrama UML mostrado anteriormente:


ServicioMedico
id
nombre
extension
idCentroMedico
FK_ServicioMedico_CentroMedico
CentroMedico
id
nombre
Resultados_Prueba_Afeccion
idPrueba_Afeccion
idPaciente
fechaNacimiento
valor
valorPrevio
detalleValor
fechaToma
positivo
Resultados_Prueba_Afeccion_Historico
idAlertaHistorica
idPrueba_Afeccion
idPaciente
fechaNacimiento
valor
valorPrevio
detalleValor
fechaToma
positivo
FK_Resultados_Prueba_Afeccion_Historico_Alerta_Historico
Comentario_Historico
id
idAlertaHistorica
profesional
fechaPublicacion
texto
FK_Comentario_Historico_Alerta_Historic
TipoPrueba
id
nombre
FK_Prueba_TipoPrueba
Prueba
id
nombre
unidadMedida
idTipoPrueba
FK_Prueba_Afeccion_Alerta_Prueba_Afeccion
FK_Prueba_Afeccion_Prueba
FK_Resultados_Prueba_Afeccion_Historico_Prueba_Afeccion
Alerta_Historico
id
idAfeccion
idPaciente
hc
sip
nombre
apellido1
apellido2
sexo
fechaNacimiento
fechaIngreso
fechaCierre
diagnostico
destino
cama
atendido
afeccionPositiva
valorControl
servicioResponsable
profesionalResponsable
fechaAlerta
Subprueba_Afeccion
id
idPrueba_Afeccion
valorComparacion
edadMinima
edadMaxima
FK_Subprueba_Afeccion_Prueba_Afeccion
Prueba_Afeccion
id
idAfeccion
idPrueba
permanenciaHoras
vigenciaHoras
indice
signoComparacion
guardarPrevio
idDesencadenante
Alerta
idAfeccion
idPaciente
hc
sip
nombre
apellido1
apellido2
sexo
fechaNacimiento
fechaIngreso
fechaCierre
diagnostico
destino
cama
atendido
afeccionPositiva
valorControl
servicioResponsable
profesionalResponsable
borrar
fechaAlerta
ba_Afeccion_Desencade
FK_Prueba_Afeccion_Afeccion
FK_Alerta_Afeccion
FK_Comentario_Alerta
Desencadenantes
id
nombreGrupo
idAfeccion
multiplicador
Afeccion
id
nombre
valorMinPositivo
telefonoAlerta
Comentario
id
idAfeccion
idPaciente
profesional
fechaPublicacion
texto
FK_Desencadenante_Afeccion

Query para obtener pacientes de Orion (Consulta Parametrizada):

```
SELECT
DISTINCT p.paci_key AS pacikey,
upper(NVL(p.paci_nd, 'sin historia clinica'))AS hc,
e.epi_key AS episodio,
e.epi_ini_episodio AS fechaingreso,
upper(NVL(e.epi_cierre_episodio, 'sin fecha cierre')) AS fechacierre,
e.tipo_epi_key AS tipoEpisodio,
upper(NVL(p.paci_sip, 'sin sip')) AS sip,
upper(s.ctma_desc) AS sexo,
p.paci_nombre AS nombre,
p.paci_primer_apellido AS ap1,
upper(NVL(p.paci_segundo_apellido, 'sin segundo apellido')) AS ap2,
p.paci_fecha_nacimiento AS fechanacimiento,
upper(NVL(trim(u.usu_primer_apellido || ' '
   ||NVL(u.usu_segundo_apellido,'') || ', ' || u.usu_nombre), 'sin
   profesional')) AS profesional,
NVL(se.secc_long_desc, 'sin servicio responsable') AS servicio,
upper(NVL(e.epi_sospecha_diag,'sin diagnostico')) AS diagnostico,
upper(NVL(ds.ctma_desc,'sin destino')) AS destino,
upper(NVL(c.cod_cama_key,'sin cama')) AS cama

FROM
campos_tablas_maestras s INNER JOIN pacientes p
ON s.ctma_key = p.paci_sexo
INNER JOIN ((episodios e
LEFT JOIN ocupacion_camas oc
ON e.epi_key = oc.epi_key AND oc.cama_ocupada = 1 AND oc.fecha_ocupa_fin IS NULL
LEFT JOIN camas c
ON oc.cama_key = c.cama_key
LEFT JOIN campos_tablas_maestras ds
ON e.mot_alta_key = ds.ctma_key  AND ds.ctma_desc != 'EXITUS')
LEFT JOIN contactos co
ON e.epi_key = co.epis_key AND co.cont_is_activo = 1 and (co.tico_key=2 or
co.tico_key=4)
LEFT JOIN secciones se
ON co.secc_key = se.secc_key
LEFT JOIN contacto_hospitalizacion ch
ON co.cont_key = ch.cont_key
LEFT JOIN ((profesionales pr
LEFT JOIN usuarios u
ON pr.usu_key = u.usu_key))
ON ch.prof_medico_adjunto_key = pr.prof_key)
ON p.paci_key = e.paci_key

WHERE
(e.tipo_epi_key = 'TEPIHOSP'
OR e.tipo_epi_key ='TEPIURG')
AND e.paci_key||\"_\"||e.epi_ini_episodio IN
(SELECT e.paci_key||\"_\"||MAX(epi_ini_episodio)

FROM episodios e

WHERE
e.paci_key IN PACIENTES
group by e.paci_key)
```

*Siendo PACIENTES una lista de “paci_key”.

Query para obtener pruebas de laboratorio de Orion (Consulta Parametrizada):

```
SELECT
DISTINCT
p.paci_key AS idPaciente,
p.paci_fecha_nacimiento AS fechaNacimiento,
upper(dl.det_lab_codi_micro) AS prueba,
upper(dl.det_lab_valor) AS valor,
'Sin Detalle' as detalleValor,
extend(dl.det_lab_fecha, YEAR TO minute) AS extraccion

FROM
pacientes p JOIN episodios e
ON p.paci_key = e.paci_key " +
JOIN informes i ON e.epi_key = i.epi_key " +
JOIN resultados_laboratorio rl ON rl.informe_key = i.informe_key " +
JOIN determinaciones_laboratorio dl ON rl.res_lab_key = dl.res_lab_key " +
JOIN solicitudes_laboratorio sl ON sl.soli_key = rl.res_lab_id_soli " +
INNER JOIN
(SELECT  p.paci_key,
dl.det_lab_codi_micro,
max(dl.det_lab_key) as toma
FROM pacientes p
JOIN episodios e ON p.paci_key = e.paci_key
JOIN informes i ON e.epi_key = i.epi_key
JOIN resultados_laboratorio rl ON rl.informe_key = i.informe_key
JOIN determinaciones_laboratorio dl ON rl.res_lab_key = dl.res_lab_key
JOIN solicitudes_laboratorio sl ON sl.soli_key = rl.res_lab_id_soli

WHERE
((e.tipo_epi_key = 'TEPIURG' and (e.epi_cierre_episodio is null or
e.mot_alta_key='DESTU06')) OR e.tipo_epi_key = 'TEPIHOSP')
AND extend(dl.det_lab_fecha, YEAR TO minute) BETWEEN
TO_DATE('ULTIMACOMPROBACION','%d/%m/%Y %H:%M:%S') and
TO_DATE('AHORA','%d/%m/%Y %H:%M:%S')
AND dl.det_lab_codi_micro IN CADENAPRUEBAS
GROUP BY
p.paci_key, dl.det_lab_codi_micro) lab
ON dl.det_lab_key=lab.toma
GROUP BY
p.paci_key, p.paci_fecha_nacimiento, dl.det_lab_valor, sl.soli_lab_fecha_extraccion,
prueba, dl.det_lab_valor, dl.det_lab_fecha
 HAVING dl.det_lab_valor matches '*[0-9]*'
```

*Siendo ULTIMACOMPROBACION la fecha y hora de la última comprobación realizada.

**Siendo AHORA la fecha y hora del momento actual.

***Siendo CADENAPRUEBAS una lista de pruebas.

Query para obtener pruebas de constantes de Orion (Consulta Parametrizada):

```
SELECT
p.paci_key AS idPaciente,
p.paci_fecha_nacimiento AS fechaNacimiento,
upper(mc.m_cnte_nombre) AS prueba,
t.toma_valor AS valor, " +
'Sin Detalle' as detalleValor,
extend(t .toma_fecha + t.toma_hora units minute, YEAR TO minute) AS extraccion

FROM
pacientes p
JOIN episodios e ON p.paci_key = e.paci_key
JOIN contactos c ON c.epis_key = e.epi_key
JOIN toma_constantes t ON c.cont_key = t.cont_key
JOIN maestro_constantes mc ON mc.m_cnte_key = t.toma_cnte_key
LEFT JOIN campos_tablas_maestras v ON t.toma_valor = v.ctma_key
INNER JOIN
(SELECT p.paci_key AS idPaciente,
date(p.paci_fecha_nacimiento) AS fechaNacimiento,
upper(mc.m_cnte_nombre) AS prueba,
max(t.toma_key) as toma
FROM pacientes p
JOIN episodios e ON p.paci_key = e.paci_key
JOIN contactos c ON c.epis_key = e.epi_key
JOIN toma_constantes t ON c.cont_key = t.cont_key
JOIN maestro_constantes mc ON mc.m_cnte_key = t.toma_cnte_key
LEFT JOIN campos_tablas_maestras v ON t.toma_valor = v.ctma_key

WHERE
((e.tipo_epi_key = 'TEPIURG' AND (e.epi_cierre_episodio IS NULL OR e.mot_alta_key =
'DESTU06')) OR e.tipo_epi_key = 'TEPIHOSP')
AND mc.m_cnte_nombre IN CADENAPRUEBAS
AND (extend(t .toma_fecha + t.toma_hora units minute, YEAR TO minute))
BETWEEN TO_DATE('ULTIMACOMPROBACION','%d/%m/%Y %H:%M:%S') and
TO_DATE('AHORA','%d/%m/%Y %H:%M:%S')
GROUP BY idPaciente, fechaNacimiento, prueba) const "+
on t.toma_key=const.toma
GROUP BY
p.paci_key, p.paci_fecha_nacimiento, mc.m_cnte_nombre, t.toma_valor,extraccion
HAVING t.toma_valor matches '*[0-9]*'
```

*Siendo ULTIMACOMPROBACION la fecha y hora de la última comprobación realizada.

**Siendo AHORA la fecha y hora del momento actual.

***Siendo CADENAPRUEBAS una lista de pruebas.

Vista de Relaciones en Orion Clinic:

Estas son las relaciones de las tablas de la base de datos Informix de Orion, de la cual extraemos los datos. Solo tenemos las conexiones de las tablas que utilizamos para obtener las pruebas y los pacientes, ya que como hemos comentado anteriormente, Orion es enorme, con más de 700 tablas.

Implementación y Código:

El lenguaje utilizado ha sido C# por los motivos expuestos anteriormente (licencias y recursos), usando un entorno de programación Visual Studio 2010, con el cual teníamos cierta experiencia, pues ha sido empleado en las prácticas de muchas asignaturas de la Universidad Politécnica de Valencia.

C# es un lenguaje orientado a objetos que desciende de C y C++. Existe mucha documentación sobre este lenguaje, pues es un lenguaje muy de moda. Además, es versátil, intuitivo y similar a otros lenguajes orientados a objetos (Java por ejemplo). Contábamos con experiencia con este lenguaje, lo cual nos hizo decantarnos por él a la hora del desarrollo.

El único requisito para su funcionamiento es la instalación obligatoria en el equipo del *NET.FRAMEWORK* cuya versión va acorde a los métodos y funciones utilizados en el código. No es necesario que el cliente que vaya a utilizar la aplicación tenga instalado el framework correspondiente, pues puede adjuntarse el paquete de instalación junto al programa.

Nuestra aplicación se divide en tres proyectos de Visual Studio (DANA, SepSOS y Web).

Mostraremos el código de los métodos más importantes y complejos, obviando métodos triviales o complementarios a las funciones principales que realiza la clase.

Proyecto DANA (Detección de Afecciones y Notificación de Alertas):

DANA es un servicio Windows para la detección de posibles casos de Sepsis, que utiliza las bibliotecas de Extracción de Datos, Tratamiento de nuestra Base de Datos y Control de Flujo de Datos.

Mediante estas bibliotecas, podría funcionar igualmente mediante un Formulario Windows, por ejemplo. Pero debido a que en el Hospital les interesaba un Servicio para no tener una sesión iniciada en el Servidor y fue lo que nos pidieron, fue lo que realizamos.

```
namespace DANA
{
    public partial class Servicio : ServiceBase
    {
        private Deteccion deteccion;
        private Borrado borrado;
        private AlarmClock alarma;
        private Queries sepSOS;
        private bool detener;
        private bool borrar;
        private bool finRutina;
        private int totalComprobaciones;
        private int comprobacionesActuales;
        private Thread thread;
        private DateTime ultimaFechaComprobada;

        public Servicio()...

        void alarma_Alarm(object sender, EventArgs e)...

        private void manejadorBorrado()...

        private void limpieza()...

        private void InitializeComponent()...

        protected override void OnStart(string[] args)...

        public void fakeStart()...

        private void resetThread()...

        protected override void OnStop()...

        protected override void OnShutdown()...

        public void arrancar(DateTime fecha)...

        private void rutina()...

        private void programarSiguienteBorrado()...
    }
}
```

Método Constructor:

Se encarga de llamar a *InitializeComponent()* y le da valor a los atributos de la clase *ServiceBase*.

Manejadores de Eventos:

Podemos encontrar *OnStart*, *OnStop*, *OnShutDown*... que se ejecutan al Iniciar el Servicio, Detenerlo o cuando el servidor se apaga.

Alarm_Alarm tiene como objetivo el control del evento Alarma de la clase del mismo nombre que le indica a la aplicación que es momento de detener el Análisis y lanzar el Borrado Diario.

Método Arrancar:

Se encarga de la puesta a punto del programa. Manda leer el archivo de configuración y da valor a los atributos de los objetos. Lanza el hilo global cuyo método asignado es *Rutina*.

Método ManejadorBorrado:

Es el responsable del uso de la parte de la biblioteca de control del flujo que controla el archivado y borrado de alertas.

Método FakeStart:

Debido a que no se puede hacer *Debug* de un Servicio sin instalarlo, este método se encarga de simular el comportamiento del servicio llamando al método de rutina.

Método Limpieza:

Debido a un posible elevado consumo de memoria, y aunque el *Net.Framework* tiene su propio recolector de basura que elimina los objetos no referenciados, este método fuerza los objetos más pesados a “null” y llamamos al recolector de basura para asegurarnos de que son borrados. Es una forma de asegurarnos de que se va a liberar dicho espacio.

Método Rutina:

Método de control principal en el que cada intervalo de tiempo marcado en el archivo de configuración, realiza una iteración solicitando una nueva fase de análisis llamando al Módulo de Detección.

Comprueba si es necesario borrar o detenerse antes de realizar dicha solicitud.

```
private void rutina()

{
    finRutina = false;

    while (!detener)
    {
         if (comprobacionesActuales < totalComprobaciones)
         {
             comprobacionesActuales++;
         }
         else
         {
            while (deteccion.Trabajando)
            {
                  System.Diagnostics.Debug.WriteLine("atascado en
                  trabajando");
                  Thread.Sleep(1000);
            }
            Configuracion.UltimaFechaComprobada =
            deteccion.UltimaFechaComprobada.ToString();
            ultimaFechaComprobada = deteccion.UltimaFechaComprobada;
            comprobacionesActuales = 0;
            System.Diagnostics.Debug.WriteLine("tarea lanzada");
             sepSOS.insertarLOG("Lanzar Deteccion", "Ultima comprobación: "
             + deteccion.UltimaFechaComprobada.ToString(), "DANA",
             DateTime.Now);

            deteccion.tarea();

            }
                  Thread.Sleep(60000);
         }

         if (borrar)
            {
                manejadorBorrado();
            }

            finRutina = true;
 }
```

Configuración del Servicio DANA:

DANA dispone de un archivo de configuración con extensión ".config" en el cual aparecen los atributos parametrizables utilizados durante la ejecución del programa.

```
  <appSettings>
    <add key="miHospital" value="Doctor Peset Aleixandre" />
    <add key="tiempoEsperaLanzarDeteccionMilisegundos" value="600000" />
    <add key="tiempoEsperaLanzarBorradoHoras" value="24" />
    <add key="tiempoPermanenciaPacientesDeAltaDias" value="2" />
    <add key="tiempoEsperaPurgadoDias" value="7" />
    <add key="ultimaFechaComprobada" value="29/05/2013 00:30:00" />
    <add key="correoEmisorInicidencias" value="xxxxxxxxxxxxxxx@gva.es" />
    <add key="correoDestinatarioIncidencias" value="xxxxxxxxxx@gva.es" />
    <add key="enviarSMS" value="true" />
    <add key="enviarAlertasAlCorreo" value="true" />
    <add key="smsSoloTelefonosAfecciones" value="true" />
  </appSettings>

  <connectionStrings>
    <add name="AccesoSepSOSbd.Properties.Settings.sepsosConnectionString"
        connectionString="Data Source=SRVALERTAS;Initial
Catalog=DANA;Integrated Security=True"
        providerName="System.Data.SqlClient" />
    <add name="AccesoDatosMedicos.Properties.Settings.cadenaConexionOrion"
           connectionString="Dsn=Orion" providerName="System.Data.Odbc"
/>
  </connectionStrings>
```

Los atributos más relevantes son:

- *tiempoEsperaLanzarDeteccionMilisegundos*: Espera 10 minutos antes de lanzar la siguiente iteración de detección.
- *tiempoEspereLanzarBorradoHoras*: Cada 24 horas se lanza el borrado de alertas marcadas a archivar.
- *tiempoPermanenciaPacientesDeAltaDias*: Hasta 2 días después de ser dado de alta un paciente, no puede ser borrada su alerta.
- *tiempoEsperaPurgadoDias*: Cuando los resultados tienes 7 días de antigüedad, pasan a ser borrados.
- *correoEmisor(Destinatario)Incidencias*: Si ocurre un fallo en el servicio, emisor y receptor de un correo de aviso del fallo.
- *enviarSMS*: Activa/Desactiva las alertas al móvil.
- *enviarAlertasAlCorreo*: Activa/Desactiva las alertas al correo.
- *smsSoloTelefonosAfecciones*: Si la alerta va solo al servicio responsable de la Sepsis o también al servicio responsable del paciente (por ejemplo, si el paciente está en urgencias, el SMS llega al servicio de Urgencias además de los responsables de la Sepsis).

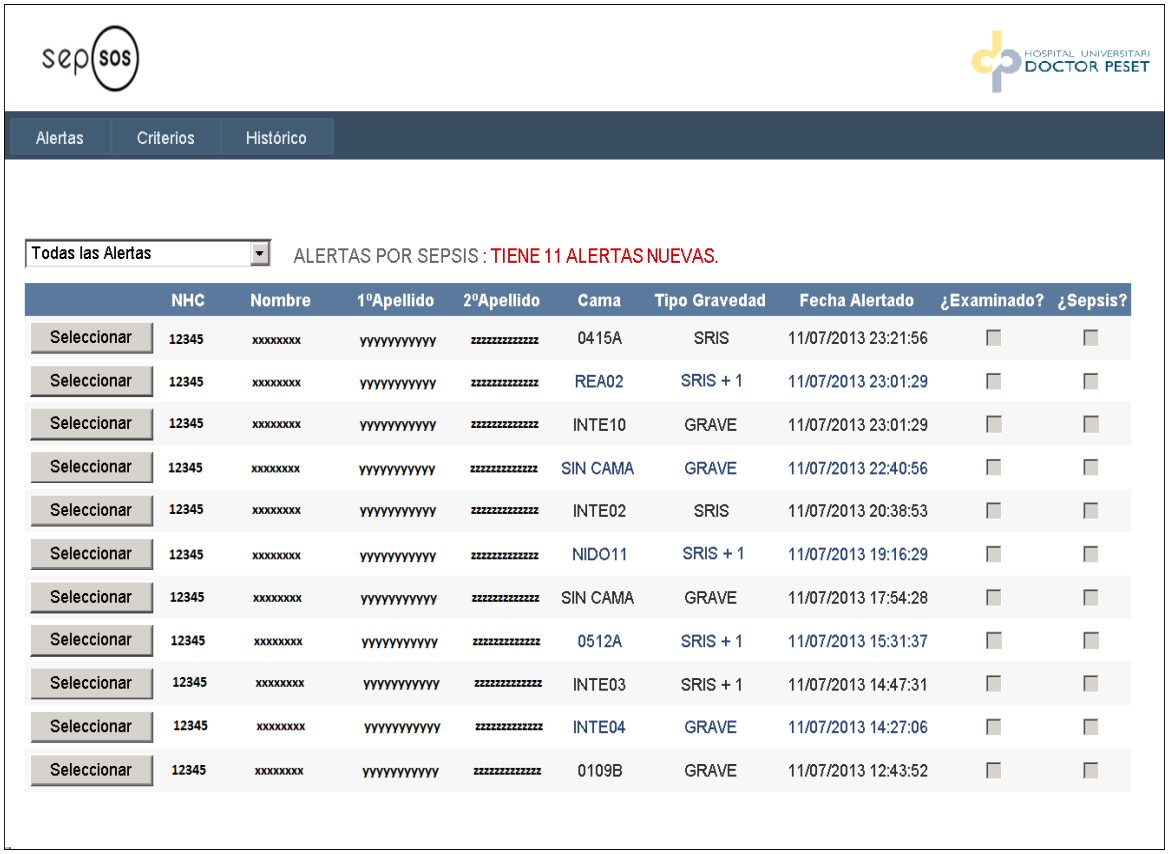
ConnectionStrings contiene las cadenas de conexión a la base de datos de Orion y la base de datos de la aplicación SepSOS. Si se moviesen o cambiasen las bases de datos, habría que modificar las cadenas.

Proyecto Página WEB SepSOS:

La página web de SepSOS es utilizada para para la gestión y visualización de las alertas, correspondiéndose con el Módulo de Visualización.Dicha página web es uno de los tres proyectos de Visual Studio que forman el proyecto SepSOS.

El portal ha sido implementado en ASP.NET para continuar el uso de Visual Studio durante el desarrollo de nuestra aplicación.
ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Se usa para construir sitios web dinámicos, aplicaciones web y servicios web XML.

Nuestro sitio web está formado por una página maestra (*Site.Master*) que contiene los elementos estáticos de la página y las tres páginas que forman la web de Sepsos (*Alertas.aspx*, *Criterios.aspx*, *Historico.aspx*).

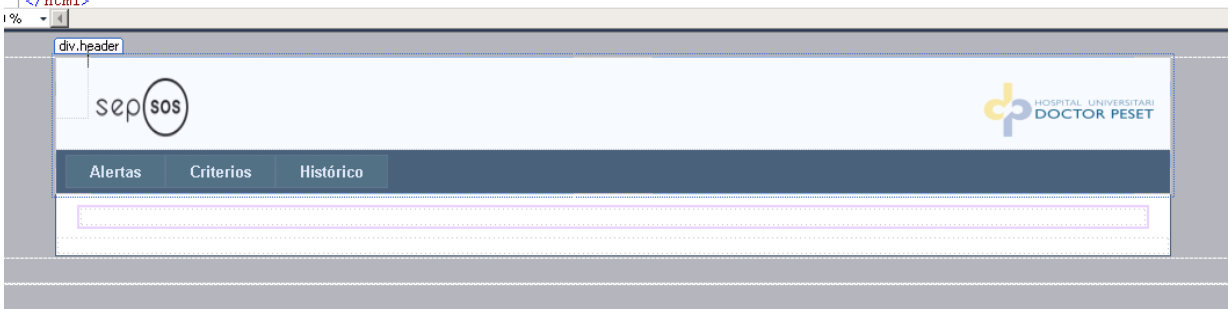


Página Maestra Site.Master:

Página Maestra encargada de contener todos los elementos estáticos de nuestro sitio web que no cambian durante la navegación(Logo, Menú, Títulos, etc...).

Está escrita en HTML, de forma simple, incuyendo información visual sobre el posicionamiento de los logos, alineación del texto, elementos del menú, colos de fondo...

```
<body>
    <form runat="server">
    <div class="page">
        <div class="header">
            <div class="title">...</div>
            <div class="loginDisplay" style="background-color: #FFFFFF">
                <asp:Image ID="Image2" runat="server" Height="60px" ImageAlign="Left"
                    ImageUrl="/logoSepsos.png" Width="90px" />
                <asp:Image ID="Image1" runat="server" Height="49px" ImageUrl="logoPeset.png"
                    style="margin-top: 8px" Width="154px" />
            </div>
            <div class="clear hideSkiplink">
                <asp:Menu ID="NavigationMenu" runat="server" CssClass="menu"
                    EnableViewState="false" IncludeStyleBlock="false" Orientation="Horizontal"
                    Font-Bold="True">
                    <Items>
                        <asp:MenuItem NavigateUrl="~/Alertas.aspx" Text="Alertas" Value="Alertas"/>
                        <asp:MenuItem NavigateUrl="~/Criterios.aspx" Text="Criterios" Value="Criterios"/>
                        <asp:MenuItem NavigateUrl="~/Historico.aspx" Text="Histórico" Value="Historico">
                        </asp:MenuItem>
                    </Items>
                </asp:Menu>
            </div>
        </div>
        <div class="main">...</div>
        <div class="clear">
        </div>
    </div>
    <div class="footer">...</div>
    </form>
</body>
</html>
```



Página Alertas.aspx:

```
public partial class Alertas : System.Web.UI.Page
{
    private const int IDAFECCION = 1;

    private Queries sepSOSbd;

    private DataTable todasAlertas;
    private DataTable alertasFiltradas;
    private AccesoSepSOSbd.dsSepSOS.ResultadosPruebasPresentablesDataTable todosResultadosPruebas;
    private AccesoSepSOSbd.dsSepSOS.ComentarioDataTable todosComentarios;
    private AccesoSepSOSbd.dsSepSOS.ValoresComparacionDataTable comparaciones;
    private string selectAlertas;
    private int indiceLinea;
    private bool estaModoArchivadas;

    protected void Page_Load(object sender, EventArgs e)...
    private void desocultar(bool x)...
    private DataTable crearDataTable()...
    private string obtenerSRIS(int codigo)...
    private void refresco()...
    private void cambiarBotonSepsis()...
    private void pedirDatosASepSOS()...
    private void bindDataGrids()...
    private void bindGridAlertas()...
    private void bindGridResultadosPruebas()...
    protected void gvAlertas_SelectedIndexChanged(object sender, EventArgs e)...
    private void activarBotones()...
    private DataTable resultadosPruebasPorPaciente(int idPaciente)...
    private string obtenerVigencia(string nombrePrueba, DateTime fechaToma)...
    protected void gvAlertas_PageIndexChanging(object sender, GridViewPageEventArgs e)...
    private void resetSeleccionGridAlertas()...
    private void cerrarAlerta(bool result, object sender, EventArgs e)...
    protected void Button1_Click(object sender, EventArgs e)...
    protected void btnSi_Click(object sender, EventArgs e)...
    private void actualizarLabels(DataRow fila)...
    private void actualizarComentarios(int idPaciente)...
    protected void btnVisto_Click(object sender, EventArgs e)...
    protected void btnConfirmarSepsis_Click(object sender, EventArgs e)...
    protected void BotonComentario_Click(object sender, EventArgs e)...
    private string maquetarTextoComentario(string texto)...
    private void contarAlertasNuevas()...
    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)...
    private void comprobacionExaminado(bool c)...
    private void ocultaBotonesArchivadas()...
    protected void btnBorrar_Click(object sender, EventArgs e)...
}
```

Método PageLoad:

Método que se desencadena cuando se carga Alertas.aspx. Realiza la puesta a punto.

Método Desocultar:

Oculta/Muestra la información adicional de la alerta de un paciente, los botones y los comentarios. Implementa la relación Maestro-Esclavo respecto a la tabla principal con el resto de elementos.

Métodos BindDataGrids, BindGridAlertas, BindGridResultadosPruebas:

BindGridAlertas y BindGridResultadosPruebas les indica a sus respectivas tablas que el origen de datos ha cambiado y de esta manera forzamos el refresco.

BindDataGrids llama a BindGridAlertas y BindGridResultadosPruebas.

Métodos gvAlertas_SelectedIndexChanged y gvAlertas_PageIndexChanged:

gvAlertas_SelectedIndexChanged indica que se ha seleccionado una nueva alerta en la tabla principal y que se tienen que cargar los datos relacionados con la alerta seleccionada.

gvAlertas_PageIndexChanged informa de que se ha cambiado de página en el grid principal y carga las alertas correspondientes a esa página.

Método ActivarBotones:

Activa los botones que se encontraban deshabilitados al seleccionar una alerta o marcarla como vista.

Método ObtenerVigencia:

Averigua la vigencia de las pruebas para mostrarlas junto al resto de valores de las pruebas y así el profesional sabe si debe repetir la prueba o si debe considerarla como válida.

Método CerrarAlerta:

Marca la alerta para que sea archivada en el histórico.

Métodos btnConfirmarSepsis_Click, btnVisto_Click:

btnConfirmarSepsis_Click, al apretar el botón de “Confirmar Sepsis” marca dicha alerta como positiva en Sepsis. Su valor en el campo “AfeccionPositiva” pasa a ser “Verdadero”.

btnVisto_Click, al apretar el botón de "Paciente Examinado" marca dicha alerta como vista. Su valor en el campo "Examinado" pasa a ser "Verdadero".

Método DropDownList1_SelectedIndexChanged:

ComboBox utilizado para filtrar Alertas Nuevas, Alertas Vistas, Marcadas para Archivar o Todas.

Método ObtenerSRIS:

Calcula, a partir del campo "ValorControl" de la alerta, la gravedad del paciente, mostrando dicha gravedad mediante la escala SRIS, o mostrando un *"GRAVE"* si se ha producido un aviso directo o *"Menor que SRIS"* si el paciente se ha recuperado de dicha alerta (pero aún no ha sido borrada dicha alerta).

```
private string obtenerSRIS(int codigo)
        {
            string sris = "SRIS";

            if (codigo >= 1000)
                sris = "GRAVE";
            else if (codigo < 1000 && codigo >= 200)
            {
                int aux = codigo % 100;
                if (aux > 0)
                    sris += " + " + aux;
            }
            else
                sris = "INFERIOR A SRIS";

            return sris;
        }
```

Método PedirDatosASepSOS:

Solicita a la BD de SepSOS los datos de los pacientes, los resultados de pruebas y los comentarios utilizando un escenario desconectado y una sola consulta.

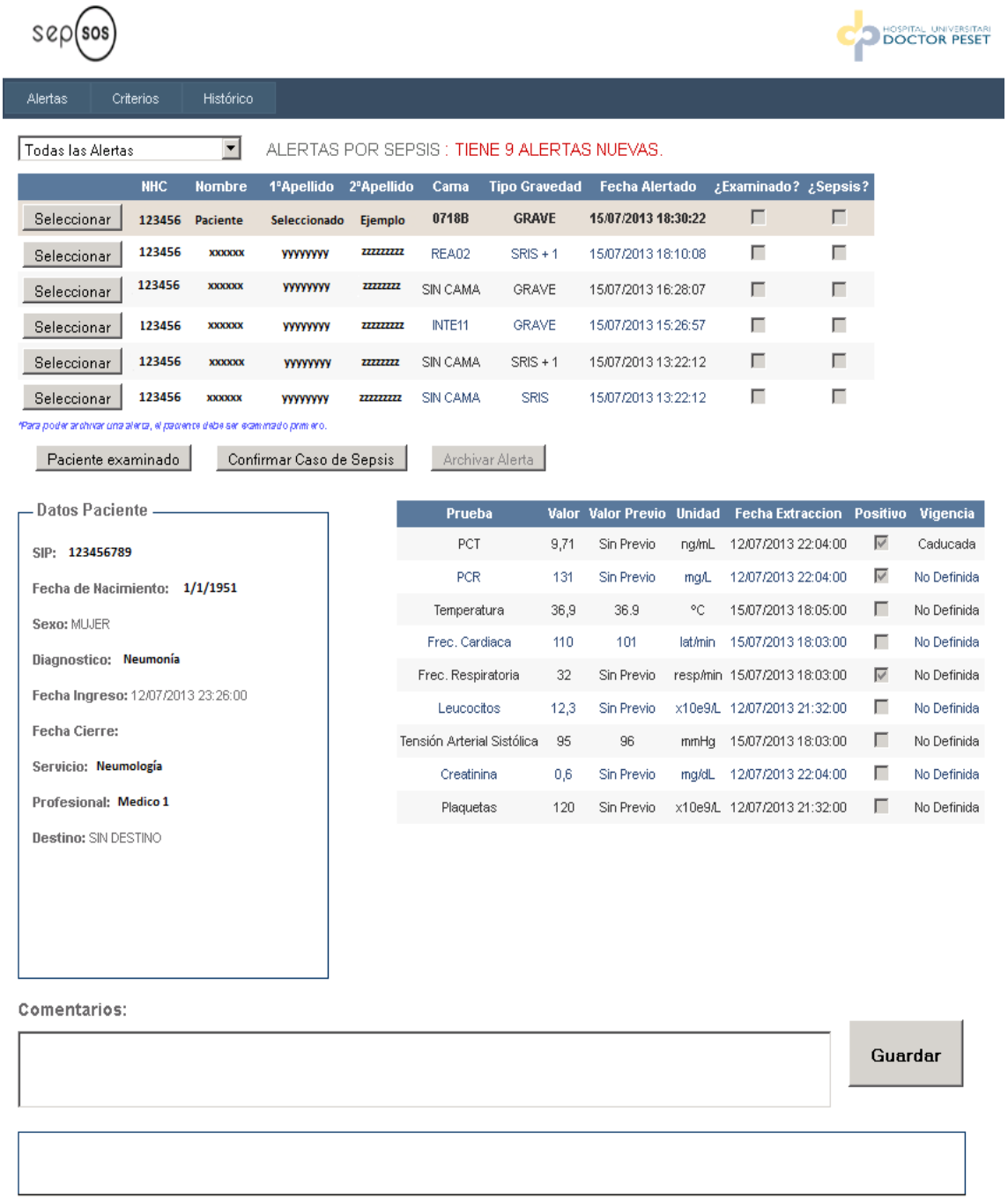
```
private void pedirDatosASepSOS()
        {
           todasAlertas = sepSOSbd.getAlertas();
            try
            {
                DataView aux2 = new
              DataView(todasAlertas.Select(selectAlertas).CopyToDataTable());
                aux2.Sort = "fechaAlerta DESC";
                alertasFiltradas = aux2.ToTable();
            }
            catch (Exception)
            {
                alertasFiltradas = new DataTable();
            }
            todosResultadosPruebas =
              sepSOSbd.getResultadosPruebasPorAfeccionPresentables(IDAFECCION);
            todosComentarios = sepSOSbd.getComentariosPorAfeccion(IDAFECCION);
            comparaciones = sepSOSbd.getValoresComparacionPruebas(IDAFECCION);
        }
```

Método CrearDataTable:

Crea y maqueta la tabla principal de alertas utilizando los valores obtenidos del método *PedirDatosASepSOS*.

```
private DataTable crearDataTable()
      {
          DataTable tablaAlertas = new DataTable();

          tablaAlertas.Columns.Add("NHC", typeof(int));
          tablaAlertas.Columns.Add("Nombre", typeof(string));
          tablaAlertas.Columns.Add("1ºApellido", typeof(string));
          tablaAlertas.Columns.Add("2ºApellido", typeof(string));
          tablaAlertas.Columns.Add("Cama", typeof(string));
          tablaAlertas.Columns.Add("Tipo Gravedad", typeof(string));
          tablaAlertas.Columns.Add("Fecha Alertado", typeof(string));
          tablaAlertas.Columns.Add("¿Examinado?", typeof(bool));
          tablaAlertas.Columns.Add("¿Sepsis?", typeof(bool));

          foreach(DataRow x in alertasFiltradas.Rows)
          {
              DataRow aux = tablaAlertas.NewRow();

              aux["NHC"] = x["hc"];
              aux["Nombre"] = x["nombre"];
              aux["1ºApellido"] = x["apellido1"];
              aux["2ºApellido"] = x["apellido2"];
              aux["Cama"] = x["cama"];
              aux["Tipo Gravedad"] =
                  obtenerSRIS(Convert.ToInt32(x["valorControl"]));
              aux["Fecha Alertado"] = x["fechaAlerta"];
              aux["¿Examinado?"] = x["atendido"];
              aux["¿Sepsis?"] = x["afeccionPositiva"];

              tablaAlertas.Rows.Add(aux);
          }

          return tablaAlertas;
      }
```

Muestra de la página web de Alertas:

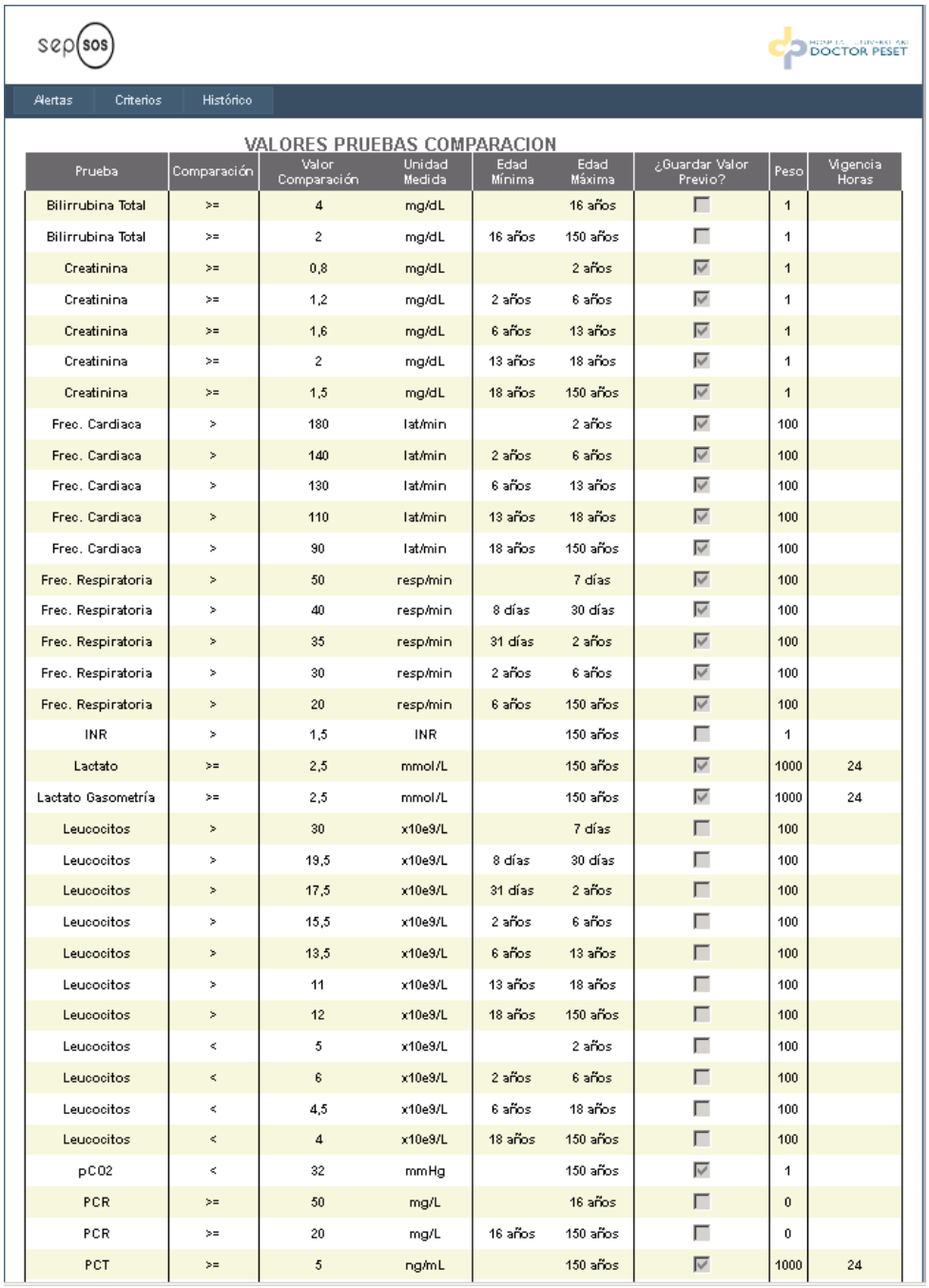


Página Criterios.aspx:

Esta web es una página estática que simplemente muestra los valores de comparación utilizados por la aplicación a la hora de diagnosticar un posible caso de Sepsis.

Solo es utilizada para consultas por parte de los médicos, y contiene la tabla de la página 12.

Muestra de la página web de Criterios:



VALORES PRUEBAS COMPARACION

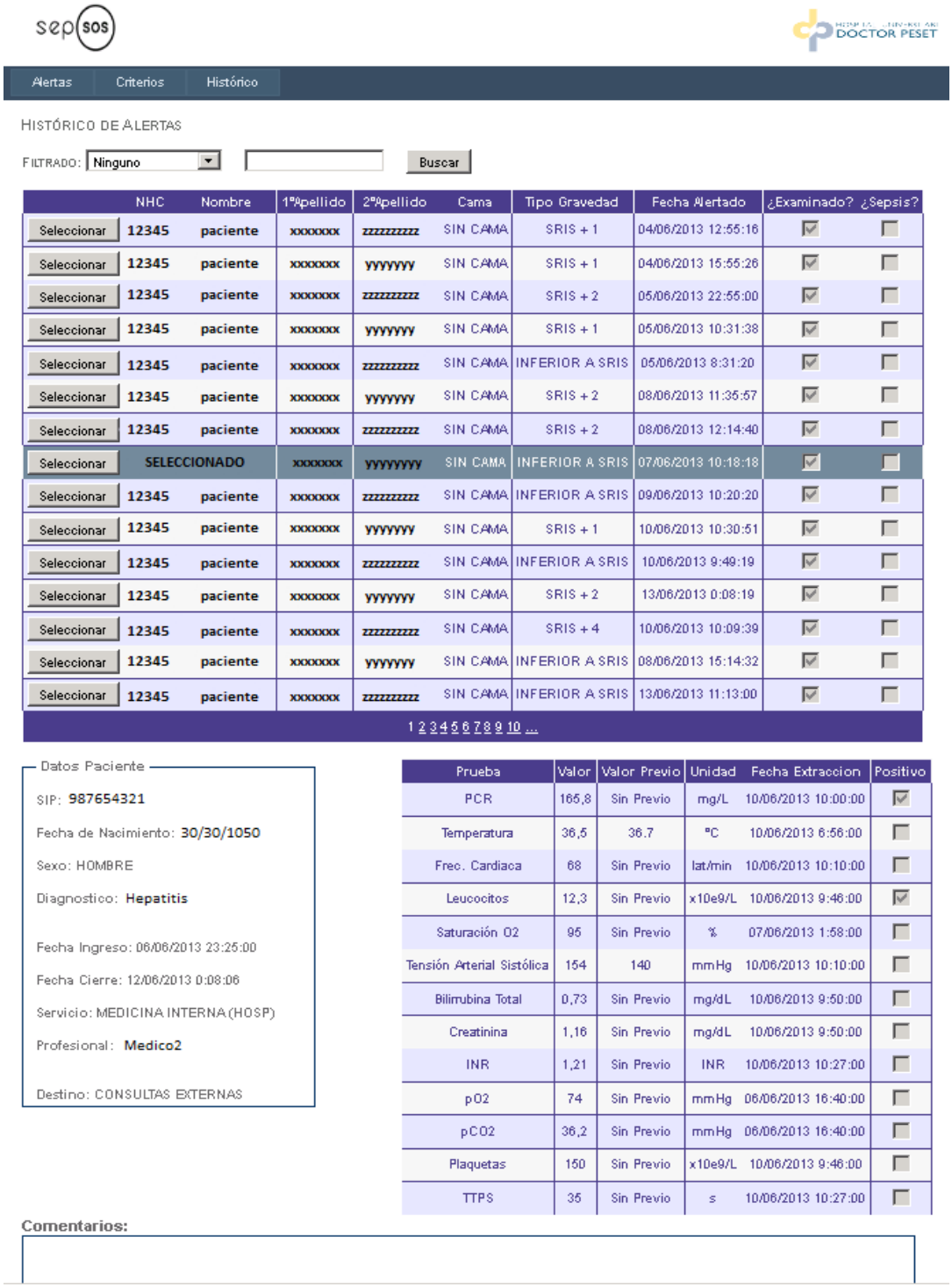
Prueba	Comparación	Valor Comparación	Unidad Medida	Edad Mínima	Edad Máxima	¿Guardar Valor Previo?	Peso	Vigencia Horas
Bilirrubina Total	>=	4	mg/dL		16 años	☐	1	
Bilirrubina Total	>=	2	mg/dL	16 años	150 años	☐	1	
Creatinina	>=	0,8	mg/dL		2 años	☑	1	
Creatinina	>=	1,2	mg/dL	2 años	6 años	☑	1	
Creatinina	>=	1,6	mg/dL	6 años	13 años	☑	1	
Creatinina	>=	2	mg/dL	13 años	18 años	☑	1	
Creatinina	>=	1,5	mg/dL	18 años	150 años	☑	1	
Frec. Cardiaca	>	180	lat/min		2 años	☑	100	
Frec. Cardiaca	>	140	lat/min	2 años	6 años	☑	100	
Frec. Cardiaca	>	130	lat/min	6 años	13 años	☑	100	
Frec. Cardiaca	>	110	lat/min	13 años	18 años	☑	100	
Frec. Cardiaca	>	90	lat/min	18 años	150 años	☑	100	
Frec. Respiratoria	>	50	resp/min		7 días	☑	100	
Frec. Respiratoria	>	40	resp/min	8 días	30 días	☑	100	
Frec. Respiratoria	>	35	resp/min	31 días	2 años	☑	100	
Frec. Respiratoria	>	30	resp/min	2 años	6 años	☑	100	
Frec. Respiratoria	>	20	resp/min	6 años	150 años	☑	100	
INR	>	1,5	INR		150 años	☐	1	
Lactato	>=	2,5	mmol/L		150 años	☑	1000	24
Lactato Gasometría	>=	2,5	mmol/L		150 años	☑	1000	24
Leucocitos	>	30	x10e9/L		7 días	☐	100	
Leucocitos	>	19,5	x10e9/L	8 días	30 días	☐	100	
Leucocitos	>	17,5	x10e9/L	31 días	2 años	☐	100	
Leucocitos	>	15,5	x10e9/L	2 años	6 años	☐	100	
Leucocitos	>	13,5	x10e9/L	6 años	13 años	☐	100	
Leucocitos	>	11	x10e9/L	13 años	18 años	☐	100	
Leucocitos	>	12	x10e9/L	18 años	150 años	☐	100	
Leucocitos	<	5	x10e9/L		2 años	☐	100	
Leucocitos	<	6	x10e9/L	2 años	6 años	☐	100	
Leucocitos	<	4,5	x10e9/L	6 años	18 años	☐	100	
Leucocitos	<	4	x10e9/L	18 años	150 años	☐	100	
pCO2	<	32	mmHg		150 años	☑	1	
PCR	>=	50	mg/L		16 años	☐	0	
PCR	>=	20	mg/L	16 años	150 años	☐	0	
PCT	>=	5	ng/mL		150 años	☑	1000	24

Página Historico.aspx:

Web idéntica a la de alertas, con la salvedad de que las alertas no pueden ser borradas ni se pueden añadir nuevos comentarios.

Es un simple histórico donde se almacenan las alertas una vez son archivadas. Simplemente se utiliza para futuras consultas.

Permite un Filtrado por Nº de Historia, Nombre, Apellidos, etc...

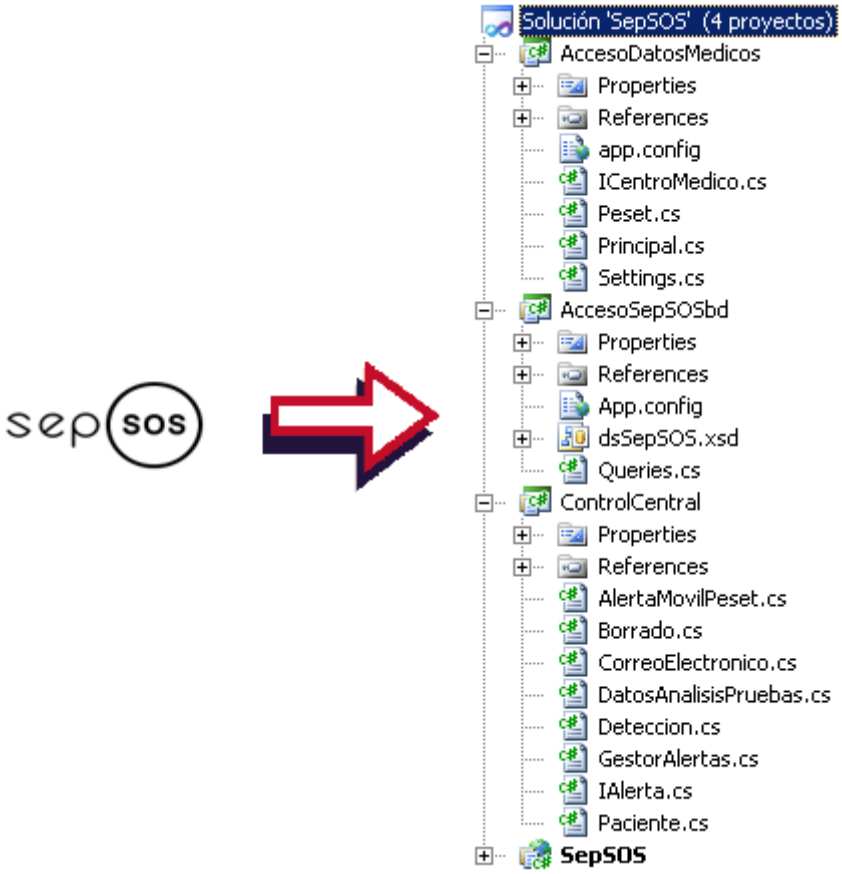


Proyecto SepSOS:

El proyecto SepSOS es el corazón de la aplicación, que contiene los módulos de Detección, Borrado, Extracción y Alerta.

Consta de tres bibliotecas de código (AccesoDatosMedicos, AccesoSepSOSdb, ControlCentral).

El motivo de usar esta estructura es el uso de criterios de calidad software tales como escalabilidad, portabilidad a otros sistemas y reusabilidad.
Todo nuestro proyecto ha sido diseñado pensando en una posible reutilización por parte de otros centros sanitarios, independientemente del sistema que estén utilizando para gestionar sus datos internos.



Biblioteca AccesoDatosMédicos:

Clase ICentroMedico:

Utilizamos esta interfaz a modo de plantilla, para asegurarnos que cualquier centro médico que utilice esta aplicación cumpla con los requisitos de estructura que necesita el módulo de análisis.

De este modo, al garantizar la obtención de datos en base a dicha estructura, conseguimos un alto nivel de escalabilidad, portabilidad y reusabilidad.

```
public interface ICentroMedico
{
        string probarConexion();
        void clearDataSetPruebas();
        void clearDataTablePacientes(int afeccion);
         void clearDataTableCurados();
         DataTable getResultadosPruebas(int tipoPrueba, string pruebas,
         string ultimaComprobacion, string ahora);
         DataTable getDatosPacientes(string pacientes, int afeccion);
        DataTable getPacientesCurados(string pacientes, int tiempoEspera);
}
```

Clase Peset:

Clase que se encarga de dar código a los métodos de la interfaz comentada anteriormente para el caso del uso de la aplicación en el hospital Peset Aleixandre de Valencia.

Entre las características que encontramos, son la conexión a *Orion* (IBM Informix), las consultas para la obtención de datos del paciente, pruebas de laboratorio, constantes, especialistas...

En caso de querer añadir una nueva consulta para la obtención de nuevas pruebas o datos (por ejemplo, datos que provengan de otro departamento como Microbiología), esta es la clase indicada en la que iría.

```
public class Peset: ICentroMedico
{
    private DataSet dataSetOrionPruebas = new DataSet();
    private DataSet dataSetOrionPacientes = new DataSet();
    private OdbcConnection connInformix;
    private OdbcDataAdapter daInformix;
    private int contadorPruebas = 0;

    public static void setCadenaConexion(string cadenaConexion)...

    public string probarConexion()...

    public void clearDataSetPruebas()...

    public void clearDataTablePacientes(int afeccion)...

    public void clearDataTableCurados()...

    //Metodo de cribado
    public DataTable getResultadosPruebas(int tipoPrueba, string pruebas, string ultimaComprobacion, string ahora)...


    //Zona para la seleccion de resultados de pruebas constantes
    public DataTable getResultados (string ultimaComprobacion, string pruebas, string query)...

    private string construirConsultaConstantes(string ultimaComprobacion, string cadenaPruebas, string ahora)...

    private string construirConsultaLaboratorio(string ultimaComprobacion, string cadenaPruebas, string ahora)...

    private string construirConsultaLaboratorioEspecifico(string ultimaComprobacion, string cadenaPruebas, string ahora)...
    //Zona para obtener los datos de los pacientes

    public DataTable getDatosPacientes(string pacientes, int afeccion)...

    private string construirConsultaPacientes(string pacientes)...

    public DataTable getPacientesCurados(string pacientes, int tiempoEspera)...

    private string construirConsultaEpisodiosCerrados(string pkeys, int tiempoEspera)...
}
```

Clase Principal:

En esta clase encontramos dos métodos:

- El primero (*getCentroMédico*) devuelve un objeto cuya parte estática es *ICentroMedico* y cuya parte dinámica es una instancia de la clase asociada al nombre que le entra como parámetro al método. Dicho nombre debe de ser el nombre del Centro Médico en el cual se ejecuta la aplicación.

  La parte estática es *ICentroMedico* debido a que el Módulo de Análisis trabaja con objetos del tipo *ICentroMedico*, no con objetos del tipo *Peset*, por ejemplo. De esta manera, como ya hemos comentado, incrementamos la calidad del software.

- El segundo (*setCadenaConexion*) se utiliza para cambiar la cadena de conexión al origen de datos según el centro médico donde se ejecute la aplicación o en el caso que el origen de datos sea trasladado a otra ruta. Interactúa con el fichero *app.config* y *Settings.cs*.

```
public class Principal
        {
            private ICentroMedico centroMedico;

            public ICentroMedico getCentroMedico(string nombreCentro)
            {
                switch (nombreCentro)
                {
                    case "Doctor Peset Aleixandre":
                        centroMedico = new Peset();
                        return centroMedico;

                    default: return null;
                }
            }

            public static void setCadenaConexion(string nombreCentro, string
            cadenaConexion)
            {
                switch (nombreCentro)
                {
                    case "Doctor Peset Aleixandre":
                        Peset.setCadenaConexion(cadenaConexion);break;

                }


    }
```

Biblioteca AccesoSepSOSbd:

Clase Queries.cs:

Contiene instancias estáticas de los objetos TableAdapter contenidos en el DataSet. De esta forma encapsulamos todo el trabajo con nuestra base de datos de SepSOS en una sola clase (al tenerlo todo junto, evitamos tener DataTables y TableAdapters en cada clase que necesite trabajar con la BD SQL y es escalable a otros productos software que quieran trabajar con nuestra Base de Datos a modo de Middleware).

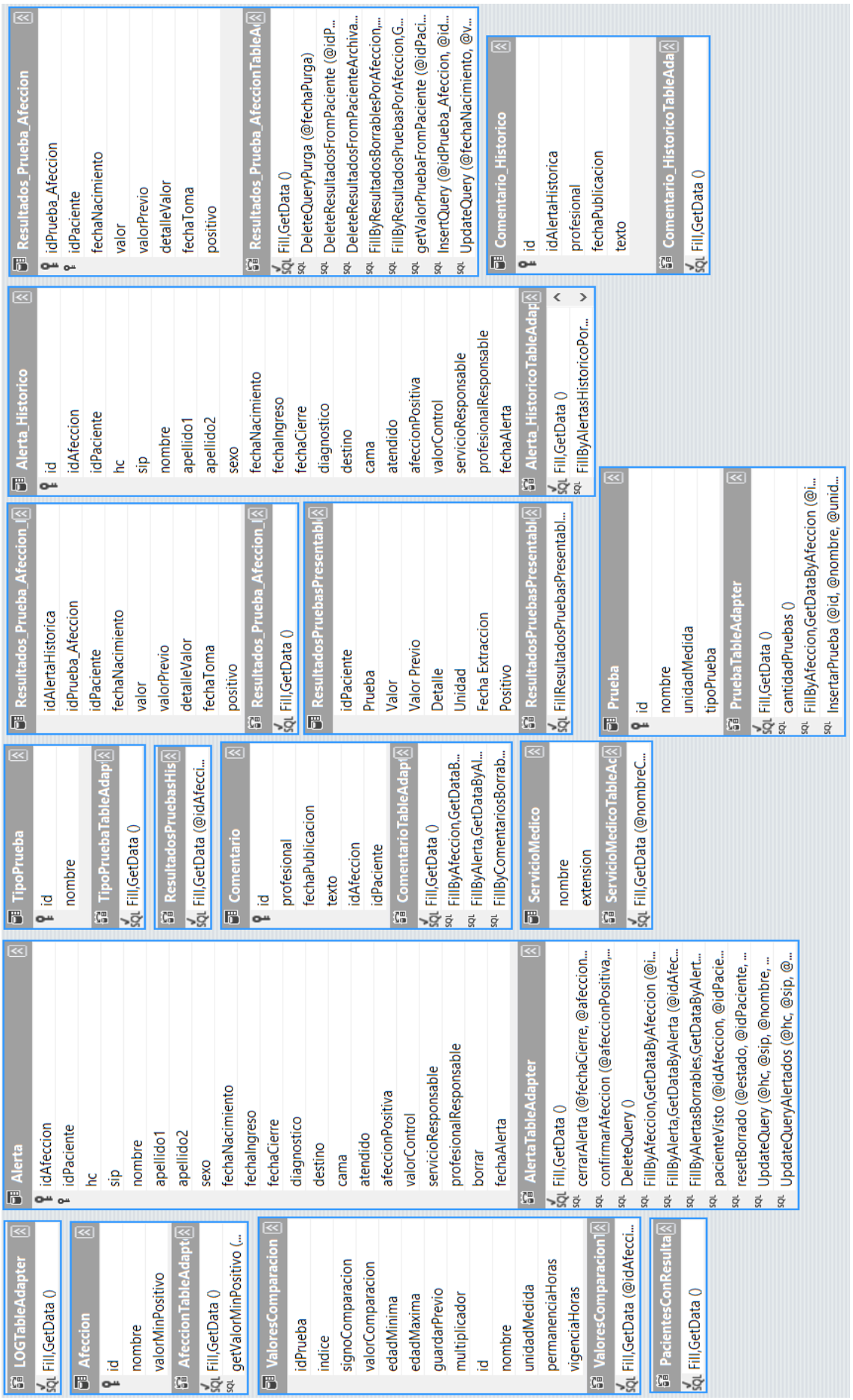
Tiene la capacidad de modificar la cadena de conexión a nuestra base de datos.

DataSet dsSepSOS.xsd:

Ya que nuestro servidor se trata de un SQL Server 2008, cuya versión es más actual que la versión de IBM Informix instalada en el Peset para trabajar con Orion, podemos utilizar los ficheros con extensión ".xsd" que contienen DataSets visibles gráficamente.

En estos esquemas encontramos esquemas de tablas de nuestra Base de Datos. Cada esquema (DataTable) tiene asociado un TableAdapter, el cual contiene todas las consultas necesarias para la gestión de datos de dicha tabla (Inserción, Borrado, Actualizado...)

Posee la capacidad de incluir restricciones de clave primaria y ajena, pero debido a que ya está implementado en el esquema del servidor, no hemos visto necesario su uso por redundancia.


Resultados_Prueba_Afeccion
idPrueba_Afeccion
idPaciente
fechaNacimiento
valor
valorPrevio
detalleValor
fechaToma
positivo
Resultados_Prueba_AfeccionTableA
Fill,GetData ()
DeleteQueryPurga (@fechaPurga)
DeleteResultadosFromPaciente (@idP...
DeleteResultadosFromPacienteArchiva...
FillByResultadosBorrablesPorAfeccion,...
FillByResultadosPruebasPorAfeccion,G...
getValorPruebaFromPaciente (@idPaci...
InsertQuery (@idPrueba_Afeccion, @id...
UpdateQuery (@fechaNacimiento, @v...
Alerta_Historico
id
idAfeccion
idPaciente
hc
sip
nombre
apellido1
apellido2
sexo
fechaNacimiento
fechaIngreso
fechaCierre
diagnostico
destino
cama
atendido
afeccionPositiva
valorControl
servicioResponsable
profesionalResponsable
fechaAlerta
Alerta_HistoricoTableAdap
Fill,GetData ()
FillByAlertasHistoricoPor...
Comentario_Historico
id
idAlertaHistorica
profesional
fechaPublicacion
texto
Comentario_HistoricoTableAda
Fill,GetData ()
Resultados_Prueba_Afeccion_I
idAlertaHistorica
idPrueba_Afeccion
idPaciente
fechaNacimiento
valor
valorPrevio
detalleValor
fechaToma
positivo
Resultados_Prueba_Afeccion_
Fill,GetData ()
ResultadosPruebasPresentabl
idPaciente
Prueba
Valor
Valor Previo
Detalle
Unidad
Fecha Extraccion
Positivo
ResultadosPruebasPresentabl
FillResultadosPruebasPresentabl...
Prueba
id
nombre
unidadMedida
tipoPrueba
PruebaTableAdapter
Fill,GetData ()
cantidadPruebas ()
FillByAfeccion,GetDataByAfeccion (@i...
InsertarPrueba (@id, @nombre, @unid...
TipoPrueba
id
nombre
TipoPruebaTableAdap
Fill,GetData ()
ResultadosPruebasHis
Fill,GetData (@idAfecci...
Comentario
id
profesional
fechaPublicacion
texto
idAfeccion
idPaciente
ComentarioTableAdapt
Fill,GetData ()
FillByAfeccion,GetDataB...
FillByAlerta,GetDataByAl...
FillByComentariosBorrab...
ServicioMedico
nombre
extension
ServicioMedicoTableAc
Fill,GetData (@nombreC...
Alerta
idAfeccion
idPaciente
hc
sip
nombre
apellido1
apellido2
sexo
fechaNacimiento
fechaIngreso
fechaCierre
diagnostico
destino
cama
atendido
afeccionPositiva
valorControl
servicioResponsable
profesionalResponsable
borrar
fechaAlerta
AlertaTableAdapter
Fill,GetData ()
cerrarAlerta (@fechaCierre, @afeccion...
confirmarAfeccion (@afeccionPositiva,...
DeleteQuery ()
FillByAfeccion,GetDataByAfeccion (@i...
FillByAlerta,GetDataByAlerta (@idAfec...
FillByAlertasBorrables,GetDataByAlert...
pacienteVisto (@idAfeccion, @idPacie...
resetBorrado (@estado, @idPaciente, ...
UpdateQuery (@hc, @sip, @nombre, ...
UpdateQueryAlertados (@hc, @sip, @...
LOGTableAdapter
Fill,GetData ()
Afeccion
id
nombre
valorMinPositivo
AfeccionTableAdapt
Fill,GetData ()
getValorMinPositivo (...
ValoresComparacion
idPrueba
indice
signoComparacion
valorComparacion
edadMinima
edadMaxima
guardarPrevio
multiplicador
id
nombre
unidadMedida
permanenciaHoras
vigenciaHoras
ValoresComparacionT
Fill,GetData (@idAfecci...
PacientesConResulta
Fill,GetData ()

Biblioteca ControlCentral:

Clase Deteccion:

Nos encontramos ante la piedra angular de la aplicación. *Deteccion* es la clase más importante del proyecto SepSOS. En ella encontramos los métodos de Análisis de resultados de pruebas, y la generación de alertas en base a esos resultados.

- **Método Deteccion:**
  Es el encargado de dar valor a los atributos de la clase acordes al Centro Médico donde se esté ejecutando la aplicación y el instante en el que se lanza el programa.

```
public Deteccion(string cm, DateTime ultimaFechaComprobada, string
correoEmisorIncidencias, string correoDestinatarioIncidencias, bool
soloTelefonosAfeccion, bool enviarAlertasMoviles, bool alertasCorreo)
        {
            puente = new Principal();
            sepSOS = new Queries();
            centroMedico = puente.getCentroMedico(cm);
            this.ultimaFechaComprobada = ultimaFechaComprobada;
            if (enviarAlertasMoviles)
            {
                gestorAlertas = new GestorAlertas(cm,
              correoEmisorIncidencias, correoDestinatarioIncidencias,
              soloTelefonosAfeccion, alertasCorreo);
                alerta = gestorAlertas.getGestorAlertas();
            }
            this.enviarAlertasMoviles = enviarAlertasMoviles;
            trabajando = false;
        }
```

- **Método Trabajando:**
  Comprueba que los hilos lanzados están en funcionamiento o no.

- **Método getTodasPruebas:**
  Utiliza una instancia de la clase Queries (Biblioteca AccesoSepSOSbd) para obtener el código de todas las pruebas que queremos analizar.

- **Método tarea:**
  Método que se ejecuta cada cierto tiempo, según lo indique el servicio. Si es la primera vez que se invoca, llama al método *precarga* y después a *tareaPool*. En caso contrario, solo a *tareaPool*.

```
public void tarea()
 {
     trabajando = true;
         if (esInicioAlgoritmo)
         {
             precargaTerminada = false;
             System.Diagnostics.Debug.WriteLine("------------fecha de
              comprobacion " + ultimaFechaComprobada);
             for (int i = 0; i < dtAfeccion.Rows.Count; i++)
                 ThreadPool.QueueUserWorkItem(precarga, dtAfeccion.Rows[i]
                    as object);
             esInicioAlgoritmo = false;
         }
         pruebasPedidas = false;
         getResultadosPruebas();
         while (!pruebasPedidas || !precargaTerminada) {
              Thread.Sleep(1000); };
         for (int i = 0; i < dtAfeccion.Rows.Count; i++)
             ThreadPool.QueueUserWorkItem(tareaPool, dtAfeccion.Rows[i] as
             object);

      }
```

- **Método tareaPool:**
  Método asociado a cada hilo de trabajo que invoca a los métodos *análisis* y generarAlerta. También informa si el hilo está trabajando.

```
private void tareaPool(object afeccion)
{
    System.Diagnostics.Debug.WriteLine("------------Inicio de la tarea");
    dsSepSOS.AfeccionRow enfermedad = afeccion as dsSepSOS.AfeccionRow;
    hilosTrabajando[enfermedad.id] = true;
    analisis(enfermedad.id);
    generarAlerta(enfermedad.id);
    hilosTrabajando[enfermedad.id] = false;
}
```

- **Método Precarga:**
  Transfiere todos los resultados de la base de datos a memoria principal para agilizar el análisis. Para ello, crea instancias de la clase *Paciente*, la cual contiene un resumen de los resultados de cada uno, tras esto lo introduce en su respectiva tabla hash. Solo es invocado cuando arranca la aplicación.

```
private void precarga(object enfermedad)
        {
            System.Diagnostics.Debug.WriteLine("------------Inicio de la
             precarga");
            dsSepSOS.AfeccionRow afeccion = enfermedad as
             dsSepSOS.AfeccionRow;
            sepSOS.insertarLOG("Inicio Precarga de "+ afeccion.nombre, "",
             "DANA", DateTime.Now);

            datosParaAnalisis[afeccion.id] = new
             DatosAnalisisPruebas(afeccion.id);
            rellenarTablaHashPacientes(afeccion.id);
            gestionAlertadosPrecarga(afeccion.id);
            precargaTerminada = true;
             System.Diagnostics.Debug.WriteLine("------------Fin de la
             precarga");
            sepSOS.insertarLOG("Finaliza Precarga de "+afeccion.nombre,
             "", "DANA", DateTime.Now);
        }
```

- **Método getResultadosPruebas:**
  Rellena la colección ResultadosPruebas con objetos DataTable compuestos por resultados de pruebas provenientes de Orion. Para ello se vale del objeto ICentroMedico “centroMedico”.

```
  private void getResultadosPruebas()
        {
            System.Diagnostics.Debug.WriteLine("-------------ENTRO PEDIR
             PRUEBAS");
            resultadosPruebas = new List<DataTable>();
            dsSepSOS.TipoPruebaDataTable tiposPruebas =
             sepSOS.getTiposPruebas();
            DateTime ahora = DateTime.Now;
            int cuantasPruebas = 0;
            foreach(dsSepSOS.TipoPruebaRow f in tiposPruebas.Rows)
            {
                DataTable aux = centroMedico.getResultadosPruebas(f.id,
                    pruebasPorTipo(f.nombre),
                    ultimaFechaComprobada.ToString(), ahora.ToString());
                cuantasPruebas += aux.Rows.Count;
                resultadosPruebas.Add(aux);
            }
            centroMedico.clearDataSetPruebas();
            ultimaFechaComprobada = ahora;
            pruebasPedidas = true;
            sepSOS.insertarLOG("Resultados de Pruebas Obtenidos",
             cuantasPruebas.ToString(), "DANA", DateTime.Now);
            System.Diagnostics.Debug.WriteLine("-------------SALGO DE
             PEDIR PRUEBAS");
        }
```

- **Método análisis:**
  Utiliza todos los datos que hemos obtenido de las consultas y peticiones anteriores para diagnosticar un posible caso de Sepsis y su correspondiente alerta mediante el cotejo con nuestros criterios de comparación.

```
private void analisis(int afeccion)
 {
     foreach (DataTable rp in resultadosPruebas)//Una tabla por cada tipo
      de prueba
     {
         try
         {
             DataTable p = rp.Select("prueba IN (" +
             datosParaAnalisis[afeccion].Pruebas + ")").CopyToDataTable();
             DataTable pacikeys = p.DefaultView.ToTable(true,
             "idPaciente");
             foreach (DataRow fila in pacikeys.Rows)
             {
                 DataRow[] resultadosPruebasPorPaciente = new DataRow[0];
                 string pkey = fila["idPaciente"].ToString();
                 try
                 {
                     resultadosPruebasPorPaciente = p.Select("[idPaciente]
                           = '" + pkey + "'");
                 }
                 catch (Exception e)
                 {
                     System.Diagnostics.Debug.WriteLine(e.Message);
                 }
                 for (int i = 0; i < resultadosPruebasPorPaciente.Length;
                    i++)
                 {
                     DateTime fNacimiento = DateTime.Now;
                     try
                     {
                         fNacimiento =
                           Convert.ToDateTime(resultadosPruebasPorPaciente
                    [i]["fechaNacimiento"]);
                     }
                     catch (Exception)
                     {
                         System.Diagnostics.Debug.WriteLine("------------
                           HA SALTADO EXCEPCION EN CONVERSION DE FECHA
                        NACIMIENTO= " +
                        resultadosPruebasPorPaciente[i]["fechaNacimiento"]);
                         break;
                     }
                     string pruebActual =
      resultadosPruebasPorPaciente[i]["prueba"].ToString();
                     List<DataRow> filasComparacion =
      getFilasComparacion(fNacimiento, afeccion, pruebActual, "", 0);
                     int[] resultados = new int[filasComparacion.Count];
                     DataRow fc = fila;
                     double miValor = 0;
                     int index = 0;
                     try
                     {
                         miValor =
      Convert.ToDouble(resultadosPruebasPorPaciente[i]["valor"]);
                     }
```

```
                    catch (Exception)
                    {

                        System.Diagnostics.Debug.WriteLine("------------
        HA SALTADO EXCEPCION EN CONVERSION DE mi valor = " +
                resultadosPruebasPorPaciente[i]["valor"]);
                    }
                    foreach (DataRow dr in filasComparacion)
                    {
                        fc = dr;

                        double valorComparacion = 0;
                        int resultado = 0;

                        try
                        {
                            valorComparacion =
        Convert.ToDouble(dr["valorComparacion"]);
                            resultado =
        compararValores(Convert.ToString(dr["signoComparacion"]), miValor,
        valorComparacion);
                            resultados[index] = resultado;
                            }
                        catch (Exception)
                        {
                        }
                        index++;
                    }
                    bool positivo = false;
                    for (int j = 0; j < resultados.Length; j++)
                    {
                        if (resultados[j] > 0)
                        {
                            positivo = true;
                            break;
                        }
                    }
                    if(positivo)
                        gestionResultadoAnalisis(afeccion, 1, miValor,
        resultadosPruebasPorPaciente[i]["detallevalor"].ToString(),
        Convert.ToDateTime(resultadosPruebasPorPaciente[i]["extraccion"]),
        fNacimiento, pkey, fc, 1);
                    else
                        gestionResultadoAnalisis(afeccion, -1, miValor,
        resultadosPruebasPorPaciente[i]["detallevalor"].ToString(),
        Convert.ToDateTime(resultadosPruebasPorPaciente[i]["extraccion"]),
        fNacimiento, pkey, fc, 1);

                }
            }
          }
          catch (Exception e)
          {
              System.Diagnostics.Debug.WriteLine("-------------HA SALTADO
        EXCEPCION EN ANALISIS = " + e.Message);
              sepSOS.insertarLOG("Fallo en Deteccion", e.Message, "DANA",
        DateTime.Now);
          }
      }
  }
```

- **Método gestionResultadoAnalisis:**
  Por cada resultado previamente analizado, lo guarda en su respectivo objeto paciente, en su respectiva tabla hash.
  Si es necesario, realiza una consulta a nuestra base de datos preguntando por el valor previo de dicho resultado.

```
    private void gestionResultadoAnalisis(int afeccion, int
        resultado,doublé miValor, string detalleValor, DateTime fechaToma,
        DateTime fNacimiento, string pkey, DataRow comparacion, int modo)
         {
      Hashtable pacientes = tablasPacientes[afeccion];
      Paciente p = null;
      if (pacientes.ContainsKey(pkey))
      {
          p = (Paciente)pacientes[pkey];
      }
      else
      {
          p = new Paciente(datosParaAnalisis[afeccion].TotalPruebas);
      }

      p.addResultadoPrueba(Convert.ToInt32(comparacion["indice"]),
        Convert.ToSByte(resultado),
        Convert.ToInt32(comparacion["multiplicador"]));
      tablasPacientes[afeccion][pkey] = p;
      if (modo > 0)//Modo == 0 -->Precarga. Modo > 0 -->Analisis
      {
          string previo = "Sin Previo";

          if (Convert.ToBoolean(comparacion["guardarPrevio"]))
          {
              try
              {
                  previo =
        sepSOS.getPruebaFromPaciente(Convert.ToInt32(pkey), afeccion,
        comparacion["idPrueba"].ToString()).ToString();
              }
              catch (Exception)
              {
              }
          }

          datosParaAnalisis[afeccion].addResultadoPrueba(Convert.ToInt32
          (comparacion["id"]), Convert.ToInt32(pkey), fNacimiento, miValor,
          previo, detalleValor, fechaToma, fromIntToBool(resultado));
      }
}
```

- **Método gestionAlertadosPrecarga:**
  Al reiniciar la aplicación, evita que las alertas que ya han sido enviadas vuelvan a lanzarse en la precarga.

- **Método generarAlerta:**
  Comprueba el estado de cada objeto paciente de las tablas hash. Si su valor de control supera al marcado por la afección como positivo, incluye su identificar de paciente en un String para pedir posteriormente sus datos a *Orion*. Si ya estuviera alertado, también se solicitan y se incluyen en un String diferente. Tras esto, se invoca a *guardarEnSepSOS*.

```
     private void generarAlerta(int afeccion)
{
    System.Diagnostics.Debug.WriteLine("-------------ENTRO EN GENERAR
     ALERTA");
    string pacientes = "";
    string pacientesModificables = "";
    Hashtable aux = tablasPacientes[afeccion].Clone() as Hashtable;
    foreach (DictionaryEntry de in aux)
    {
        Paciente p = de.Value as Paciente;
        if ((!p.Alertado && p.Modificado && p.Suma >=
     datosParaAnalisis[afeccion].ValorMinPositivo) || (p.Alertado &&
     p.Modificado && p.Suma > p.SumaPrevia))
        {
            pacientes +="'" + de.Key.ToString() + "', ";
            p.Alertado = true;
            tablasPacientes[afeccion][de.Key] = p;

            System.Diagnostics.Debug.WriteLine(de.Key + " Alertado");
        }

        if ((p.Alertado && p.Modificado && p.Suma <= p.SumaPrevia) ||
            (p.Alertado && !p.Modificado))
        {
            pacientesModificables += "'" + de.Key.ToString() + "', ";
            p.Alertado = true;
            tablasPacientes[afeccion][de.Key] = p;

            System.Diagnostics.Debug.WriteLine(de.Key+" Modificado");
        }
    }
    aux = null;
    guardarEnSepSOS(afeccion, pacientes, pacientesModificables);
    centroMedico.clearDataTablePacientes(afeccion);
    sepSOS.insertarLOG("Termina Deteccion", "", "DANA", DateTime.Now);
    System.Diagnostics.Debug.WriteLine("-------------TERMINA GENERAR
     ALERTAS");
}
```

- **Método guardarEnSepSOS:**
  Pregunta a Orion por los datos de los pacientes, independientemente de que tengan que ser alertados o actualizados. Después, guarda esos datos en nuestra base de datos junto con los resultados de las pruebas obtenidos anteriormente. Para ello, llama a *gestiónPacientesSepSOS* y *escribirResultadosPruebasSepSOS*.

```
        private void guardarEnSepSOS(int afeccion,  string pacientes,
             String pacientesModificables)
 {
     if (datosParaAnalisis[afeccion].ResultadosPruebas.Count > 0)
     {
         escribirResultadosPruebasSepSOS(afeccion);
     }
     else
     {
         System.Diagnostics.Debug.WriteLine("-------------SIN PRUEBAS QUE
             INSERTAR");
     }
       if (pacientes.Length > 0)//Hay pacientes que alertar
     {
         pacientes = pacientes.Substring(0, pacientes.Length - 2);
         DataTable pacientesAlertados = getDatosPaciente(pacientes,
             afeccion);
         System.Diagnostics.Debug.WriteLine("-------------HAY PACIENTES
             QUE ALERTAR");
         sepSOS.insertarLOG("Pacientes Alertados",
             pacientesAlertados.Rows.Count.ToString(), "DANA",
      DateTime.Now);
         gestionPacientesSepSOS(afeccion, pacientesAlertados, 0);
         if (enviarAlertasMoviles)
         {
             alerta.alertar(pacientesAlertados, afeccion);
         }
     }
     else
     {
         System.Diagnostics.Debug.WriteLine("-------------SIN PACIENTES
             QUE ALERTAR");
     }
       if (pacientesModificables.Length > 0)//Hay pacientes que modificar
     {
         pacientesModificables = pacientesModificables.Substring(0,
             pacientesModificables.Length - 2);
         DataTable pacientesModificados =
             getDatosPaciente(pacientesModificables, afeccion);
         System.Diagnostics.Debug.WriteLine("-------------HAY PACIENTES
             QUE MODIFICAR");
         sepSOS.insertarLOG("Pacientes Modificados Actualizados",
             pacientesModificados.Rows.Count.ToString(), "DANA",
      DateTime.Now);
         gestionPacientesSepSOS(afeccion, pacientesModificados, 1);
     }
     else
     {
         System.Diagnostics.Debug.WriteLine("-------------SIN PACIENTES
             QUE MODIFICAR");
     }
            GC.Collect();
 }
```

- **Método comporarValores:**
  Compara los valores de comparación de nuestra afección (Sepsis) con el valor de control de cada paciente para comprobar si da positivo o negativo.

```
private int compararValores(string signoComparacion, double
resultadoPrueba, double valorComparacion)
 {
     int resultadoComparacion = -1;

     switch (signoComparacion.ToLower())
     {
         case "menorestrictoque": if (resultadoPrueba < valorComparacion)
             resultadoComparacion = 1; break;
         case "mayorestrictoque": if (resultadoPrueba > valorComparacion)
             resultadoComparacion = 1; break;
         case "menorigualque": if (resultadoPrueba <= valorComparacion)
             resultadoComparacion = 1; break;
         case "mayorigualque": if (resultadoPrueba >= valorComparacion)
             resultadoComparacion = 1; break;
         case "igualigual": if (resultadoPrueba == valorComparacion)
             resultadoComparacion = 1; break;
         case "igual": if (resultadoPrueba == valorComparacion)
             resultadoComparacion = 1; break;
         case "diferente": if (resultadoPrueba != valorComparacion)
             resultadoComparacion = 1; break;

     }
     return resultadoComparacion;
 }
```

- **Método getDatosPacientes:**
  Solicita los datos del paciente a *Orion*, entre los cuales se encuentras la ubicación del paciente, sus datos personales, médico responsable, etc...

```
private DataTable getDatosPaciente(string pacientes, int afeccion)
 {
     System.Diagnostics.Debug.WriteLine("-------------ENTRO PEDIR
       PACIENTES AFECTADOS");
     return centroMedico.getDatosPacientes(pacientes, afeccion);
 }
```

- **Método getFilasComparacion:**
  Obtiene la fila de comparación de la propia prueba que estemos analizando y la edad del paciente.

```
 private List<DataRow> getFilasComparacion(DateTime fNacimiento, int
      afeccion, string codPrueba, string codPruebaAfeccion, int modo)
{
    List<DataRow> filas = new List<DataRow>();
    DataRow[] valoresPorPrueba = new DataRow[0];
    switch(modo)
    {
        case 0: valoresPorPrueba =
            datosParaAnalisis[afeccion].ValoresComparacionPruebas.Select(
            "idPrueba = '" + codPrueba + "'"); break;
        case 1: valoresPorPrueba =
            datosParaAnalisis[afeccion].ValoresComparacionPruebas.Select(
            "id = '" + codPruebaAfeccion + "'"); break;
    }
    for (int i = 0; i < valoresPorPrueba.Length; i++)
    {
        if (
            (DateTime.Today.AddDays(
            Convert.ToInt32(valoresPorPrueba[i]["edadMaxima"])).CompareTo
            (fNacimiento) <= 0)//Mi edad es mayor o igual a la minima
            &&
            (DateTime.Today.AddDays(
            Convert.ToInt32(valoresPorPrueba[i]["edadMinima"])).CompareTo
            (fNacimiento) >= 0)//Mi edad es menor o igual a la maxima
            )
        {
            filas.Add(valoresPorPrueba[i]);
        }
    }
    return filas;
}
```

Clase Borrado:

- **Método Borrado:**
  Constructor que da valores a los atributos de la clase.

```
{
      fin = false;
      sepSOS = new Queries();
      puente = new Principal();
      centroMedico = puente.getCentroMedico(cm);
      tiempoEsperaBorrado = teb;
      tiempoEsperaPurgado = tep;
}
```

- **Método IniciarBorrado:**
  Lanza método *tarea*.

- **Método Tarea:**
  Contiene todas las acciones a realizar al lanzar el borrado, incluyendo los métodos *archivarAlertas* y *borrarResultadosPruebas*.

```
private void tarea()
 {
      try
      {
          archivarAlertas();
          borrarResultadosPruebas();
          purgarSistema(DateTime.Today.AddDays(-tiempoEsperaPurgado));
          fin = true;
          sepSOS.insertarLOG("Finaliza Borrado", "", "DANA", DateTime.Now);
          System.Diagnostics.Debug.WriteLine("------------TERMINA
              BORRADO");
      }
      catch (Exception e)
      {
          fin = true;
          sepSOS.insertarLOG("Fallo de Borrado", e.Message, "DANA",
              DateTime.Now);
          System.Diagnostics.Debug.WriteLine("------------FALLO DE
              BORRADO");
      }
}
```

- **Método archivarAlertas:**
  Archiva las alertas que hayan sido marcadas para ser archivadas y que en Orion aparecen como finalizadas al menos el tiempo indicado en *tiempoEsperaBorrado*. Los comentarios también son almacenados en el histórico junto con su respondiente alerta.

```
{
      actualizarAlertasNoBorrables();

      dsSepSOS.AlertaDataTable alertasABorrar =
        sepSOS.getAlertasBorrables();
      dsSepSOS.Resultados_Prueba_AfeccionDataTable todosResultadosABorrar;
      dsSepSOS.ComentarioDataTable todosComentariosABorrar =
        sepSOS.getComentariosBorrables();
      DataTable afecciones = new DataTable();
      try
      {
          afecciones = alertasABorrar.DefaultView.ToTable(true,
        "idAfeccion");
      }
      catch (Exception) { }

      foreach (DataRow afeccion in afecciones.Rows)
      {
          DataTable alertasABorrarPorAfeccion = new DataTable();
          try
          {
              alertasABorrarPorAfeccion = alertasABorrar.Select("idAfeccion
       = " + afeccion["idAfeccion"].ToString()).CopyToDataTable();
          }
          catch (Exception) { }

          todosResultadosABorrar =
              sepSOS.getResultadosPruebasBorrablesPorAfeccion(Convert.
              ToInt32(afeccion["idAfeccion"]));

          foreach (DataRow fila in alertasABorrarPorAfeccion.Rows)
          {

              sepSOS.insertarAlertaHistorica(Convert.ToInt32(fila["idAfecci
              on"]), Convert.ToInt32(fila["idPaciente"]),
              Convert.ToInt32(fila["hc"]), fila["sip"].ToString(),
              fila["nombre"].ToString(),fila["apellido1"].ToString(),
              fila["apellido2"].ToString(),fila["sexo"].ToString(),
              Convert.ToDateTime(fila["fechaNacimiento"]),
              Convert.ToDateTime(fila["fechaIngreso"]), DateTime.Now,
              fila["diagnostico"].ToString(),fila["destino"].ToString(),
              fila["cama"].ToString(),
              Convert.ToBoolean(fila["atendido"]),
              Convert.ToBoolean(fila["afeccionPositiva"]),
              Convert.ToInt32(fila["valorControl"]),
              fila["servicioResponsable"].ToString(),
              fila["profesionalResponsable"].ToString(),
              Convert.ToDateTime(fila["fechaAlerta"]));

            try
              {
               int idHistorico =
              sepSOS.getUltimoIdHistorico(Convert.ToInt32
              (fila["idPaciente"]),
              Convert.ToInt32(fila["idAfeccion"]));
```

```
                try
                   {
                   DataTable comentariosABorrar=
                   todosComentariosABorrar.Select("idPaciente = " +
                   Convert.ToInt32(fila["idPaciente"]) + " and idAfeccion = " +
                   Convert.ToInt32(fila["idAfeccion"])).CopyToDataTable();
                           foreach (DataRow row in comentariosABorrar.Rows)
                           {
                               sepSOS.insertarComentariosHistoricos(idHistorico,
                                 row["profesional"].ToString(),
                                 Convert.ToDateTime(row["fechaPublicacion"]),
                                 row["texto"].ToString());
                           }
                       }
                       catch (Exception)
                       {
                           System.Diagnostics.Debug.WriteLine("-----------
                                 Alerta sin Comentarios");
                       }

                       try
                       {
                          DataTable resultadosABorrar =
                          todosResultadosABorrar.Select("idPaciente = " +
                         Convert.ToInt32(fila["idPaciente"])).CopyToDataTable();
                           foreach (DataRow row in resultadosABorrar.Rows)
                           {

                   sepSOS.insertarResultadoPruebaAfeccionHistorico(idHistorico,
                   Convert.ToInt32(row["idPrueba_Afeccion"]),
                   Convert.ToInt32(row["idPaciente"]),
                   Convert.ToDateTime(row["fechaNacimiento"]),
                   Convert.ToDouble(row["valor"]),
                   Convert.ToString(row["valorPrevio"]),
                   Convert.ToString(row["detalleValor"]),
                   Convert.ToDateTime(row["fechaToma"]),
                   Convert.ToBoolean(row["positivo"]));
                           }
                       }
                       catch (Exception)
                       {
                           System.Diagnostics.Debug.WriteLine("-----------
                                 Alerta sin pruebas");
                       }
                   }
                   catch (Exception) { }
               }
           }

           sepSOS.borrarAlertas();

           remarcarAlertasComoBorrables();
       }
```

- **Método borrarResultadosPruebas:**
  De los resultados de pruebas que tenemos guardados cuyo paciente no ha generado alerta, si en Orion aparece como finalizado, pasan a ser borrados.

```
private void borrarResultadosPruebas()
 {
     try
     {
         dsSepSOS.AlertaDataTable todasAlertas = sepSOS.getAlertas();
         DataTable alertasBorrarFalse = new DataTable();
         try
         {
             alertasBorrarFalse = todasAlertas.Select("borrar =
              false").CopyToDataTable();
         }
         catch (Exception){ }
         foreach (DataRow fila in episodiosTerminados.Rows)
         {
             int pkey = Convert.ToInt32(fila["idPaciente"]);

             bool esta = false;
             foreach (DataRow row in alertasBorrarFalse.Rows)
             {
                 if (Convert.ToInt32(row["idPaciente"]) == pkey)
                 {
                     esta = true;
                     break;
                 }
             }

             if(!esta)sepSOS.deletePruebasFromPaciente(pkey);
         }
         centroMedico.clearDataTableCurados();
     }
     catch (Exception e)
     {
         sepSOS.insertarLOG("Fallo Borrado", e.Message, "DANA",
              DateTime.Now);
     }
 }
```

- **Método purgarSistema:**
  Borra todos los resultados de pruebas guardados por encima de un tiempo indicado. Es una medida de seguridad/prevención para evitar que la base de datos se llene con resultados desfasados e inútiles.

- **Método getPacientesCurados:**
  De todos los pacientes que tengo, comprueba cuales superan el tiempo *tiempoEsperaBorrado* para realizar el borrado o no.

```
private void getPacientesCurados()
 {
     DataTable pacientesDiferentes = sepSOS.getPacientesConResultados();

     string pacientes = "";
     foreach (DataRow fila in pacientesDiferentes.Rows)
     {
         pacientes += "'" + fila["idPaciente"].ToString() + "', ";
     }
     if (pacientes.Length > 0)
         pacientes = pacientes.Substring(0, pacientes.Length - 2);

     episodiosTerminados = centroMedico.getPacientesCurados(pacientes,
      tiempoEsperaBorrado);
 }
```

- **Método getAlertasNOBorrables:**
  Obtiene las alertas que todavía no están curadas en Orion, pero que si están marcadas en sepSOS para archivar, y las marca a "false". Una vez finalizado el borrado, vuelven a ponerse a "true" (para esto, utiliza los métodos *actualizarAlertasNoBorrables* y *remarcarAlertasComoBorrables* respectivamente). De esta manera, evitamos borrar una alerta que si está para archivarse pero que Orion nos indica que todavía no ha sido dado de alta el paciente.

```
private List<DataRow> getAlertasNOBorrables()
 {

     List<DataRow> alertasNoBorrables = new List<DataRow>();

     try
     {
         episodiosTerminados.PrimaryKey = new DataColumn[] {
             episodiosTerminados.Columns["idPaciente"] };

         dsSepSOS.AlertaDataTable alertasABorrar =
             sepSOS.getAlertasBorrables();

         foreach (DataRow fila in alertasABorrar)
         {
             if (!episodiosTerminados.Rows.Contains(fila["idPaciente"]))
                 alertasNoBorrables.Add(fila);
         }
     }
     catch (Exception) { }

     return alertasNoBorrables;
       }
```

Clase DatosAnalisisPruebas:

- **Método DatosAnalisisPruebas:**
  El método recibe el código de afección a analizar y obtiene todos los datos relacionados con dicha enfermedad:
  Que afección es, por que pruebas está formada, cuantas pruebas son, cuáles son sus valores de comparación, cual es el valor mínimo para dar un positivo y los resultados procesados para después guardarlo todo de golpe.

```
public DatosAnalisisPruebas(int enfermedad)
 {
     afeccion = enfermedad;
     totalPruebas = sepSOS.getCantidadPruebas();
     getCodigosPruebas();
     valoresComparacionPruebas =
       sepSOS.getValoresComparacionPruebas(enfermedad);
     valorMinPositivo = sepSOS.getValorMinPositivoAfeccion(enfermedad);
     resultadosPruebas = new
       dsSepSOS.Resultados_Prueba_AfeccionDataTable();

 }
```

- **Método codPruebasFromDataTableToString:**
  Por cada código de prueba a revisar, que viene por DataTable, se coge dicho código y se añade a un String de código de pruebas para poder utilizarlo en una consulta SQL mediante un IN.

```
private    void    codPruebasFromDataTableToString(dsSepSOS.PruebaDataTable
dtPruebas)

 {

     foreach (dsSepSOS.PruebaRow fila in dtPruebas.Rows)

     {

         pruebas += "'" + fila["id"].ToString() + "', ";

     }

     pruebas = pruebas.Substring(0, pruebas.Length - 2);

 }
```

Clase Paciente:

En paciente se guarda todo lo relativo a las pruebas que se le han realizado a un paciente, su valor de control, el previo para saber si empeora...

Además, con añadir un resultado nuevo, se actualiza en su posición de la colección de resultados.

- **Método addResultadoPrueba:**

  Método que incluye el resultado de una prueba dentro de la colección de resultados del objeto paciente. Una prueba solo puede ocupar una posición de la colección y su valor puede ser "0" si aún no tenemos el resultado, "+1" si es positivo o "-1" en caso contrario. Además al atributo *suma* de la clase se le añade o resta el valor del multiplicador en función del resultado de la prueba. El campo suma es el encargado de controlar la gravedad global de paciente y es el que se comparará con el valor mínimo de la afección que estemos analizando para discernir si debemos mandar una alerta.

```
public void addResultadoPrueba(int pos, sbyte valor, int multiplicador)
 {
     //System.Diagnostics.Debug.WriteLine("-------------TENIA " +
       arrayPruebas[pos] + ", MI VALOR ES " + valor+", SUMA VALIA "+suma);
     switch(arrayPruebas[pos])
     {
         case 1: if (valor < 0) suma += valor * multiplicador;  break;
         case 0:
         case -1: if (valor > 0) suma += valor * multiplicador; break;
     }
     //System.Diagnostics.Debug.WriteLine("-------------SUMA VALE " +
       suma);
     arrayPruebas[pos] = valor;
     modificado = true;
 }
```

- **Método Alertado:**

  Además de cambiar el valor del campo alertado, en su modificador pone "modificado" a false y "sumaPrevia" pasa a valer "suma".

```
  public bool Alertado
 {
     get { return alertado; }
     set
     {
         alertado = value;
         sumaPrevia = suma;
         modificado = false;
     }
 }
```

Clase IAlerta:

Utilizamos esta interfaz a modo de plantilla, para asegurarnos que cualquier centro médico que utilice esta aplicación cumpla con los requisitos de estructura que necesita el módulo de análisis, pues puede que no todos los centros alerten de la misma manera.

```
interface IAlerta
{
        void alertar(DataTable pacientes, int afeccion);
}
```

Clase AlertaMovilPeset:

Se encarga de enviar los mensajes de texto y correos electrónicos a todos los que deban ser alertados. También se distingue si se quieren mandar esos correos o si solo quieren mandarlos al teléfono asignado.

- **Método obtenerServiciosMedicos:**
  Consulta a nuestra base de datos para saber todos los servicios médicos que tenemos con sus respectivos números de teléfono a partir del nombre del centro que le indiquemos.

- **Método obtenerAfecciones():**
  Obtenemos todas las afecciones que estamos estudiando para enviar las alertas de la afección que corresponda.

```
private void obtenerAfecciones()
{
    afecciones = accesoSepSOS.getAfecciones();
}
```

- **Método obtenerExtensionServicio:**
  A partir de la tabla de servicios, busca el número de teléfono de un servicio concreto.

```
private int obtenerExtensionServicio(string servicio)
        {
            DataRow[] extensiones = serviciosMedicos.Select("nombre = '" +
              servicio + "'");
            if (extensiones.Length > 0)
            {
                return Convert.ToInt32(extensiones[0]["extension"]);
            }
            else
            {
                return -1;
            }
        }
```

- **Método obtenerExtensionAfeccion:**
  Idéntico al método anterior. A partir de un identificador de afección, obtiene el número asociado a esa afección.

- **Método alertar:**
  Método principal que está obligado a dar código desde la interfaz. Comprueba a quien tiene que enviar la alerta, si envía mensaje de texto y correo, solo correo, solo SMS, etc... y tras fabricar el mensaje lo envía. También averigua el número del servicio al que debe enviar la alerta en base al identificador de afección.

```
public void alertar(DataTable pacientes, int afeccion)
 {
     int extensionAfeccion = obtenerExtensionAfeccion(afeccion);

     foreach (DataRow alerta in pacientes.Rows)
     {
         string mensaje = obtenerMensaje(alerta, afeccion);
         if (!soloTelefonosAfeccion)
         {
             int extensionServicio =
       obtenerExtensionServicio(alerta["servicioResponsable"].ToString());
             if (extensionServicio > 0 && mensaje.Length > 0)
                 enviarSMS(mensaje, extensionServicio);
         }
         enviarSMS(mensaje, extensionAfeccion);
         if (alertasACorreo)
         {
             CorreoElectronico.enviarCorreoElectronico(clienteSMTP,
             correoEmisorIncidencias, correoDestinatarioIncidencias,
             "Alerta por Sepsis", mensaje);
         }
     }
 }
```

- **Método obtenerMensaje:**
  Formatea el mensaje para una alerta y afección en concreto. Aparecen los campos que ya han sido vistos. Está acotado a 160 caracteres.

```
private string obtenerMensaje(DataRow alerta, int afeccion)
 {
     string msj = "";
     DataRow[] nombres = afecciones.Select("id = '"+afeccion+"'");
     if (nombres.Length > 0)
     {
         msj += "Alerta por " + nombres[0]["nombre"].ToString().ToUpper()
             + ". NHC: " + alerta["hc"].ToString().Trim() + ", " +
                "en el servicio: " + alerta["servicio"].ToString() + " y
             en la " +
                "cama: " + alerta["cama"].ToString();
     }
     return msj;
 }
```

- **Método gestionEnvioSMS:**
  Comprueba que la respuesta de la web coincide con la que genera *generarRespuestaEnvioSMSCorrecto.* En caso de no coincidir, es que algo ha fallado, y se envía un correo de incidencias.

```
private void gestionEnvioSMS(string resultado, int numero)
 {
     string respuestaEsperada = generarRespuestaEnvioSMSCorrecto(numero);

     if (!resultado.Equals(respuestaEsperada))
     {
         enviarEmailError(resultado);
     }
 }
```

- **Método enviarSMS:**
  Invocamos la url de un servicio de mensajería utilizado por el Hospital Peset Aleixandre para enviar mensajes de texto a teléfonos corporativos. Le pasamos el mensaje de texto mediante un String, que el servicio se encarga de enviar al número indicado.

```
private void enviarSMS(string msg, int nro)
 {
     HttpWebRequest request =
       (HttpWebRequest)WebRequest.Create("http://172.17.185.10/
       cgibin/EnvioSms?msg=" + msg + "&nro=" + nro);
     HttpWebResponse response = (HttpWebResponse)request.GetResponse();
     Stream receiveStream = response.GetResponseStream();
     StreamReader readStream = new StreamReader(receiveStream,
       Encoding.UTF8);
     string inputLine = readStream.ReadToEnd();
     gestionEnvioSMS(inputLine,nro);
     response.Close();
     readStream.Close();
     System.Diagnostics.Debug.WriteLine("respuesta de envio del sms: " +
       inputLine);
}
```

- **Método enviarEmailError:**
  Método responsable de enviar los correos electrónicos de incidencias. Se vale del método estático *enviarCorreoElectronico* de la clase ***CorreoElectronico***.

```
public void enviarEmailError(string msjError)
 {
     CorreoElectronico.enviarCorreoElectronico(clienteSMTP,
       correoEmisorIncidencias, correoDestinatarioIncidencias, "Fallo en
       SepSOS", "El envío de SMS de SRVALERTAS ha fallado. Error: " +
       Environment.NewLine + msjError);
  }
```

Clase CorreoElectonico:

Clase de apoyo que solo contiene un método estático para enviar los correos electrónicos (ya sean de incidencias o alertas). Se parametrizan en el constructor los atributos necesarios.

```
class CorreoElectronico
    {
        public static void enviarCorreoElectronico(string clienteSMTP, string
             emisor, string destinatario, string asunto, string mensaje)
        {
            MailMessage email = new MailMessage();

            email.To.Add(new MailAddress(destinatario));

            email.From = new MailAddress(emisor);

            email.Subject = asunto;

            email.Body = mensaje;

            SmtpClient clienteSmtp = new SmtpClient(clienteSMTP);

            try
            {
                clienteSmtp.Send(email);
                System.Diagnostics.Debug.WriteLine("Enviado");
            }

            catch (Exception ex)
            {
                System.Diagnostics.Debug.WriteLine(ex.Message);
            }
        }
    }
```

Clase GestorAlertas:

A partir del nombre del centro y de los atributos necesarios para enviar los mensajes de texto y correos electrónicos, devuelve una instancia de la clase que implementa la interfaz IAlerta para ese centro médico. (Tiene la misma función que ***principal*** en la biblioteca ***accesoDatosMedicos***)

```
class GestorAlertas
    {
        private IAlerta alerta;
        private string centroMedico;
        private string correoEmisorIncidencias;
        private string correoDestinatarioIncidencias;
        private bool soloTelefonosAfeccion;
        private bool alertasACorreo;

        public GestorAlertas(string centroMedico, string correoEmisorIncidencias,
              string correoDestinatarioIncidencias, bool soloTelefonosAfeccion,
              bool alertasACorreo)
        {
            this.centroMedico = centroMedico;
            this.correoDestinatarioIncidencias = correoDestinatarioIncidencias;
            this.correoEmisorIncidencias = correoEmisorIncidencias;
            this.soloTelefonosAfeccion = soloTelefonosAfeccion;
            this.alertasACorreo = alertasACorreo;
        }

        public IAlerta getGestorAlertas()
        {
            switch (centroMedico)
            {
                case "Doctor Peset Aleixandre":
                    alerta = new
                    AlertaMovilPeset(centroMedico,correoEmisorIncidencias,
                    correoDestinatarioIncidencias,soloTelefonosAfeccion,
                    alertasACorreo);
                    return alerta;

                default: return null;
            }
        }
    }
```

7. Fase de Pruebas

Una vez diseñado e implementado un programa, debe someterse a una serie de pruebas para comprobar su correcto funcionamiento.

eXtreme Programming exige la superación de una serie de pruebas unitarias para dar como válido un software o prototipo. Dichas pruebas vuelven a realizarse en formato de tarjeta, siguiendo una plantilla estándar, para poder trabajar con ellas, modificarlas...

Casos de Prueba de Aceptación:

	Caso de Prueba de Aceptación
Código Caso de Prueba: 1-1	**Número Historia de Usuario:** 1
Descripción de la Prueba: Los datos de pruebas obtenidos son los mismos que los que aparecen en Orión Clinic.	
Condiciones de Ejecución: Que en Orion haya pruebas y Orion funcione.	
Entrada / Pasos de ejecución: Ejecutamos la Query y nos traemos datos de pruebas de Orion.	
Resultado Esperado: Los datos coinciden	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1-2	**Número Historia de Usuario:** 1
Descripción de la Prueba: La obtención de datos de pruebas no precisa de un tiempo excesivo	
Condiciones de Ejecución: Que en Orion haya pruebas y Orion funcione	
Entrada / Pasos de ejecución: Ejecutamos la Query y nos traemos datos de pruebas de Orion. Comprobamos el tiempo de respuesta.	
Resultado Esperado: Recuperación prácticamente inmediata	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1,2-1	**Número Historia de Usuario:** 1, 2
Descripción de la Prueba: La aplicación conecta satisfactoriamente con Orión Clinic (IBM Informix).	
Condiciones de Ejecución: Que la cadena de conexión sea correcta y Orion funcione.	
Entrada / Pasos de ejecución:	
Resultado Esperado: Conexión satisfactoria.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2-1	**Número Historia de Usuario:** 2
Descripción de la Prueba: Las consultas con el tiempo parametrizado se ejecutan cumpliendo con el plazo previsto.	
Condiciones de Ejecución: Que en Orion haya datos de pacientes y Orion funcione	
Entrada / Pasos de ejecución: Comprobamos que al pasar el tiempo indicado, se lanzan las consultas.	
Resultado Esperado: Las consultas son lanzadas.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2-2	**Número Historia de Usuario:** 2
Descripción de la Prueba: Los datos de pacientes obtenidos coinciden con los de Orión Clinic.	
Condiciones de Ejecución: Que en Orion haya pruebas y Orion funcione	
Entrada / Pasos de ejecución: Ejecutamos la Query y nos traemos datos pacientes de Orion	
Resultado Esperado: Los datos coinciden	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2-3	**Número Historia de Usuario:** 2
Descripción de la Prueba: La obtención de datos de pruebas no precisa de un tiempo excesivo	
Condiciones de Ejecución: Que en Orion haya pacientes y Orion funcione	
Entrada / Pasos de ejecución: Ejecutamos la Query y nos traemos datos de pacientes de Orion. Comprobamos el tiempo de respuesta.	
Resultado Esperado: Recuperación prácticamente inmediata	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 3-1	**Número Historia de Usuario:** 3
Descripción de la Prueba: Los datos de pruebas insertados en nuestra base de datos coinciden con los previamente recuperados.	
Condiciones de Ejecución: Que hayamos traído previamente pruebas y que nuestro servidor este activo.	
Entrada / Pasos de ejecución: Insertamos en nuestra BD pruebas traídas de Orion.	
Resultado Esperado: Los datos coinciden.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 3-2	**Número Historia de Usuario:** 3
Descripción de la Prueba: La inserción de datos no conlleva un tiempo excesivo.	
Condiciones de Ejecución: Que hayamos traído previamente pruebas y que nuestro servidor este activo.	
Entrada / Pasos de ejecución: Insertamos en nuestra BD pruebas traídas de Orion. Comprobamos el tiempo de insertado.	
Resultado Esperado: Inserción prácticamente inmediata.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 4-1	**Número Historia de Usuario:** 4
Descripción de la Prueba: Los datos de pacientes insertados en nuestra base de datos coinciden con los previamente recuperados.	
Condiciones de Ejecución: Que hayamos traído previamente datos de pacientes y que nuestro servidor este activo.	
Entrada / Pasos de ejecución: Insertamos los datos de pacientes traídos de Orion en nuestra BD y los cotejamos.	
Resultado Esperado: Los datos coinciden.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 4-2	**Número Historia de Usuario:** 4
Descripción de la Prueba: La inserción de datos no conlleva un tiempo excesivo.	
Condiciones de Ejecución: Que hayamos traído previamente datos de pacientes y que nuestro servidor este activo.	
Entrada / Pasos de ejecución: Insertamos en nuestra BD datos de pacientes traídas de Orion. Comprobamos el tiempo de insertado.	
Resultado Esperado: Inserción prácticamente inmediata.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7-1	**Número Historia de Usuario:** 7
Descripción de la Prueba: El algoritmo no tarda un tiempo excesivo en procesar las pruebas nuevas	
Condiciones de Ejecución: Que tengamos pruebas nuevas. Que nuestro servicio esté en funcionamiento.	
Entrada / Pasos de ejecución: Analizamos las pruebas nuevas. Comprobamos el tiempo transcurrido en el Análisis.	
Resultado Esperado: Tiempo lineal con el número de pruebas a analizar.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7-2	**Número Historia de Usuario:** 7
Descripción de la Prueba: El algoritmo discierne correctamente si el valor obtenido de la prueba es superior o inferior a su respectivo valor de comparación, marcando dicha prueba como positiva o negativa respectivamente.	
Condiciones de Ejecución: Que tengamos pruebas y que nuestro servicio de detección esté en funcionamiento.	
Entrada / Pasos de ejecución: Cogemos una prueba y comprobamos si su valor supera o no el valor umbral declarado para considerarla positiva para esa afección.	
Resultado Esperado: Evalúa correctamente como positiva/negativa la prueba	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7-3	**Número Historia de Usuario:** 7
Descripción de la Prueba: El algoritmo lanza la consulta a Orión de los pacientes cuyas pruebas han dado positivas	
Condiciones de Ejecución: Que tengamos un conjunto de pruebas que hayan dado positivo para la afección. Que Orion funcione.	
Entrada / Pasos de ejecución: Comprobamos que los valores de las pruebas son positivos y que el valor de control es superior al umbral para considerar un posible caso positivo en la afección.	
Resultado Esperado: La consulta es lanzada	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7-4	**Número Historia de Usuario:** 7
Descripción de la Prueba: Antes de empezar el análisis, el algoritmo intenta traer de nuestra base de datos (si existen) el resto de pruebas de un paciente en concreto.	
Condiciones de Ejecución: Que tengamos un conjunto de pruebas que hayan dado positivo para la afección. Que Orion funcione. Que nos hayamos traído el paciente.	
Entrada / Pasos de ejecución: Se lanza una consulta para el paciente cuyas pruebas son positivas para esa afección.	
Resultado Esperado: La consulta es correcta si existen resultados.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7-5	**Número Historia de Usuario:** 7
Descripción de la Prueba: El algoritmo lanza la Query de inserción de pacientes en nuestra base de datos (si procede)	
Condiciones de Ejecución: Que tengamos un conjunto de pruebas que hayan dado positivo para la afección. Que Orion funcione. Que nos hayamos traído el paciente.	
Entrada / Pasos de ejecución: El paciente cuyas pruebas han sido positivas para la afección debe almacenarse en nuestra BD.	
Resultado Esperado: Query lanzada y paciente insertado.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7-6	**Número Historia de Usuario:** 7
Descripción de la Prueba: El algoritmo lanza la Query de inserción de pruebas en nuestra base de datos (si procede)	
Condiciones de Ejecución: Que tengamos un conjunto de pruebas que hayan dado positivo para la afección. Que Orion funcione. Que nos hayamos traído el paciente.	
Entrada / Pasos de ejecución: Las pruebas que han sido positivas para la afección debe almacenarse en nuestra BD.	
Resultado Esperado: Query lanzada y pruebas insertadas.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7-7	**Número Historia de Usuario:** 7
Descripción de la Prueba: El algoritmo lanza al algoritmo de envío de alertas si es necesario	
Condiciones de Ejecución: Que haya pruebas positivas suficientes como para lanzar una alerta.	
Entrada / Pasos de ejecución: Comprueba si el Valor de Control de las pruebas positivas supera el valor umbral.	
Resultado Esperado: Algoritmo lanzado.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 8-1	**Número Historia de Usuario:** 8
Descripción de la Prueba: El destinatario del Mensaje de Texto es el correcto.	
Condiciones de Ejecución: Que haya pruebas positivas suficientes como para lanzar una alerta.	
Entrada / Pasos de ejecución: Comprueba el número de teléfono del servicio asociado al paciente al que debe enviarse el SMS y se envía.	
Resultado Esperado: El sms es recibido por el mismo número que el pasado paramétricamente.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 8-2	**Número Historia de Usuario:** 8
Descripción de la Prueba: El contenido del Mensaje de Texto recibido es el mismo que el del enviado.	
Condiciones de Ejecución: Que haya pruebas positivas suficientes como para lanzar una alerta	
Entrada / Pasos de ejecución: Se procede a enviar el mensaje y se comprueba si el contenido recibido es idéntico al enviado.	
Resultado Esperado: El contenido coincide.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 8-3	**Número Historia de Usuario:** 8
Descripción de la Prueba: En caso de que la alerta no haya sido atendida en el margen de tiempo previsto, el SMS es reenviado.	
Condiciones de Ejecución: Que se haya enviado un mensaje previamente y el paciente no haya sido marcado como atendido en la web de sepSOS.	
Entrada / Pasos de ejecución: Se comprueba que tras pasar el periodo de tiempo indicado, el SMS vuelve a ser enviado.	
Resultado Esperado: SMS reenviado.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 8-4	**Número Historia de Usuario:** 8
Descripción de la Prueba: El SMS es enviado y recibido.	
Condiciones de Ejecución: Que haya pruebas positivas suficientes como para lanzar una alerta.	
Entrada / Pasos de ejecución: Se envía un SMS y se comprueba en el teléfono receptor que ha sido recibido sin problemas y sin un retraso de tiempo.	
Resultado Esperado: Se envía y se recibe correctamente	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 9-1	**Número Historia de Usuario:** 9
Descripción de la Prueba: El destinatario del correo lo recibe correctamente.	
Condiciones de Ejecución: Que haya pruebas positivas suficientes como para lanzar una alerta.	
Entrada / Pasos de ejecución: Se envía el correo electrónico de alerta a la dirección asociada al paciente y se comprueba que ha sido recibido correctamente y sin un retraso de tiempo.	
Resultado Esperado: El correo es recibido.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 9-2	**Número Historia de Usuario:** 9
Descripción de la Prueba: El contenido del correo enviado coincide con el recibido	
Condiciones de Ejecución: Que haya pruebas positivas suficientes como para lanzar una alerta.	
Entrada / Pasos de ejecución: Se envía el correo electrónico y se comprueba en el receptor que el contenido recibido coincide con el contenido enviado.	
Resultado Esperado: El contenido coincide.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 9-3	**Número Historia de Usuario:** 9
Descripción de la Prueba: En caso de que la alerta no sea atendida, se procederá a un reenvío del correo electrónico.	
Condiciones de Ejecución: Que se haya enviado un correo electrónico previamente.	
Entrada / Pasos de ejecución: Se observa si al transcurrir el tiempo necesario sin que se haya atendido al paciente, se reenvia el correo electrónico de alerta.	
Resultado Esperado: El correo es reenviado.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 10-1	**Número Historia de Usuario:** 10
Descripción de la Prueba: Los datos visualizados coinciden con los de la base de datos de SepSOS	
Condiciones de Ejecución: Que tengamos alguna alerta en la web.	
Entrada / Pasos de ejecución: Se comprueba que los datos mostrados en la web de sepSOS se correspondan con los datos de nuestra BD.	
Resultado Esperado: Los datos coinciden.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 10-2	**Número Historia de Usuario:** 10
Descripción de la Prueba: Al marcar un paciente como 'Visto', el filtro para alertas "No vistas" no muestra dicha alerta.	
Condiciones de Ejecución: Que tengamos alguna alerta sin ver en la Web	
Entrada / Pasos de ejecución: Se marca como vista una alerta en la web y se comprueba que al ver las "Alertas sin ver" no aparece la alerta marcada como vista.	
Resultado Esperado: La visualización se realiza ahora en "Alertas Vistas".	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 10-3	**Número Historia de Usuario:** 10
Descripción de la Prueba: Al eliminar un posible caso de Sepsis, desaparecen de la web los datos del paciente y de sus pruebas y pasan al histórico.	
Condiciones de Ejecución: Que la alerta esté marcada para archivar en la web y que Orion indique que el paciente lleva más de X días curado.	
Entrada / Pasos de ejecución: Una vez eliminada la alerta, se comprueba que ahora está en el histórico y que ya no aparece en la tabla alertas.	
Resultado Esperado: La alerta desaparece y en su lugar aparece en la tabla del histórico.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 10-4	**Número Historia de Usuario:** 10
Descripción de la Prueba: Las modificaciones de datos del paciente y los comentarios del médico se guardan correctamente.	
Condiciones de Ejecución: Que haya alguna alerta en la web.	
Entrada / Pasos de ejecución: El profesional realiza algún comentario o cambio en una alerta del paciente, comprobamos que dichos cambios se mantienen tanto en la web como en la BD.	
Resultado Esperado: Los cambios se guardan correctamente, pudiéndose visualizar la información en el siguiente acceso.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 11-1	**Número Historia de Usuario:** 11
Descripción de la Prueba: Las variables y datos precargados mostrados en la configuración coinciden con los de la base de datos de sepSOS	
Condiciones de Ejecución: Que tenga que realizarse la precarga	
Entrada / Pasos de ejecución: Se realiza una precarga de datos para el inicio de la aplicación. Comprobación de que los datos precargados coinciden con los de la base de datos.	
Resultado Esperado: Los datos mostrados son los correctos	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 11-2	**Número Historia de Usuario:** 11
Descripción de la Prueba: Al realizar un cambio en cualquiera de los datos, estos se almacenan y se conservan.	
Condiciones de Ejecución: Que realicemos algún cambio en valores umbrales, intervalo de refresco, etc…	
Entrada / Pasos de ejecución: Al realizar los cambios, estos son conservados en nuestra BD	
Resultado Esperado: El resultado se almacena en la base de datos de la aplicación.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 11-3	**Número Historia de Usuario:** 11
Descripción de la Prueba: El programa se detiene mostrando antes un mensaje de alerta.	
Condiciones de Ejecución: Que se realice alguna parada o cambio que comprometa la integridad de la aplicación.	
Entrada / Pasos de ejecución: Se intenta realizar una parada del servicio o un cierre inadecuado.	
Resultado Esperado: El programa se detiene, mostrando previamente un mensaje avisando de los peligros que conlleva detener la ejecución.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 11-4	**Número Historia de Usuario:** 11
Descripción de la Prueba: Al realizar un cambio, el programa pregunta si estamos seguros de que queremos realizar ese cambio.	
Condiciones de Ejecución: Que intentemos realizar algún cambio importante en nuestra aplicación.	
Entrada / Pasos de ejecución: Intentamos cambiar valores umbrales de una prueba.	
Resultado Esperado: Pop-up de confirmación, solicitando aceptar el cambio o denegarlo.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 12-1	**Número Historia de Usuario:** 12
Descripción de la Prueba: Visualización correcta de las acciones realizadas por los usuarios de “sepSOS”	
Condiciones de Ejecución: Que algún usuario haya realizado alguna actividad.	
Entrada / Pasos de ejecución: Comprobación de que las acciones realizadas por un usuario son almacenadas.	
Resultado Esperado: Los datos mostrados son correctos.	
Evaluación de la Prueba: Positivo	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 13-1	**Número Historia de Usuario:** 13
Descripción de la Prueba: Los casos positivos de sepsis se almacenan/recuperan/borran de forma correcta del histórico.	
Condiciones de Ejecución: Que hayan alertas en el histórico o marcadas para archivar.	
Entrada / Pasos de ejecución: Se archiva una alerta. Comprobamos que pasa a estar en el histórico y que los datos mostrados son correctos.	
Resultado Esperado: El acceso a los casos archivados es correcto.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 14-1	**Número Historia de Usuario:** 14
Descripción de la Prueba: La búsqueda devuelve valores correspondientes a los buscados por el usuario, funcionando correctamente.	
Condiciones de Ejecución: Que haya alertas archivadas en el histórico	
Entrada / Pasos de ejecución: Se realiza una búsqueda en el histórico	
Resultado Esperado: Los datos obtenidos son satisfactorios.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 14-2	**Número Historia de Usuario:** 14
Descripción de la Prueba: El filtrado devuelve solo los valores que le solicitamos (a partir de un rango, solo determinado grupo de enfermos, etc…)	
Condiciones de Ejecución: Que haya alertas archivadas en el histórico	
Entrada / Pasos de ejecución: Se realiza un filtrado en el histórico	
Resultado Esperado: Devuelve solo los resultados filtrados, sin datos que pertenezcan a información que no nos interesa.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 14-3	**Número Historia de Usuario:** 14
Descripción de la Prueba: Los gráficos estadísticos y los porcentajes mostrados se corresponden con los datos de “sepSOS”.	
Condiciones de Ejecución: Que haya alertas archivadas en el histórico	
Entrada / Pasos de ejecución: Comprobamos que las estádisticas se corresponden con los resultados arrojados por la aplicación de sepSOS.	
Resultado Esperado: Los datos obtenidos pertenecen a un estudio sobre las alertas recibidas en “sepSOS”, consiguiendo una información coherente y verídica.	
Evaluación de la Prueba: Positiva	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 15-1	**Número Historia de Usuario:** 15
Descripción de la Prueba: Al finalizar la ventana activa de un dato, este es borrado.	
Condiciones de Ejecución: Que haya algún dato.	
Entrada / Pasos de ejecución: Se comprueba la validez de un dato.	
Resultado Esperado: Cuando se sobrepasa el periodo de validez de un dato, se borra.	
Evaluación de la Prueba: Positiva	

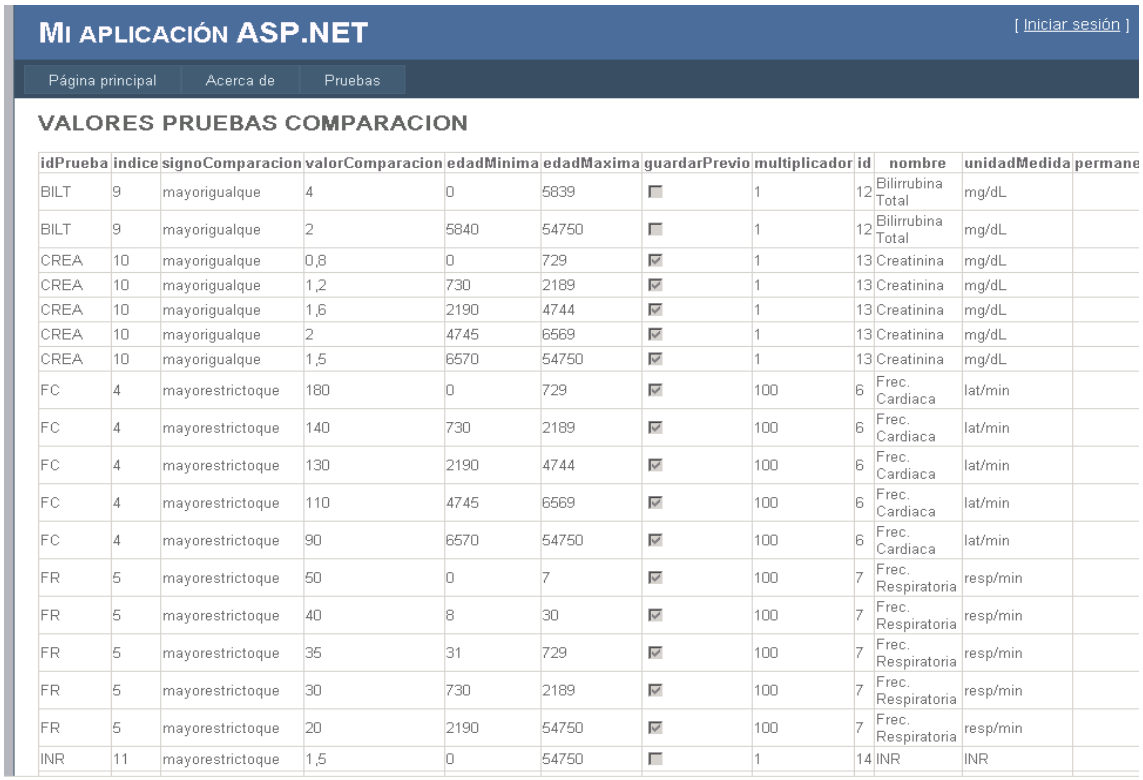
Caso de Prueba de Aceptación	
Código Caso de Prueba: 15-2	**Número Historia de Usuario:** 15
Descripción de la Prueba: Al recibir un dato de nuevo, aunque el periodo de validez del anterior no haya finalizado, este se sobrescribe por el más reciente, pues se considera más preciso.	
Condiciones de Ejecución: Que haya un dato antiguo y uno nuevo.	
Entrada / Pasos de ejecución: Se comprueban que al entrar un nuevo valor, este machaca al anterior.	
Resultado Esperado: Al entrar un nuevo dato, aunque tuviéramos uno anterior sin sobrepasar el umbral de validez, este macha al anterior, siendo el nuevo el valor utilizado para el diagnóstico.	
Evaluación de la Prueba: Positiva	

Web de Pruebas:

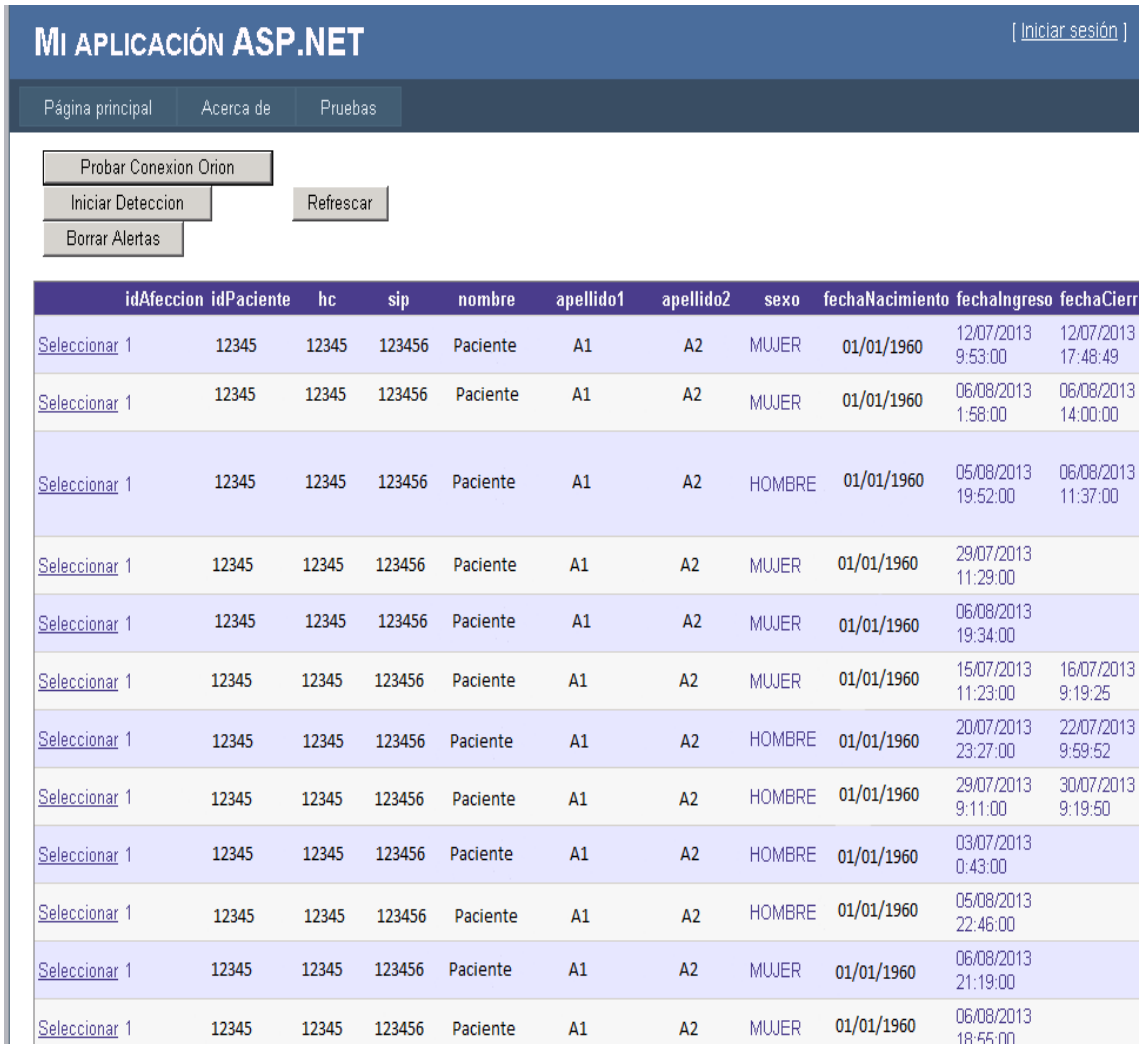
Dentro del proyecto de Visual Studio “SepSOS” podemos encontrar la biblioteca “SepSOS”, de la cual no hablamos en la fase de diseño, al tratarse de una web de pruebas básica realizada para probar visualizaciones.

Se trata de una página maestra de Visual Studio por defecto junto con 2 tablas sin maquetas que contienen los valores de comparación que utilizamos para detectar la Sepsis, y la tabla de Alertas junto con 4 botones (Probar Conexión, Iniciar Detección, Borrar Alertas y Refrescar)

Es un funcionamiento muy básico, pero ha sido pensada para comprobar que se lanzaban alertas y que estas se borraban.



idPrueba	indice	signoComparacion	valorComparacion	edadMinima	edadMaxima	guardarPrevio	multiplicador	id	nombre	unidadMedida	permane
BILT	9	mayorigualque	4	0	5839	☐	1	12	Bilirrubina Total	mg/dL	
BILT	9	mayorigualque	2	5840	54750	☐	1	12	Bilirrubina Total	mg/dL	
CREA	10	mayorigualque	0,8	0	729	☑	1	13	Creatinina	mg/dL	
CREA	10	mayorigualque	1,2	730	2189	☑	1	13	Creatinina	mg/dL	
CREA	10	mayorigualque	1,6	2190	4744	☑	1	13	Creatinina	mg/dL	
CREA	10	mayorigualque	2	4745	6569	☑	1	13	Creatinina	mg/dL	
CREA	10	mayorigualque	1,5	6570	54750	☑	1	13	Creatinina	mg/dL	
FC	4	mayorestrictoque	180	0	729	☑	100	6	Frec. Cardiaca	lat/min	
FC	4	mayorestrictoque	140	730	2189	☑	100	6	Frec. Cardiaca	lat/min	
FC	4	mayorestrictoque	130	2190	4744	☑	100	6	Frec. Cardiaca	lat/min	
FC	4	mayorestrictoque	110	4745	6569	☑	100	6	Frec. Cardiaca	lat/min	
FC	4	mayorestrictoque	90	6570	54750	☑	100	6	Frec. Cardiaca	lat/min	
FR	5	mayorestrictoque	50	0	7	☑	100	7	Frec. Respiratoria	resp/min	
FR	5	mayorestrictoque	40	8	30	☑	100	7	Frec. Respiratoria	resp/min	
FR	5	mayorestrictoque	35	31	729	☑	100	7	Frec. Respiratoria	resp/min	
FR	5	mayorestrictoque	30	730	2189	☑	100	7	Frec. Respiratoria	resp/min	
FR	5	mayorestrictoque	20	2190	54750	☑	100	7	Frec. Respiratoria	resp/min	
INR	11	mayorestrictoque	1,5	0	54750	☐	1	14	INR	INR	



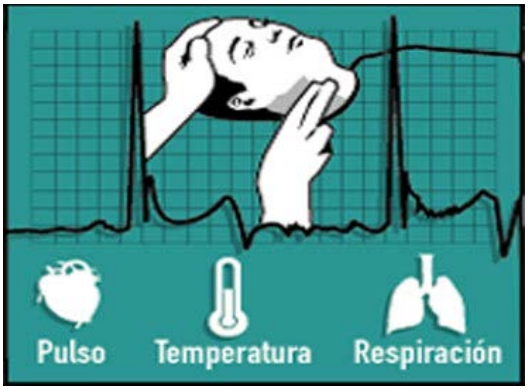
	idAfeccion	idPaciente	hc	sip	nombre	apellido1	apellido2	sexo	fechaNacimiento	fechaIngreso	fechaCierr
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	MUJER	01/01/1960	12/07/2013 9:53:00	12/07/2013 17:48:49
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	MUJER	01/01/1960	06/08/2013 1:58:00	06/08/2013 14:00:00
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	HOMBRE	01/01/1960	05/08/2013 19:52:00	06/08/2013 11:37:00
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	MUJER	01/01/1960	29/07/2013 11:29:00	
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	MUJER	01/01/1960	06/08/2013 19:34:00	
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	MUJER	01/01/1960	15/07/2013 11:23:00	16/07/2013 9:19:25
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	HOMBRE	01/01/1960	20/07/2013 23:27:00	22/07/2013 9:59:52
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	HOMBRE	01/01/1960	29/07/2013 9:11:00	30/07/2013 9:19:50
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	HOMBRE	01/01/1960	03/07/2013 0:43:00	
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	HOMBRE	01/01/1960	05/08/2013 22:46:00	
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	MUJER	01/01/1960	06/08/2013 21:19:00	
Seleccionar	1	12345	12345	123456	Paciente	A1	A2	MUJER	01/01/1960	06/08/2013 18:55:00	

8. Ejemplo de Funcionamiento

Para que se entienda el funcionamiento de la aplicación, vamos a hacer un simulacro de detección de Sepsis para un paciente recién llegado a Urgencias.

I. El paciente llega al hospital debido a un malestar y unos síntomas cualesquiera (dolor de cabeza, fiebre, angustia...). Ya que no era una cita programada, accede a la sección de Urgencias.

II. El paciente es llamado a la consulta de Triaje, donde es atendido por un enfermero que le hace las pertinentes tomas de constantes (temperatura, frecuencia cardiaca...).

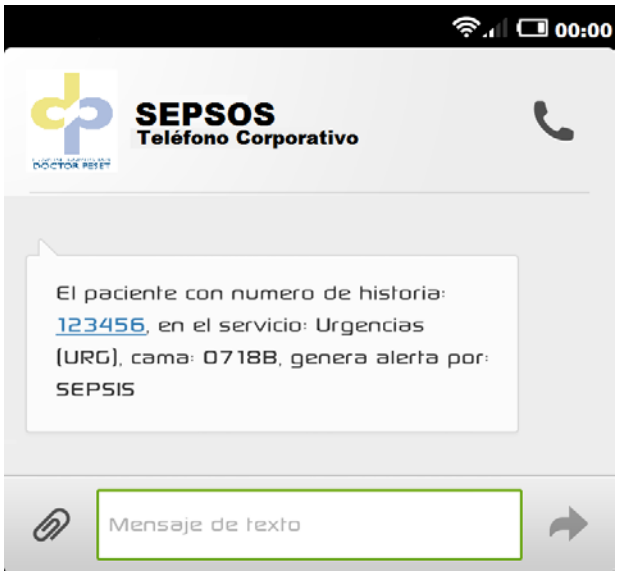


III. Tras el triaje, un médico examina al paciente y si lo considera oportuno, solicita una analítica para dicho paciente.

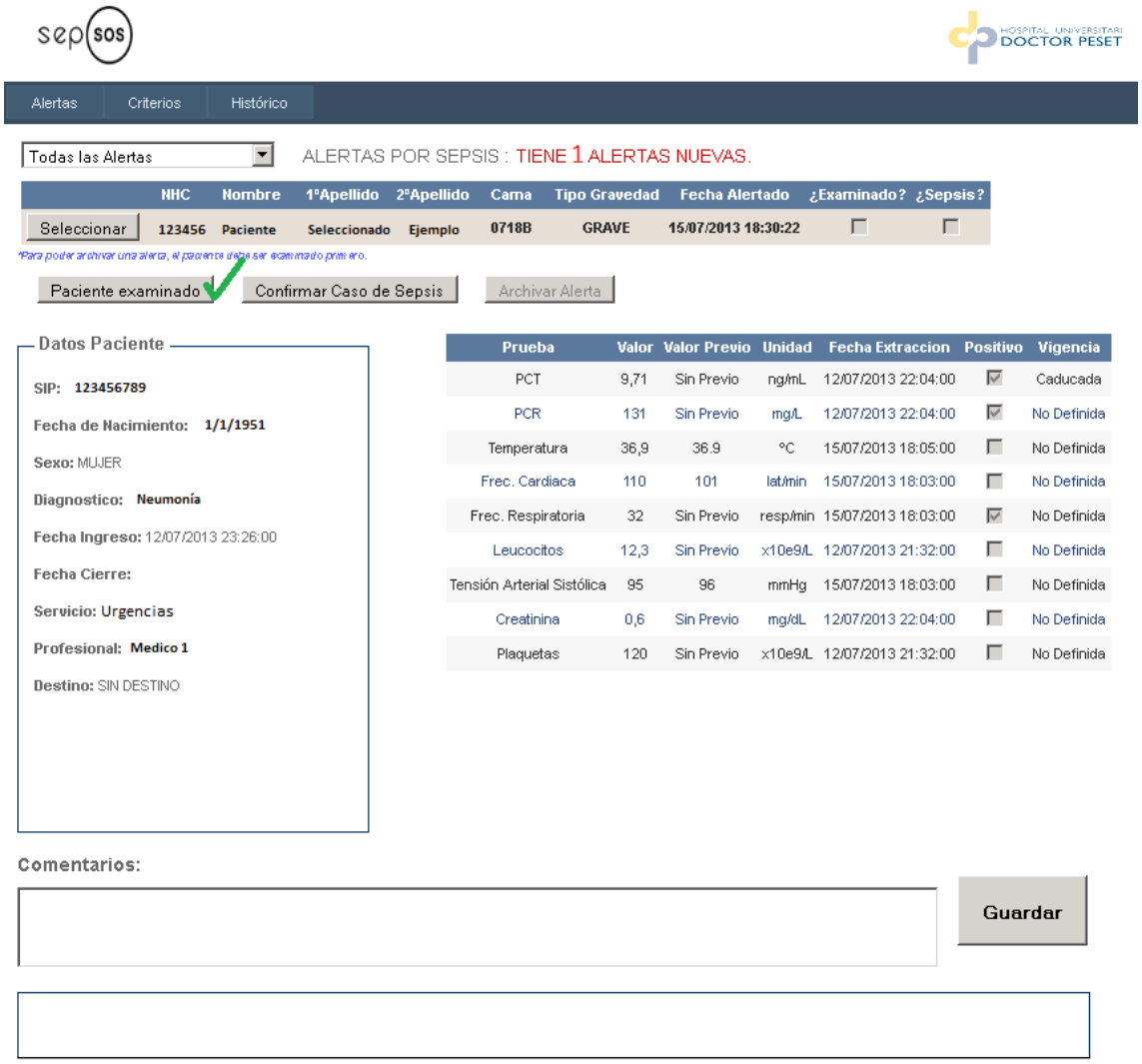
IV. Nuestro módulo de detección intercepta los resultados de los análisis y de las constantes, incluso antes de que el médico tenga dichos resultados, reduciendo el retraso con el que el paciente es tratado. Analizando dichos resultados, podemos detectar un posible caso de Sepsis reduciendo la ventana de tiempo de trato al paciente, algo fundamental con dicha enfermedad.

V. Varias pruebas superan el valor umbral declarado para la Sepsis, y el valor de control supera el declarado para un caso de Sepsis, de manera que lanzamos una alerta.

VI. El profesional responsable recibe en su móvil corporativo la alerta, indicándole los datos del paciente más relevantes a la hora de localizarlo y tratarlo.

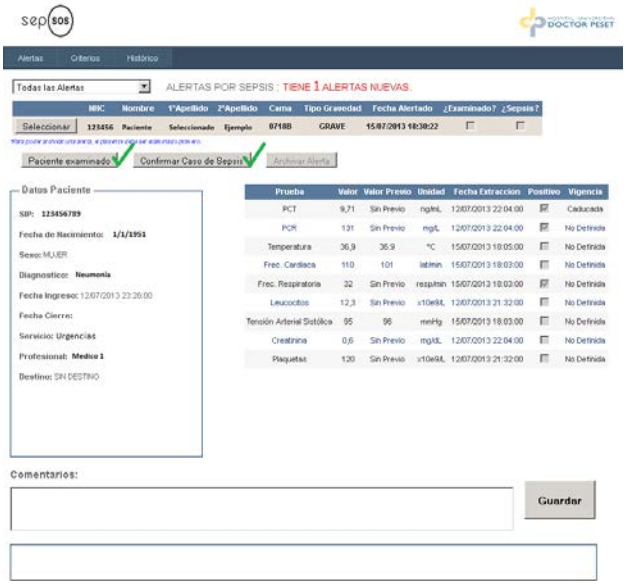


VII. El médico se conecta a su terminal y accede a la web de *SepSOS*. Comprueba que tiene una nueva alerta, la cual coincide con la recibida en su teléfono. A continuación, revisa los datos del paciente, así como los valores de las pruebas, y marca la alerta como **Vista**.

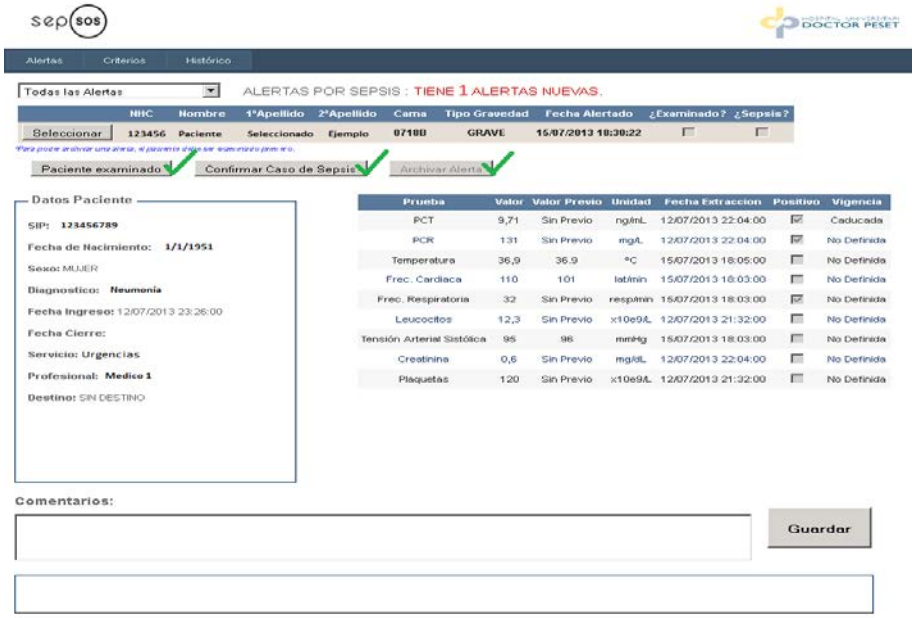


VIII. El profesional comienza a tratar al paciente mediante antibióticos para la Sepsis, además de cualquier otro tratamiento que requiera dicho paciente, pues puede estar en el hospital por otros motivos diferentes a la Sepsis.

IX. Si durante la estancia o el tratamiento, se confirma que es un caso de Sepsis, el profesional confirma mediante la web que dicha alerta era correcta y que efectivamente, se trataba de un caso de sepsis. Además de para comprobar si es útil y efectivo tener dicha aplicación en el hospital, marcar los casos de sepsis positivos (y por tanto, diferenciarlos de los falsos positivos que de la aplicación) sirve para calibrar el programa, pudiéndolo hacer menos sensible a alertas.



X. Si creemos que la alerta ya ha cumplido su función o la estancia en el hospital del paciente ha finalizado, el profesional marcará la opción para archivar la alerta. Si en su momento no había confirmado si se trataba de un caso de sepsis o no, un cuadro de dialogo le pedirá que confirme o no la afección. Una vez Orion indique que el paciente ha sido curado, la alerta se archivará en el histórico.



9. Instalación del Servicio

Sopesamos dos maneras para instalar el Servicio Windows en el servidor del Hospital:

- ✓ La primera es crear un paquete de instalación que incluya todo lo necesario para el funcionamiento de la aplicación. Desechamos esta opción debido a que no nos interesaba ya que el servidor de instalación coincidía con el servidor de desarrollo y a los cambios en la aplicación cada dos por tres (pues al comienzo, SEPSOS puede producir demasiados falsos positivos y necesitará pasar por una fase de calibrado, en la cual se modificarán los valores umbrales, la candidad de alertas, la frecuencia de las alertas, etc...)

- ✓ La segunda opción es utilizar las Visual Studio Tools. Dichas herramientas nos permiten abrir una consola y ejecutar la instrucción ***installutil*** sobre el archivo ".exe" de la aplicación (que se encuentra en la carpeta "BIN"). Dicha instrucción instala el servicio directamente en el equipo. A continuación en la pestaña *Servicios* de Windows Server, podemos encontrar el servicio, y haciendo doble click en él, accedemos a las propiedades y las credenciales de ejecución. Le indicamos que ante un fallo reinicie el servicio, dado que los cambios de versión de Orion, y sobre todo en el Hospital Peset que lleva la versión más reciente son muy corrientes.
  Nos hemos decantados por esta opción por ser más rápida (ya que para probar un servicio, requiere instalarlo) y por aprovechar las utilidades de Visual Studio.

10. Mejoras implementadas

Pese a los requisitos captados en primera instancia con el Hospital, se han producido ciertas mejoras que no estaban planificadas desde el principio:

- Sin duda alguna la mejora más importante es la ***Multi-Enfermedad***.
  La aplicación ha sido concebida para poder utilizarse tanto para detectar Sepsis, como para detectar Neumonía, Gastritis e incluso Cáncer, por ejemplo. El flujo de análisis normal se divide en un hilo para cada enfermedad. Cada hilo comprueba en su objeto de datos análisis "Afección", los datos que necesita para detectar la afección, y procesa todas las pruebas que necesita. Puede darse el caso que un mismo paciente tenga insertada en la base de datos dos (o más) pruebas idénticas, negativa en un caso para una afección, pero positiva para otra enfermedad. Cada Afección tiene sus pruebas propias, sus desencadenantes y su tabla Hash de pacientes. Esto puede verse por ejemplo en el diagrama de clases. No se ha mencionado durante el resto del proyecto por ser una funcionalidad independiente de la Sepsis.

- El programa ha sido pensado para ser multi-centro. Puede aplicarse a cualquier centro médico sin ningún problema.

- La aplicación también tiene diferentes maneras de alertar, según el centro médico que la utilice, puede necesitar alertar de una manera u otra.

- Las pruebas son solicitadas una sola vez y los resultados son usados por cada hilo de análisis que lo necesite, así nos evitamos pedir duplicados.

11. Posibles mejoras a implementar

Una vez finalizado el desarrollo de la aplicación, surgen nuevos requisitos o funcionalidades que por tiempo, o por no estar pensado desde un primer momento de esa forma, no han podido ser aplicadas.

- Añadir queries de consulta dinámicamente, actualmente es necesario recompilar la clase del centro médico en cuestión (algo que buscábamos evitar).

- Quizá se podría buscar una forma de relacionar directamente para cada resultado de prueba que venga, encontrar su valor de comparación acorde a su edad, sin necesidad de comprobar entre qué valores de edades nos encontramos.

- Con el paso del tiempo, se ha solicitado guardar el previo de más pruebas de las que estaba inicialmente pensado. Por esto hay demasiados fallos contra nuestra base de datos para saber cuál es el previo. Habría que mantener los valores previos más a mano (añadirlo al objeto Paciente en otro array, tener en Memoria Principal la tabla de resultados, etc...)

- Podrían declararse fácilmente (con cambios mínimos) varios orígenes de datos de diferentes hospitales y tenerlo todo centralizado en un único servidor.

12. Repercusiones de SepSOS

20minutos.es
20minutos EE UU | 20minutos México

BAZAAR
LA REVISTA DE LAS MUJERES QUE MUEVEN EL MUNDO
20minutos.t

Valencia

La automatización de la detección precoz de la sepsis en el Peset permite reducir la mortalidad por esta patología

La automatización de la detección precoz de la sepsis en el Hospital Universitario Doctor Peset de Valencia ha permitido reducir la mortalidad por esta patología en el centro hasta un 23,1 por ciento, diez puntos por debajo de la media nacional, que se sitúa en un 33% por ciento.

EUROPA PRESS. 10.04.2013

La automatización de la detección precoz de la sepsis en el Hospital Universitario Doctor Peset de Valencia ha permitido reducir la mortalidad por esta patología en el centro hasta un 23,1 por ciento, diez puntos por debajo de la media nacional, que se sitúa en un 33% por ciento.


Ampliar foto

El conseller de Sanitat, Manuel Llombart, ha visitado este miércoles el hospital para conocer los tres primeros meses de funcionamiento de su Unidad de Sepsis y su aplicación informática desarrollada para automatizar la detección precoz de la sepsis grave y el shock séptico, una de las principales causas de muerte en los hospitales españoles.

Así, ha destacado que se han conseguido "unos resultados increíbles" ya que el centro durante estos tres meses ha conseguido situar su tasa de mortalidad por sepsis 6 puntos por debajo de la media de la Comunitat Valenciana, que es del 29 por ciento. Especialmente se ha reducido la mortalidad por sepsis grave que, concretamente en este hospital valenciano, ha pasado de 22,5 por ciento a 15,2 por ciento en este periodo(7 puntos menos).

Llombart ha destacado el esfuerzo conjunto de los profesionales del Hopital Universitario Doctor Peset para conseguir disminuir la mortalidad por sepsis y lograr un manejo más eficiente de los pacientes afectados por esta patología.

La Unidad, tras un proceso iniciado en 2010 que ha involucrado prácticamente a todas las especialidades de este centro, se puso en marcha en enero de 2013. Se trata de la segunda que existe en España y la quinta de Europa con estas características.

De este modo, se ha conseguido que la estancia media hospitalaria de los pacientes con sepsis se haya reducido hasta los 10,2 días, un dato "sensiblemente inferior" a la media nacional, puesto que los pacientes afectados por esta enfermedad permanecen ingresados una media de 21 días en los hospitales españoles.

Asimismo, también es menor a la que existía en este hospital hace sólo unos meses (11,5 días) que, a pesar de ser muy buena, se ha conseguido reducir aún más. "Esta experiencia pionera y sus resultados podrán exportase a otros centros para seguir mejorando la atención sanitaria", ha comentado.

Casos de sepsis

La sepsis es una respuesta del organismo ante una infección que, clínicamente, puede manifestarse de diversas formas, desde en un simple resfriado hasta, en casos graves, en un fallo de los órganos vitales que puede provocar la muerte de la persona afectada.

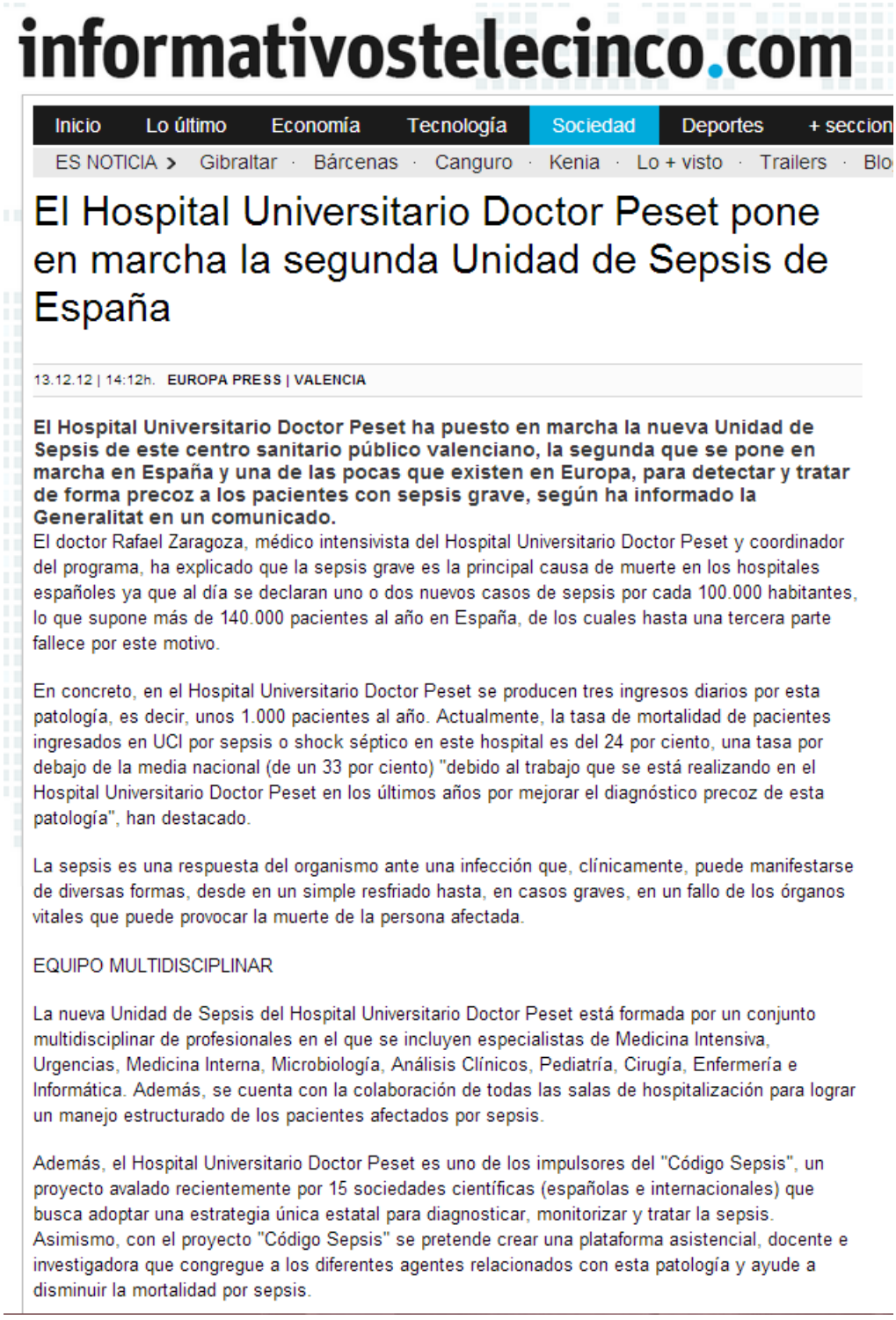
Desde su puesta en funcionamiento, la Unidad de Sepsis del Hospital Universitario Doctor Peset ha atendido a 193 pacientes, de los cuales 151 tenían sepsis grave y shock séptico (un 67,2 por ciento con sepsis grave y un 32,8 por ciento con shock séptico).

Por su parte, el coordinador de la Unidad de Sepsis del Hospital Doctor Peset, el doctor Rafael Zaragoza, ha explicado que en este hospital se producen tres ingresos diarios por sepsis, es decir, unos 1.000 pacientes al año. En líneas generales, al día se declaran 1 ó 2 nuevos casos de sepsis por cada 100.000 habitantes, lo que supone más de 140.000 pacientes al año en España, de los cuales hasta una tercera parte fallece por este motivo.

Esta enfermedad, que suele producir fiebre y dolor, tiene como mejor tratamiento el antibiótico, que será más efectivo si se aplica antes de una hora desde la aparición de la sepsis y, como máximo, en el plazo de tres horas. De ahí la importancia de la detección precoz de esta patología.

Precisamente para detectar a los pacientes con sepsis cuanto antes y administrarles el tratamiento correcto, la Unidad de Informática del Hospital Universitario Doctor Peset ha diseñado un sistema de detección automatizado ante la sospecha de sepsis en pacientes ingresados, a partir de los datos disponibles en su historia clínica electrónica (ORION Clinic). Este sistema de control actúa como un motor de búsqueda que analiza y cruza continuamente datos analíticos y de constantes vitales en busca de valores que se correspondan con un cuadro de sepsis.

Link a la Noticia: http://www.20minutos.es/noticia/1782476/0/



informativostelecinco.com

Inicio | Lo último | Economía | Tecnología | Sociedad | Deportes | + seccion

ES NOTICIA > Gibraltar · Bárcenas · Canguro · Kenia · Lo + visto · Trailers · Blo

El Hospital Universitario Doctor Peset pone en marcha la segunda Unidad de Sepsis de España

13.12.12 | 14:12h. EUROPA PRESS | VALENCIA

El Hospital Universitario Doctor Peset ha puesto en marcha la nueva Unidad de Sepsis de este centro sanitario público valenciano, la segunda que se pone en marcha en España y una de las pocas que existen en Europa, para detectar y tratar de forma precoz a los pacientes con sepsis grave, según ha informado la Generalitat en un comunicado.

El doctor Rafael Zaragoza, médico intensivista del Hospital Universitario Doctor Peset y coordinador del programa, ha explicado que la sepsis grave es la principal causa de muerte en los hospitales españoles ya que al día se declaran uno o dos nuevos casos de sepsis por cada 100.000 habitantes, lo que supone más de 140.000 pacientes al año en España, de los cuales hasta una tercera parte fallece por este motivo.

En concreto, en el Hospital Universitario Doctor Peset se producen tres ingresos diarios por esta patología, es decir, unos 1.000 pacientes al año. Actualmente, la tasa de mortalidad de pacientes ingresados en UCI por sepsis o shock séptico en este hospital es del 24 por ciento, una tasa por debajo de la media nacional (de un 33 por ciento) "debido al trabajo que se está realizando en el Hospital Universitario Doctor Peset en los últimos años por mejorar el diagnóstico precoz de esta patología", han destacado.

La sepsis es una respuesta del organismo ante una infección que, clínicamente, puede manifestarse de diversas formas, desde en un simple resfriado hasta, en casos graves, en un fallo de los órganos vitales que puede provocar la muerte de la persona afectada.

EQUIPO MULTIDISCIPLINAR

La nueva Unidad de Sepsis del Hospital Universitario Doctor Peset está formada por un conjunto multidisciplinar de profesionales en el que se incluyen especialistas de Medicina Intensiva, Urgencias, Medicina Interna, Microbiología, Análisis Clínicos, Pediatría, Cirugía, Enfermería e Informática. Además, se cuenta con la colaboración de todas las salas de hospitalización para lograr un manejo estructurado de los pacientes afectados por sepsis.

Además, el Hospital Universitario Doctor Peset es uno de los impulsores del "Código Sepsis", un proyecto avalado recientemente por 15 sociedades científicas (españolas e internacionales) que busca adoptar una estrategia única estatal para diagnosticar, monitorizar y tratar la sepsis. Asimismo, con el proyecto "Código Sepsis" se pretende crear una plataforma asistencial, docente e investigadora que congregue a los diferentes agentes relacionados con esta patología y ayude a disminuir la mortalidad por sepsis.

Link a la Noticia: http://www.telecinco.es/informativos/sociedad/Hospital-Universitario-Doctor-Unidad-Espana_0_1524075396.html

Noticias agencias

Unidad Sepsis Doctor Peset sitúa tasa mortalidad 10 puntos por debajo media

10-04-2013 / 15:11 h EFE

La Unidad de Sepsis del Hospital Doctor Peset de Valencia, la segunda de estas características que funciona en España, ha conseguido situar su tasa de mortalidad por sepsis en un 23,1 por ciento, diez puntos por debajo de la media nacional, situada en el 33 por ciento

Estos son algunos de los resultados de los tres primeros meses de funcionamiento de esta unidad, que hoy ha visitado el conseller de Sanidad, Manuel Llombart, para conocer la aplicación informática desarrollada en el centro para automatizar la detección precoz de la sepsis grave y el shock séptico, una de las principales causas de muerte en los hospitales españoles.

Este sistema de control actúa como un motor de búsqueda que analiza y cruza continuamente datos analíticos y de constantes vitales en busca de valores que se correspondan con un cuadro de sepsis y en cuanto se detecta genera una alerta utilizando distintas tecnologías (SMS, móviles, e-mail y web de sepsis).

En respuesta a la alerta, el equipo de Sepsis valora al paciente para confirmar la sospecha y realizar las actuaciones necesarias de forma precoz para controlar la evolución, según las fuentes, que han indicado que en tres meses este sistema se ha activado en 108 ocasiones.

Además, se ha diseñado un sistema para la gestión de los casos de sepsis confirmados, lo que permite al profesional disponer de un protocolo o guía de actuación para el tratamiento de los pacientes durante el proceso y hacer así un mejor seguimiento.

Llombart ha destacado el "esfuerzo conjunto" de los profesionales del hospital para conseguir disminuir la mortalidad por sepsis y lograr un manejo más eficiente de los pacientes afectados por esta patología.

"Tras un proceso iniciado en 2010 que ha involucrado a prácticamente todas las especialidades de este centro, en enero de 2013 se puso en marcha la Unidad, la segunda que existe en España y la quinta de Europa con estas características", ha destacado el conseller.

En estos tres primeros meses, y tras una fase de rodaje que empezó en noviembre de 2012, la Unidad ha conseguido situar su tasa de mortalidad por sepsis en un 23,1 por ciento, diez puntos por debajo de la media nacional, que se sitúa en un 33 por ciento, y 6 puntos por debajo de la media de la Comunitat Valenciana, del 29 %.

Además, la estancia media hospitalaria de los pacientes con sepsis se ha reducido hasta los 10,2 días, un dato sensiblemente inferior a la media nacional, puesto que los pacientes afectados por esta enfermedad permanecen ingresados una media de 21 días en los hospitales españoles.

Twittear 0

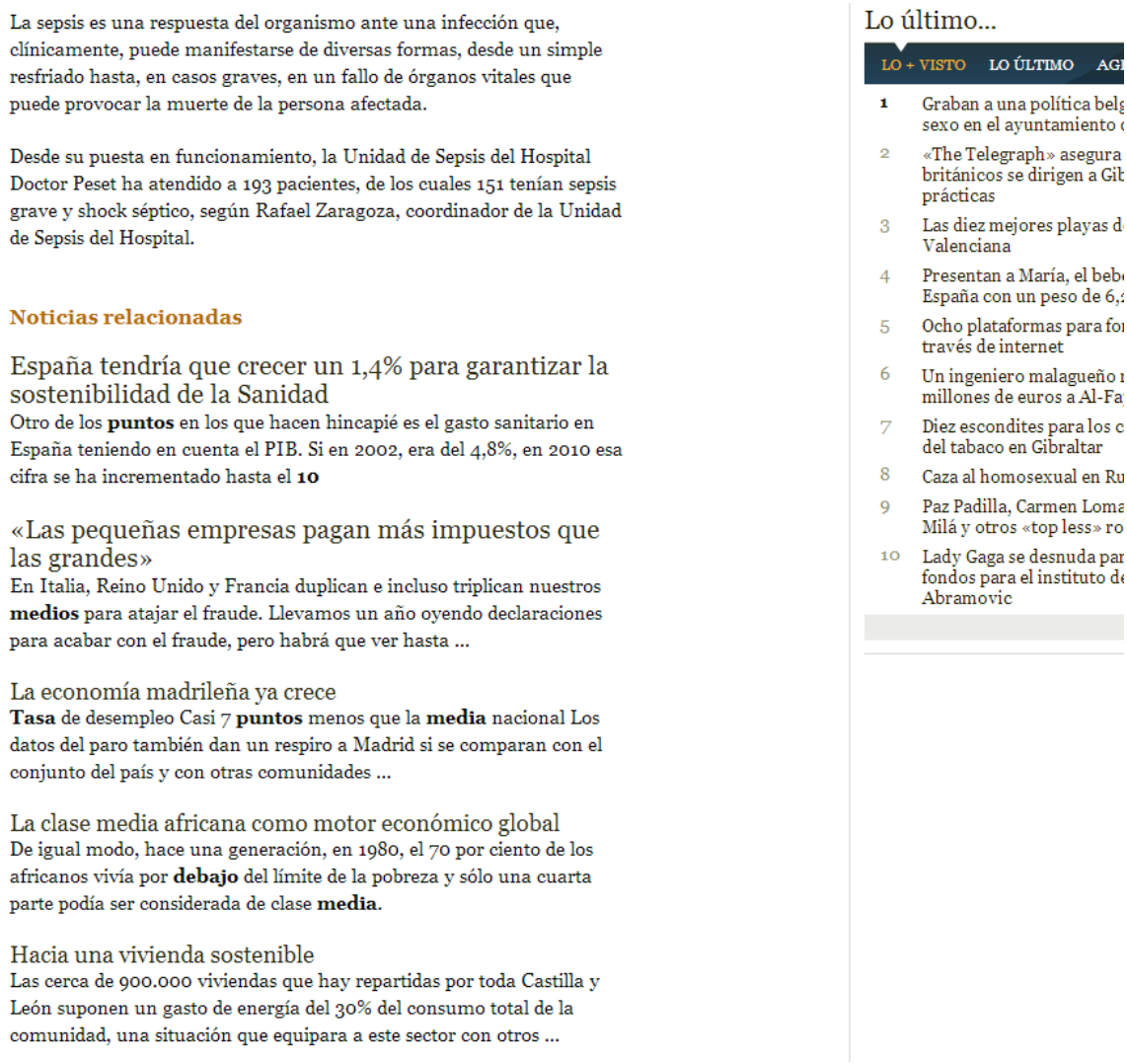
Por comunidades

- Andalucía
- Aragón
- Baleares
- Cantabria
- Castilla La Mancha
- Castilla y León
- Cataluña
- Ceuta
- Comunidad Valenciana
- País Vasco
- Córdoba
- Extremadura
- Galicia
- La Rioja
- Madrid
- Melilla
- Murcia
- Navarra
- Sevilla
- Canarias

- Todas las comunidades
- Noticias Internacionales
- Noticias Deportivas

Enlaces

- ABC.es
- Lotería del Niño 2012
- Buscador Lotería del Niño 2012
- Lotería de Navidad 2012
- Elecciones Andalucía



La sepsis es una respuesta del organismo ante una infección que, clínicamente, puede manifestarse de diversas formas, desde un simple resfriado hasta, en casos graves, en un fallo de órganos vitales que puede provocar la muerte de la persona afectada.

Desde su puesta en funcionamiento, la Unidad de Sepsis del Hospital Doctor Peset ha atendido a 193 pacientes, de los cuales 151 tenían sepsis grave y shock séptico, según Rafael Zaragoza, coordinador de la Unidad de Sepsis del Hospital.

Noticias relacionadas

España tendría que crecer un 1,4% para garantizar la sostenibilidad de la Sanidad
Otro de los **puntos** en los que hacen hincapié es el gasto sanitario en España teniendo en cuenta el PIB. Si en 2002, era del 4,8%, en 2010 esa cifra se ha incrementado hasta el **10**

«Las pequeñas empresas pagan más impuestos que las grandes»
En Italia, Reino Unido y Francia duplican e incluso triplican nuestros **medios** para atajar el fraude. Llevamos un año oyendo declaraciones para acabar con el fraude, pero habrá que ver hasta ...

La economía madrileña ya crece
Tasa de desempleo Casi 7 **puntos** menos que la **media** nacional Los datos del paro también dan un respiro a Madrid si se comparan con el conjunto del país y con otras comunidades ...

La clase media africana como motor económico global
De igual modo, hace una generación, en 1980, el 70 por ciento de los africanos vivía por **debajo** del límite de la pobreza y sólo una cuarta parte podía ser considerada de clase **media**.

Hacia una vivienda sostenible
Las cerca de 900.000 viviendas que hay repartidas por toda Castilla y León suponen un gasto de energía del 30% del consumo total de la comunidad, una situación que equipara a este sector con otros ...

Lo último...

LO + VISTO | LO ÚLTIMO | AG

1 Graban a una política belg sexo en el ayuntamiento
2 «The Telegraph» asegura británicos se dirigen a Gib prácticas
3 Las diez mejores playas d Valenciana
4 Presentan a María, el beb España con un peso de 6,
5 Ocho plataformas para fo través de internet
6 Un ingeniero malagueño millones de euros a Al-Fa
7 Diez escondites para los c del tabaco en Gibraltar
8 Caza al homosexual en Ru
9 Paz Padilla, Carmen Loma Milá y otros «top less» ro
10 Lady Gaga se desnuda par fondos para el instituto d Abramovic

Link a la Noticia: http://www.abc.es/agencias/noticia.asp?noticia=1391591

Estos son algunos ejemplos de la repercusion que ha tenido la creación de un departamento específico para la Sepsis en el Hospital Universitario Doctor Peset Aleixandre y el desarrollo de una aplicación de Detección Automática de la Sepsis Grave.

13. Bibliografía

-Sepsis:

- http://es.wikipedia.org/wiki/Sepsis
- http://www.nlm.nih.gov/medlineplus/spanish/ency/article/000666.htm
- Documentos Internos (pdfs, extractos de reuniones, etc...) del Hospital Doctor Peset Aleixandre

-Metodologias:

- http://es.wikipedia.org/wiki/Metodolog%C3%ADa
- http://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema
- http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software
- http://www.extremeprogramming.org/

-Bases de Datos:

- http://pic.dhe.ibm.com/infocenter/idshelp/v117/index.jsp
- Libro: ***Beginning SQL SERVER 2005 Programming***

-Visual Studio:

- http://msdn.microsoft.com/es-es/vstudio/aa718325.aspx
- Videotutorial: Desarrollo de Aplicaciones y Acceso a Datos con C#.NET
  http://videocursos.es/FormsAdoCS.aspx

-ASP:

- http://www.asp.net/
- Videotutorial: Aplicaciones Web con ASP.NET 2.0
  http://www.videotutoriales.es/temario-curso-asp2/

14. Opinión Personal

Como primer acercamiento al mundo laboral como ingenieros informáticos, la realización del proyecto final de carrera en el Hospital Peset Aleixandre nos ha venido perfecto para conocer un ambiente de trabajo, el desarrollo en equipo, el acercamiento a nuevas tecnologías...

Hay que tener en cuenta, que durante nuestro paso por la Universidad, los grupos de trabajo nunca han sido superiores a 4 o 5 personas. Formar parte de uno o varios departamentos de profesionales especializados de 30 o 40 personas nos ha enseñado como trabajar y organizarnos a la hora de un desarrollo en equipo.

Además, ocupar diferentes roles como Clientes, Analistas o Programadores nos ha permitido familiarizarnos con las diferentes ramas de la Ingeniería del Software, además de practicar con Metodologías Ágiles.

Desde un primer momento, hemos intentado suplir la necesidad del hospital de un sistema automatizado de diagnóstico de Sepsis, desarrollando un software a medida codo con codo con los trabajadores del departamento de informática, los cuales han aportado sus conocimientos específicos en cada área de la informática. Conocimientos que hemos tratado de absorber y de los cuales hemos querido aprender lo máximo posible.

También hemos podido descubrir por nosotros mismos los problemas que nos podemos encontrar a la hora de desarrollar software y que se llevan arrastrando desde la Crisis del Software, pues hemos visto cambios de especificaciones constantemente (lo cual no ha supuesto un problema debido al uso de metodologías ágiles) y un incremento del tiempo de desarrollo que se nos ha ido de las manos.

Finalmente, y a modo de resumen, se podría decir que la realización de este proyecto final de carrera ha sido un simulacro perfecto, a modo de aclimatación sobre lo que nos encontraremos al incorporarnos al mundo laboral, durante el cual hemos adquirido muchos conocimientos que hemos aplicado al desarrollo de la aplicación y que nos ha permitido crecer como persona y como informáticos.

15. Agradecimientos

En primer lugar, agradecer a José Miguel Puig, David Estellés y Emilio Villanueva su ayuda y su disposición a la hora de desarrollar la aplicación. Durante todo el proceso habéis estado disponibles para atendernos en todo momento, pese a vuestras diversas y abundantes obligaciones.
Muchísimas gracias.

En segundo lugar, no podríamos obviar al resto de personas del departamento de Informática del Peset. Desde José Manuel, Sergio, Eva, Antonio... hasta Manuela. Sería difícil nombraros a todos, pero nos habéis facilitado nuestra estancia allí y habéis colaborado durante todo el proyecto. Sin vosotros SepSOS no existiría.
Gracias a tod@s.

Además debemos nombrar al Doctor Rafael Zaragoza y al resto de miembros relacionados con la UCI y la Unidad de Sepsis. Nos ha ayudado muchísimo poder asistir a las reuniones internas, visitar la UCI y saber como trabajais. El deseo de todos es que este proyecto funcione perfectamente y de verdad esperamos que así sea.
Gracias por el apoyo recibido.

Y por último y no menos importante, gracias a Jon Ander. Nuestro director del proyecto de la Universidad nos ha recibido gustosamente en su despacho incluso en verano, ha contestado nuestros e-mails desde fuera del país y se tomó la molestia de desplazarse hasta el Hospital Doctor Peset Aleixandre para conocer más sobre la enfermedad, sobre el desarrollo de nuestra aplicación y sobre la Unidad de Sepsis. Ha sido una integración digna de admirar, que nosotros no podemos hacer más que agradecer.
Muchas gracias por el tiempo invertido en este proyecto y por las molestias tomadas.