Universitat Autònoma
de Barcelona

Escritorio de trabajo del médico

Memoria del proyecto de
Ingeniería Técnica en
Informática de Sistemas
realizado por

Daniel Cáliz Vílchez

y dirigido por

Jordi Pons Aróztegui

Escola d'Enginyeria

Sabadell, julio de 2011

Jordi Pons Aróztegui,
profesor de la Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección por **Daniel Caliz Vilchez**

Y para que conste firma la presente.
Sabadell, **julio** de **2011**

Firmado: **Jordi Pons Aróztegui**

Daniel Fuster y **Oscar Lechago**,
de la empresa UNIT4,

CERTIFICAN:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección por **Daniel Caliz Vilchez**

Y para que conste firman la presente.
Sabadell, **julio** de **2011**

Firmado: **Daniel Fuster** y **Oscar Lechago**

HOJA DE RESUMEN – PROYECTO FIN DE CARRERA DE LA ESCUELA DE INGENIERIA

Titulo del proyecto: Escritorio de trabajo del médico	
Autor: Daniel Cáliz Vílchez	**Data:** *Julio de 2011*
Tutor: Jordi Pons Aróztegui	
Titulación: Ingeniería Técnica en Informática de Sistemas	

Palabras clave

- Castellano: escritorio, médico, multiaplicación.
- Catalán: escriptori, metge, multiaplicació.
- Inglés: desktop, doctor, multitask.

Resumen del proyecto

- Castellano:

El proyecto se desarrolla en el marco de un convenio entre la Universidad Autónoma de Barcelona y la empresa UNIT4 en las oficinas de UNIT4 Ibérica en Barberà del Vallès.

Consiste en realizar y disponer de una nueva versión del escritorio, o estación, de trabajo del médico totalmente remodelada, más visual y energizante con la ayuda de las nuevas usabilidades de creación de escritorios de la empresa, mejorando sus fortalezas a la vez que reduciendo las debilidades y aportando nuevas perspectivas de usabilidad que permitirán potenciar la productividad de los usuarios en el empleo de la aplicación.

- Catalán:

El projecte es desenvolupa en el marc d'un conveni entre la Universitat Autònoma de Barcelona i l'empresa UNIT4 en les oficines de UNIT4 Ibèrica a Barberà del Vallès.

El projecte consisteix a realitzar i disposar d'una nova versió de l'escriptori, o estació, de treball del metge totalment remodelada, més visual i energitzant amb l'ajuda de les noves usabilitats de creació d'escriptoris de l'empresa millorant les seves fortaleses al mateix temps que reduint les debilitats i aportant noves perspectives d'usabilitat que permetran potenciar la productivitat dels usuaris en l'ús de l'aplicació.

- Inglés:

The project is developed under an agreement between the Autonomous University of Barcelona and the company UNIT4 at the offices of Iberian UNIT4 in Barberà del Vallès.
The project is to perform and have a new version of the desktop, or station, doctor working completely remodeled, more visual and energetic with the help of the new building usability desktop creation improve your strengths while reducing the weaknesses and providing new perspectives of usability that will enhance the productivity of users in the use of the application.

Tabla de contenido

Índice de Tablas

Índice de Figuras

1. Introducción

1.1. Convenio UAB-Empresa

Éste proyecto se ha desarrollado dentro del marco de un convenio de colaboración entre la Universitat Autònoma de Barcelona y la empresa UNIT4, en el cual, se estipulan 560 horas para realizar un trabajo de desarrollo de software. Mediante este convenio, el alumno puede adquirir conocimientos de cómo funciona una empresa de desarrollo de software de gestión y de tecnologías de la información, a la vez que este desarrollo sirve como trabajo de final de carrera.

1.2. La Empresa

UNIT4 Ibérica es la filial española del grupo UNIT4. UNIT4 Ibérica es un fabricante de software de gestión (ERP, CRM, RRHH, BI, CPM) para empresas, organismos públicos y sanidad, tanto en modalidad de venta como de pago por uso (ASP). Tiene más de cuarenta años de experiencia en el mundo de la ingeniería de software. Posee un amplio abanico de soluciones de gestión de última generación lo cual la hace ser una de las primeras empresas españolas en tecnologías de la información y comunicaciones.

Surge el año 2007 de la integración de las empresas Agresso Spain, S.L. y Centro de Cálculo de Sabadell, S.A.U., compañías ambas con una dilatada experiencia en el sector de las tecnologías de la información y las comunicaciones (TIC). Desde el 1 de febrero de 2010, la compañía cambia su denominación de CCS Agresso a UNIT4, al mismo tiempo que el resto de filiales nacionales del grupo al que pertenece.

Es la primera empresa española certificada en I+D+i por el Ministerio de Industria, Comercio y Turismo.

Unit4 cuenta con, aproximadamente 4.230 empleados y con oficinas y distribuidores en todo el mundo para garantizar un acceso fácil y local a las ventas, servicios y soporte. Se encuentran en: Alemania, Australia, Bélgica, Canadá, Dinamarca, España, Estados Unidos, Estonia, Francia, Holanda, Hungría, Irlanda, Malasia, Noruega, Portugal, Reino Unido, República Checa, Singapur, Suráfrica, Suecia y Uganda.

UNIT4 Ibérica aplica a sus soluciones la tecnología *karat*. Su característica principal es que permite la ejecución del software desarrollado con ella en múltiples plataformas tecnológicas, sistemas operativos (MS Windows, Linux, MAC OS) y bases de datos sin modificar el código fuente. Además, ofrece herramientas para la personalización de procesos de negocio en las soluciones, cuya evolución queda garantizada al mismo tiempo que el estándar.

Uno de los productos con los que trabaja la empresa es Unit4 *ekon*, software de planificación de recursos empresariales (ERP) basado y diseñado para ayudar a las empresas a lograr un proceso total de integración de sus herramientas. Actualmente se está trabajando en la versión *ekon 6*, que incluye una nueva interfaz de usuario denominada *Walnut*, basada en la "User experience". Con el fin de optimizar el trabajo

cotidiano con la aplicación, centra la innovación en aprovechar la experiencia de los usuarios. Además es mucho más dinámica, personalizable y potencia la operatividad.

Basado en este producto, Unit4 ofrece diversas soluciones de software según las necesidades de las empresas: Finanzas, Logística, Human Capital Management, Salus, etc. Siguiendo las diferentes soluciones del producto *ekon,* la empresa se divide en diferentes departamentos que se centran única y exclusivamente en el desarrollo y mantenimiento del producto correspondiente.

Por los requisitos del proyecto, éste se sitúa dentro del departamento de Desarrollo *ekon Salus,* que es el grupo que se encarga del desarrollo, soporte y gestión de los mantenimientos del producto *ekon Salus*, una avanzada solución informática de gestión sanitaria integral para las empresas y organismos del sector sanitario que potencia y facilita al máximo el tratamiento y la calidad de la información asistencial.

1.3. Objetivos del proyecto

El objetivo principal del proyecto consiste en realizar y disponer de una nueva versión del escritorio o estación de trabajo del médico con la ayuda de la nueva usabilidad de creación de escritorios de *ekon 6,* que permitirán potenciar la productividad de los usuarios en el empleo de la aplicación.

· En el ámbito de la empresa UNIT4, los objetivos son:

- Crear una aplicación que facilite y optimice el trabajo del médico.
- Adaptar las funcionalidades para que sean sencillas y útiles, y que disponga de una alta capacidad para ser entendido, usado y ser muy atractivo para el usuario.
- Debe incorporar todas las herramientas necesarias para que el usuario tenga una experiencia satisfactoria.

· Los objetivos del proyecto a nivel personal son:

- Adquirir experiencia en el trabajo diario dentro de una gran empresa y aprender metodologías de trabajo.
- Aprender Java, un lenguaje de programación orientado a objetos con un gran uso en el mundo laboral.
- Desarrollar una herramienta desde cero, donde poder realizar un ciclo completo en el desarrollo de software (investigación, análisis, codificación y pruebas).

1.4. Contenido de la memoria

La memoria del proyecto consta de los siguientes capítulos:

1. **Introducción:** Apartado en el cual se explica de forma breve y concisa el marco de ubicación del proyecto, la empresa y los objetivos marcados.
2. **Estudio de Viabilidad**: En este apartado se hace una pequeña descripción del sistema actual, del sistema propuesto (objetivos, funcionalidades, etc.), y de los

beneficios y riesgos del proyecto. Además observaremos la planificación inicial de las tareas a realizar en el proyecto y su duración.

3. **Fase de análisis**: En este apartado se explican de forma detallada los requisitos funcionales y no funcionales que se han determinado a partir del análisis de la situación y de los objetivos a conseguir, así como los diferentes usuarios de la aplicación.
4. **Fase de diseño**: En primer lugar, se hace una pequeña introducción a la tecnología utilizada en el desarrollo de la aplicación. Además, se explica más detalladamente cómo se ha diseñado y estructurado la aplicación y su interfaz.
5. **Fase de codificación y pruebas**: Apartado en el cual se describe como se han hecho las diferentes partes de la aplicación y la interfaz, así como el estilo de codificación usado y la estructura del código. Además, describimos en este apartado todas las pruebas realizadas sobre la aplicación y el resultado de éstas para comprobar la eficacia del programa.
6. **Conclusiones**. En este apartado, vemos cuáles han sido los objetivos conseguidos, cuales han quedado pendientes, las líneas de trabajo abiertas, así como una valoración personal sobre el proyecto.
7. **Bibliografía**: Detalle de recursos informativos utilizados para realizar el proyecto.

2. Estudio de Viabilidad

2.1. Situación actual

2.1.1. Contexto

Como se ha explicado con anterioridad, Unit4 dispone de *ekon Salus*, un producto basado en la plataforma tecnológica de *karat*, para la gestión de las empresas relacionadas con el sector de Sanidad, que aporta un nuevo concepto de soluciones basado en la independencia total y real de entornos, facilitando el tratamiento de la información asistencial y aportando una solución global que agrupa el conjunto de aplicaciones necesarias en las entidades sanitarias, así como:

- Gestión asistencial: listas de espera, admisiones, urgencias, agendas y citaciones, farmacia, etc.
- Gestión clínica: estación médica y de enfermería, historia clínica, etc.
- Gestión económica: cuadros de mando, control de costes, estadísticas, facturación, gestión de cobros, etc.
- Organismos públicos: facturación, listas de espera, etc.
- Herramientas de administración.

Ekon Salus nació con la finalidad de servir como plataforma de gestión sanitaria en el control de empleados y pacientes para las empresas que precisen ayuda en el tema.

En la actualidad, se dispone de un escritorio del médico en el entorno, o versión, ekon 3 para facilitar o ayudar, en modo visual, en el trabajo del personal Sanitario (en especial del médico) que emplea el producto *ekon Salus*.

La finalidad de este escritorio, es ofrecer al médico un entorno sencillo, agradable y útil que agrupa todas sus tareas importantes y más frecuentemente utilizadas. Todo ello, sin la necesidad de tener que navegar a través de las múltiples pantallas o módulos del producto *ekon Salus*.

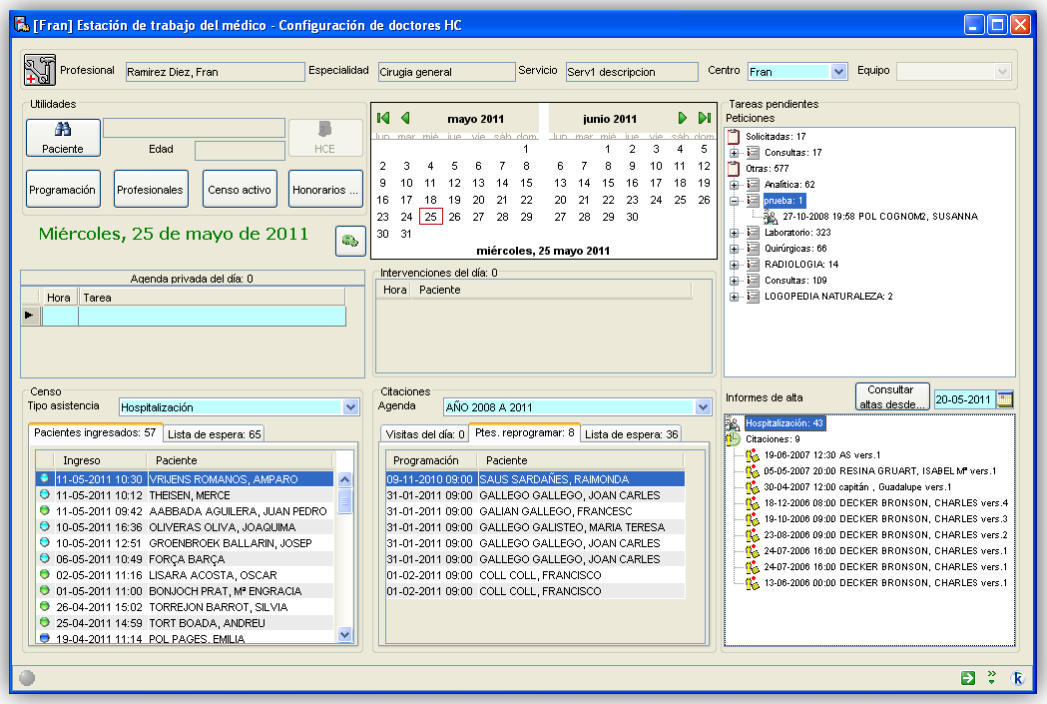

Figura 1: Esquema del modelo BoxApp

A través de este entorno, el usuario puede acceder y visualizar todos los pacientes, sus datos clínicos y sus asistencias que según la configuración están relacionados con él, así como poder visualizar sus tareas pendientes e intervenciones.

Por todo ello, como resumen, podríamos decir que el escritorio o estación de trabajo del médico es un entorno o interfaz desde la cual el usuario puede acceder a gran parte, sino toda, de la información de *ekon Salus*, de manera más sencilla y entendedora, evitando tener que navegar por multitud de pantallas.

Cabe destacar, que la principal tarea del escritorio es mostrar información, pero no añadir ni modificarla, ya que de esta tarea se encargan el resto de pantallas o módulos de *ekon Salus.* Lo que sí permite, además de visualizar la información más importante y relevante, es acceder a dichos resto de módulos, a partir de los cuales se gestiona el resto de funcionalidades de *ekon Salus*.

2.1.2. Diagnóstico y conclusiones

Como se ha dicho antes, el escritorio del médico actual fue un primer entorno creado para facilitar el control de pacientes y tareas del médico en la gestión sanitaria de manera sencilla y rápida a través del producto *ekon Salus*. Pero poco a poco, desde su desarrollo, las herramientas *karat* y los productos *ekon* han ido evolucionando, mejorándose y añadiendo nuevas funcionalidades, posibilitando mejoras visuales y dinámicas en el escritorio provocando que sea más atractivo y entendible para el usuario.

Mejoras como la nueva interfaz de usuario y estilo de codificación estándar, englobadas dentro del proyecto *Walnut*, o la herramienta de creación de escritorios de *karat*, que permite más dinamicidad en los escritorios y que éstos puedan ser adaptados al gusto de cada usuario.

Precisamente, una de las grandes carencias del actual escritorio del médico es su estaticidad: el entorno o interfaz siempre muestra la misma información, solo cambiando los datos según el usuario de la aplicación.

Por ello, siguiendo estas mejoras podemos decir que el actual escritorio del médico se ha quedado un poco desfasado. El producto actual está forzado visualmente, está muy cargado, es poco dinámico y energizante, y además no permite ningún tipo de modificación.

Por todo ello, se ha propuesto la realización de un nuevo escritorio del médico utilizando todas las nuevas funcionalidades y mejoras que presentan las nuevas herramientas de *karat*.

2.2. Objetivos y análisis de requerimientos

2.2.1. Sistema propuesto

El objetivo propuesto es disponer de una nueva versión de escritorio del médico del producto *Salus* totalmente remodelado, más visible y energizante, reforzando sus fortalezas pero reduciendo las debilidades visuales que presenta e integrando nuevas perspectivas de usabilidad y flexibilidad de cara al usuario final.

Entre otras las características del escritorio del médico definitivo deberían ser:

- Más visual y energizante.
- Menos forzado y cargado.
- Más dinámica.
- El usuario podrá adaptarse y gestionarse a su medida el entorno.
- Debe tener una alta capacidad para ser entendido, usado y ser muy atractivo para el usuario.
- Interfaz de usuario utilizando la nueva usabilidad de *ekon 6, Walnut*.
- Utilización de la herramienta de creación de escritorios de *ekon 6* siguiendo el modelo de Desarrollo *BoxApp*, que será explicado más adelante.
- Producto multiidioma. Los idiomas contemplados serán el español y el inglés.

2.2.2. Descripción y priorización de objetivos del proyecto

El objetivo propuesto para este proyecto es que el nuevo escritorio no solo debe tener una gran funcionalidad sino que también tenga las siguientes características:

O.1. Hacer más interactivo y visual el actual escritorio del médico.

O.2. Debe disponer de una alta capacidad para ser entendido, usado y ser muy atractivo para el usuario.

O.3. Deberá utilizar la nueva usabilidad de *ekon 6*, *Walnut*, proporcionando una UI ("User Interface") basada en la 'User Experience'.

O.4. Deberá seguir y basarse en el modelo de Desarrollo *BoxApp*.

O.5. Deberán dar la posibilidad al profesional de configurarse a su manera y gusto el entorno de trabajo.

O.6. Producto multi-idioma. Idiomas contemplados español e inglés.

Objetivo	Crítico	Primario	Secundario
O1	X		
O2	X		
O3	X		
O4	X		
O5		X	

Tabla 1: Tabla de clasificación de objetivos

2.2.3. Definiciones, acrónimos y abreviaciones

Karat: Plataforma tecnológica para la gestión de empresas, que aporta un nuevo concepto de soluciones basado en la independencia total y real de entornos.

Ekon Salus: producto basado en la plataforma tecnológica de *karat*, para la gestión de las empresas relacionadas con el sector de Sanidad.

Walnut: Nueva interfaz gráfica de *karat*, basada en la 'User Experience'.

LOPD: Ley Orgánica de protección de datos.

2.2.4. Partes interesadas

A continuación presentemos las partes interesadas del proyecto (StakeHolders):

Nombre	Descripción	Responsabilidad
Departamento de Desarrollo ekon Salus	Departamento de la empresa UNIT4 dedicado al producto ekon Salus	Departamento encargado de crear y gestionar los mantenimientos del producto ekon Salus

Tabla 2: Tabla de las partes interesadas del proyecto

2.2.5. Equipo de proyecto

En la figura siguiente se muestra cómo queda confeccionado el equipo del proyecto y una pequeña explicación de la responsabilidad de cada miembro:

Nombre	Descripción	Responsabilidad
Jordi Pons Aróztegui	Tutor del Proyecto(DP)	Supervisa el trabajo del alumno durante el proyecto.
Ramón Torres	Jefe de Proyecto UNIT4 (CP)	Define, gestiona y controla el proyecto.
Daniel Cáliz Vílchez	Analista (A)	Desarrolla el estudio de viabilidad y la planificación. Análisis de la aplicación: arquitectura, metodología, especificación, estándares... Participa en el diseño y validación.
Daniel Fuster y Oscar Lechago	Analista (A)	Asesoramiento en el análisis de la aplicación y asesoramiento técnico a lo largo del proyecto.
Daniel Cáliz Vílchez	Programador (P)	Desarrolla la aplicación de acuerdo al análisis y la planificación prevista. Hace la implantación del proyecto.
Daniel Cáliz Vílchez	Diseñador (D)	Diseña y desarrolla el aspecto de la aplicación de acuerdo al análisis y las normas de usabilidad establecida.

Tabla 3: Tabla del equipo de proyecto

2.2.6. Requisitos funcionales

A continuación podemos ver los requisitos funcionales necesarios que debe cumplir el nuevo escritorio de trabajo del médico:

RF1. Dar la posibilidad al usuario de modificar, auto gestionar y auto adaptarse el entorno a su medida, como más le guste.

RF2. Basarse en el modelo BoxApp de creación de escritorios y aplicaciones, dispondrá de 3 vistas: normal, ampliada y minimizada.

RF3. Al 'loguearse' el profesional, la aplicación cargará su configuración e información, basándose en sus pacientes.

RF4. Posibilidad de acceder a la información de sus pacientes, sus historiales clínicos y de asistencias, así como su programación para la fecha indicada.

RF5. Mantenimiento de los Antecedentes y alergias de sus pacientes según la asistencia.

RF6. Mantenimiento de las órdenes médicas de sus pacientes.

RF7. Dispondrá de un apartado de Tareas Pendientes, donde podrá gestionar sus peticiones e informes de alta.

RF8. Acceso al Censo del Hospital según el tipo de asistencia y su urgencia.

RF9. Disponer de una Agenda personal para el usuario, donde poder mirar todas sus tareas pendientes y/o realizadas.

RF10. Acceso a sus Intervenciones, tanto realizadas como pendientes de realizar, a partir de una fecha indicada.

2.2.7. Requisitos no funcionales

Los requisitos no funcionales de nuestra aplicación serán:

RNF1. Cumplimiento de la LOPD por lo que hace referencia a los ficheros de datos y a los derechos de los clientes.

RNF2. Cumplimiento de la guía de estilo *Java* para *karat.*

RNF3. Cumplimiento de la guía de estilo *Walnut*.

RNF4. Normalización de la base de datos y accesos según el estándar SQL 99 (ISO/IEC 9075:1999).

RNF5. El software ha de permitir ser ampliable y modificable.

RNF6. Tolerancia a errores y acciones incorrectas.

2.2.8. Restricciones del sistema

El escritorio de trabajo del médico deberá cumplir con una serie de limitaciones o restricciones:

1. Utilización de la herramienta *karat*.
2. Seguimiento de los estándares de UNIT4.
3. El proyecto debe estar finalizado antes del 30 de Junio de 2011.

2.2.9. Catalogación y priorización de los requisitos

A continuación se puede observar los diferentes requisitos según su priorización:

Req.	Esencial	Condicional	Opcional
RF1	X		
RF2	X		
RF3	X		
RF4	X		
RF5	X		
RF6	X		
RF7	X		
RF8	X		
RF9	X		
RF10	X		
RNF1		X	
RNF2		X	
RNF3		X	
RNF4		X	
RNF5		X	
RNF6		X	

Tabla 4: Tabla de catalogación de los requisitos

2.3. Alternativas

2.3.1. Alternativa 1

La única alternativa posible, por requisitos, restricciones y características es adaptar o actualizar el actual escritorio del médico ya existente a las nuevas necesidades y mejoras de los productos *karat*, cubriendo, además, todos los requisitos que se nos indican.

Para ello, se necesita crear desde cero nuestros formularios y nos ayudaremos de la nueva usabilidad de *ekon 6*, *Walnut*, y de la herramienta de creación de escritorios de *karat*, que más adelante se explicará.

Se seguirá el actual módelo de Datos de ekon Salus, ya que nuestro escritorio será una pequeña aplicación de este producto.

2.3.2. Solución propuesta

En la siguiente comparativa se ponen en común los diferentes aspectos más relevantes y la solución más optima.

	Coste adquisición	**Coste adaptación**	**Coste exploración**	**Coste desarrollo**	**Ajuste requerimientos**
Alternativa1	0€	bajo	bajo	alto	alto

Al disponer de una sola alternativa, puesto que las especificaciones son muy concretas, se ha escogido la Alternativa 1, que aunque supone un tiempo más elevado de desarrollo se ajusta perfectamente a las necesidades y requerimientos que se nos indican.

2.4. Planificación del proyecto

2.4.1. Recursos del proyecto

- Herramientas de planificación y control: Microsoft Project.

1. Recursos humanos:

Recursos humanos	Valoración
Jefe del proyecto	100 €/h
Analista	50 €/h
Programador	30 €/h
Técnico de pruebas	20 /h

2. Recursos materiales:

Se utilizarán los recursos materiales disponibles en la empresa:

- Licencias karat, Oracle, Microsoft Office y Microsoft Project.
- PC personal: PC – Intel Core 2Q8400 @ 2.66GHz
  6,00 GB RAM
  Disco duro 500GB
  Conexión a internet
- Software: Windows 7 y Eclipse Helios.

2.4.2. Tareas del proyecto

A continuación explicamos mediante esta tabla las diferentes tareas que componen el proyecto así como su duración en días:

Nombre de tarea	Duración	Predecesora
Escritorio de trabajo del médico	147 días	
Curso de Formación	10 días	
Planificación	6 días	
Estudio de viabilidad	2 días	
Aprobación Estudio de Viabilidad	1 día	4
Plan del Proyecto	3 días	4
Aprobación Plan de Proyecto	1 día	6
Análisis de la aplicación	14 días	
Reunión con departamento de ekon Salus	1 día	3
Análisis de requisitos	2 días	9
Investigación y formación sobre el escritio del médico actual	7 días	10
Documentación del Análisis	3 días	11

Aprobación del Análisis	1 día	12
Diseño de la Aplicación	34 días	13
Investigación sobre la herramienta de creacion de escritorios	20 días	
Diseño de la interfaz gráfica	10 días	15
Documentación del diseño	3 días	16
Aprobación de diseño	1 día	17
Desarrollo de la Aplicación	46 días	14
Preparación del entorno de desarrollo	1 día	
Desarrollo de los formularios	15 días	20
Desarrollo de las clases de personalización	25 días	21
Integración en la interfaz gráfica	5 días	22
Test y pruebas	17 días	19
Pruebas unitarias	2 días	
Pruebas funcionales	5 días	25
Pruebas de integración	2 días	26
Documentación de las pruebas	2 días	27
Aprobación de las pruebas y resultados	1 día	28
Corrección de errores	5 días	29
Generación de la Documentación	5 días	24
Cierre del proyecto	1 día	31
Defensa del Proyecto	1 día	32

Tabla 5: Tabla de las tareas del proyecto

2.4.3. Planificación temporal

El proyecto se desarrollará desde Noviembre de 2010 hasta Junio de 2011 con una dedicación de 16 horas semanales. El total de horas serán 560.

-Fecha de comienzo: 16/11/2010

- Fecha de finalización: 30/06/2010

A continuación se muestra el diagrama de Gantt de la planificación del proyecto con las tareas numeradas para facilitar su comprensión:

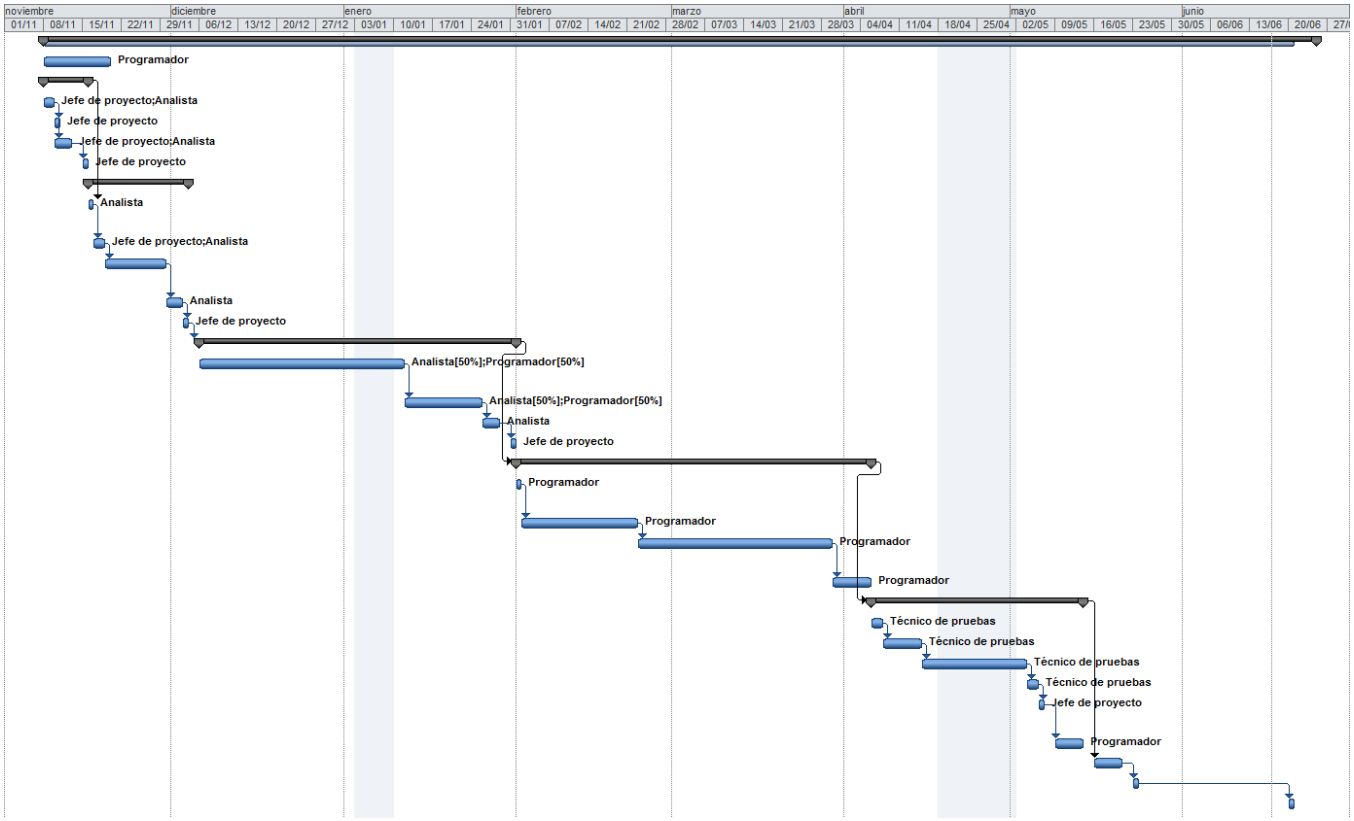

Figura 2: Diagrama de Gantt

2.4.4. Definición y catalogación de riesgos

Riesgo	Consecuencia
R1. Planificación temporal optimista	No se acaba en la fecha prevista y aumentan los recursos.
R2. Falta de alguna tarea necesaria	Menor calidad y funcionalidad de la aplicación.
R3. Cambio de requisitos	Retraso en el desarrollo y el resultado de la aplicación.
R4. Dificultad para acceder a los StakeHolders	Faltan requisitos o son inadecuados. Retraso del proyecto e insatisfacción de los usuarios
R5. Fase de pruebas incorrecta	Falta de calidad e insatisfacción de los usuarios.
R6. Herramientas de desarrollo inadecuadas o en proceso de finalización	Retraso en la finalización del proyecto y no cumplimiento de los objetivos del proyecto.
R7. Falta de conocimientos	Pérdida de tiempo den la formación y retraso del proyecto

Tabla 6: Tabla de definición de riesgos

Riesgo	Probabilidad	Impacto
1	Alta	Crítico
2	Alta	Crítico
3	Alta	Crítico
4	Media	Crítico
5	Alta	Crítico
6	Media	Catastrófico
7	Alta	Crítico

Tabla 7: Tabla de catalogación de riesgos

2.4.5. Plan de contingencia

	Solución
R1	Aplazar o recortar alguna funcionalidad que nos crítica. Aumentar horas de trabajo.
R2	Revisar planificación y estudio de viabilidad.
R3	Si son funcionalidades críticas, recortar funcionalidades no críticas. Aumentar horas de trabajo.
R4	Planificar un calendario de reuniones. SCRUM.
R5	Diseñar las pruebas de test con antelación, realizar test automáticos.
R6	Prevenir herramientas alternativas.
R7	Más horas de trabajo, retraso del proyecto.

Tabla 8: Tabla del plan de contingencia

2.4.6. Análisis de costes

Coste Personal:

Función	Coste/h	Horas	Coste total
Jefe de Proyecto	100€/h	50	5000€
Analista	50€/h	90	4500€
Programador	30€/h	380	11400€
Técnico de pruebas	20€/h	40	800€

Tabla 9: Tabla de análisis de costes

Total: 12.700€

Coste de los Recursos:

En nuestro caso, los recursos no tienen ningún coste adicional, ya que las herramientas o material necesario para realizar el proyecto, como el ordenador, nos lo facilita la empresa UNIT4.

Además, tenemos licencias corporativas tanto en Microsoft Office 2007 como en MS Project, así como en la base de datos Oracle. Utilizamos la herramienta corporativa *karat*, con su respectiva licencia, y la herramienta gratuita de Eclipse Helios para desarrollar nuestro proyecto.

Resumen y análisis coste beneficio:

Coste de desarrollo del proyecto…… 12.700€
Coste de amortización del material…….. 0€
Total: 12.700€

Si tenemos en cuenta que el coste personal asignado es ficticio, ya que se correspondería solo al coste del proyecto realizado por un grupo de trabajo en cualquier empresa y no por un becario en prácticas de proyecto final de carrera, cuyo coste real es de 2300€ y que, además, disponemos de la herramienta *karat* para trabajar y todas las facilidades de la empresa UNIT4, podemos decir que el proyecto es viable y económicamente.

La amortización del proyecto a nivel económico será rápida, ya que la aceptación del nuevo escritorio será total en los usuarios que ya dispongan de la herramienta *ekon Salus*, ya que sigue las mismas pautas que el anterior pero con las mejoras visuales y en funcionalidad citadas anteriormente, que les facilitarán y ayudarán en su trabajo.

2.5. Conclusiones

2.5.1. Beneficios esperados

Por lo que se refiere a beneficios, primero constatar, que esta herramienta ya es conocida entre los clientes de la empresa y más concretamente entre los usuarios que emplean el producto *ekon Salus*, por lo que no requiere ningún coste en publicidad.

Este nuevo diseño del escritorio del médico, utilizando la nueva usabilidad del proyecto *Walnut*, provocará una visión del entorno de trabajo del médico más dinámica, ajustable, adaptable, energizante y más fácil de trabajar que el actual.

Además, mejorará la experiencia del usuario con la aplicación, ya que este podrá retocar a su gusto y semejanza el entorno de trabajo y adaptarlo a sus necesidades más inmediatas e importantes.

2.5.2. Inconvenientes

El único inconveniente previsto, es que los usuarios necesitarán la instalación previa del escritorio, y una pequeña explicación de cómo adaptarse y ajustarse el entorno a su gusto. A pesar de esto, la herramienta es bastante intuitiva, y los usuarios más experimentados no necesitarán grandes ayudas.

BENEFICIOS + INCONVENIENTES = PROYECTO VIABLE

3. Fase de Análisis

3.1. Perfiles de usuario

La aplicación solo es usada por un tipo de usuario:

PU1. Profesional: Accederá a la aplicación y tendrá acceso completo a todos los recursos del entorno. Este escritorio está pensado para cualquier tipo de personal médico de un centro sanitario que tenga relación con los pacientes.

3.2. Requisitos funcionales

A continuación se detallarán los requisitos funcionales de la aplicación:

RF1. Autogestión y modificación del entorno por parte del usuario:

Es uno de los requisitos más importantes e indispensables, por lo menos uno de los más tenidos en cuenta a la hora de proponer el desarrollo de este nuevo escritorio. Las nuevas herramientas *karat* permiten nuevas funcionalidades al usuario, como añadir, modificar o eliminar información de su entorno de trabajo.

Por lo tanto, el usuario podrá ver única y exclusivamente la información que más le satisfaga o que más utilidad y ayuda le proporcione. Podrá decidir qué información ve y cómo quiere verla.

Para ello en el desarrollo de la aplicación se crearán diferentes módulos para, después, poder dar a elegir al profesional los que más le gusten o satisfagan sus necesidades.

RF2. Basarse en el modelo *BoxApp* de creación de escritorios y aplicaciones:

Para ello, dispondremos de la herramienta de creación de escritorios de *karat.* Esta herramienta permite al usuario añadir información al escritorio a través de mini aplicaciones (o cajitas).

La información que el usuario quiere visualizar sobre cada aspecto o apartado del producto, se divide en formularios[1] individuales. Estos formularios se insertan en mini aplicaciones o cajitas de nuestro escritorio. Cada mini aplicación puede moverse e intercambiarse con otras mini aplicaciones a lo largo del escritorio, además de poder tener un color y medidas variables, y disponer de un refresco.

[1] Formulario es un objeto *karat* en el cual se muestra un conjunto de registros que contienen los datos relevantes con los cuales el usuario puede interactuar de forma directa para su modificación o visualización.

Las mini aplicaciones pueden estar, o no, relacionadas entre sí a través de parámetros de entrada y salida. El parámetro de salida de una puede ser el parámetro de entrada de otra. Normalmente, las cajitas que estén relacionadas entre sí llevarán el mismo color, para que sea fácil e intuitivo para el profesional.

Además, cada mini aplicación dispondrá de 3 vistas[2]: normal (box_std), minimizada (box_min) y ampliada (ventana base). Cada vista hace referencia a la manera en que el formulario es visto. La vista estándar corresponde a la ventana que se verá normalmente en el escritorio, la vista ampliada a la que veremos cuando ampliemos cualquier vista estándar y la minimizada será la vista que veamos en pequeñito a la derecha cuando alguna ventana esté ampliada.

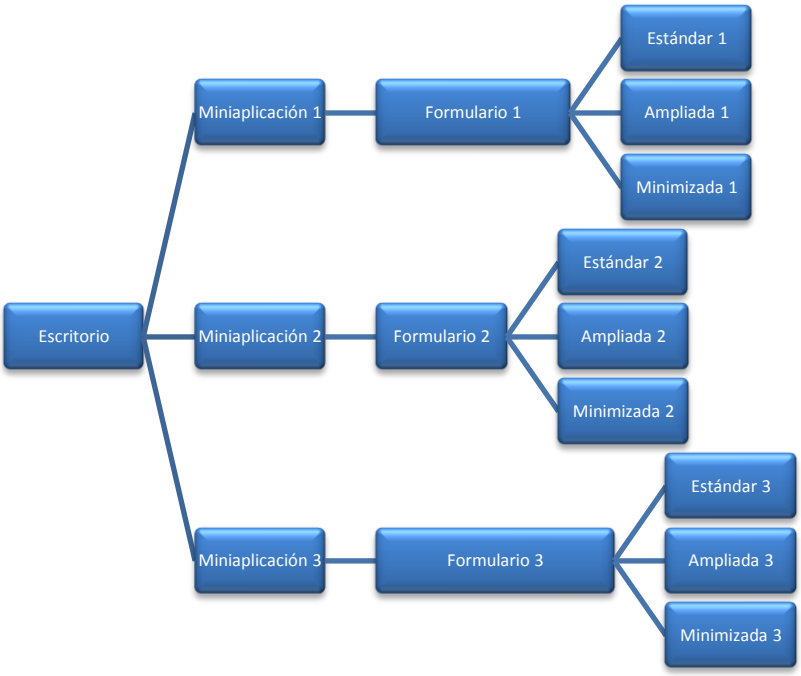

Figura 3: Esquema del modelo BoxApp

RF3. Configuración personal de la información al 'loguearse':

El usuario, al acceder a la aplicación con sus datos, seleccionará un profesional y un centro de trabajo como configuración activa. A partir de aquí, todos los datos e información del escritorio, dependerán de dicha configuración.

El usuario siempre tiene la posibilidad de cambiar de profesional y/o centro cuando le apetezca.

[2] Una vista hace referencia a la manera en que un formulario es visualizado.

RF4. Acceso a la información de sus pacientes, sus historiales clínicos y de asistencias, así como a su programación para la fecha indicada:

El usuario podrá acceder y ver la información de sus pacientes: datos generales, datos de afiliación sanitaria e incluso datos bancarios proporcionados por los pacientes. También podrá acceder a sus historiales clínicos y/o de asistencias. Además podrá disponer de la programación médica de dichos pacientes.

Para ello se crearán diversas ventanas donde se pueda ver esta información. El profesional dispondrá de:

- Datos del Paciente, donde poder ver toda la información personal, bancaria o de afiliación de sus pacientes.
- Historial de asistencias del Paciente, donde podrá ver todas las asistencias que ha tenido un paciente a lo largo de su vida en un centro determinado.

RF5. Mantenimiento de Antecedentes y alergias:

El usuario dispondrá de los antecedentes personales y familiares del paciente, sus hábitos o vicios e incluso su situación económica, así como una lista detallada de las alergias y información relacionada de sus pacientes, si tienen.

Podrá modificar, añadir o eliminar información de cada paciente, según le convenga y se ajuste a sus necesidades.

RF6. Mantenimiento de las órdenes médicas:

El usuario tendrá acceso a las órdenes médicas de sus pacientes: los fármacos que toma o ha tomado, la cantidad y dosis de dichos fármacos, el tiempo que lleva en tratamiento, etc.

Además podrá modificar y actualizar esta información.

RF7. Disponer de las Tareas Pendientes:

El usuario dispondrá de un apartado donde poder visualizar las tareas que aún tiene pendientes de realizar. Se dividirán en dos:

- Informes de alta: El usuario podrá ver los informes de alta de sus pacientes que aún están abiertos y/o por cerrar.
- Peticiones: También verá las peticiones de dichos pacientes.

RF8. Acceso al Censo del Hospital y a las Citaciones:

El usuario tendrá acceso al censo del Hospital donde trabaje y/o practique su actividad. En él podrá ver los pacientes ingresados según el tipo de asistencia y su urgencia y los pacientes que tiene en Lista de espera.

También tendrá acceso a las citaciones médicas de dichos pacientes.

RF9. Disponer de una Agenda personal:

El usuario dispondrá de una pequeña agenda privada y personal donde poder mirar todas sus tareas o actividades pendientes y/o realizadas, así como reuniones o eventos a los que debería asistir.

Además se le dará la posibilidad de añadir, modificar o eliminar información de esta agenda.

RF10. Acceso a sus Intervenciones:

Por último, dispondrá de un apartado sobre las Intervenciones médicas que tiene para el día o los días seleccionados. Puede ser el día en el que se encuentra o cualquier otra fecha que quiera consultar.

3.3. Requisitos no funcionales

A continuación se muestran los requisitos no funcionales que tendrá que cumplir nuestra aplicación:

RNF1. Cumplimiento de la LOPD:

La ley orgánica 15/1999 de protección de datos de carácter personal (LOPD) es una ley orgánica que tiene como objetivo garantizar y proteger los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente su intimidad y privacidad personal y familiar.

Esta ley afecta todos los datos que hacen referencia a personas físicas registradas sobre cualquier soporte, informático o no. Quedan exentos del cumplimiento de esta normativa aquellos datos recogidos para uso doméstico, las materias clasificadas del estado y aquellos ficheros que recogen datos sobre el terrorismo y otras formas de delincuencia organizada.

El trato de los datos personales que haga la aplicación deberá cumplir con esta ley.

En nuestra aplicación es importante el cumplimiento de esta ley ya que tratamos con información privada y muy personal sobre los pacientes y trabajadores de los centros sanitarios. Si no cumpliéramos esta ley, estariamos violando los derechos a la privacidad de toda la gente que esté relacionada directa o indirectamente con estos centros.

RNF3. Cumplimiento de la guía de estilo *Walnut*:

Es necesario que el diseño de la interfaz de usuario cumpla con los estándares de *Walnut*. Esto es necesario ya que la aplicación formará parte del producto *ekon 6*, el cual engloba la interfaz y estilo de codificación Walnut, y se está intentando que todo el producto siga una homogeneidad.

RNF4. Normalización de la base de datos y accesos según el estándar SQL 99: (ISO/IEC 9075:1999).

Nuestra base de datos deberá estar normalizada siguiendo los estándares correspondientes.

RNF5. El software ha de permitir ser ampliable y modificable:

La aplicación tiene que estar diseñada correctamente para permitir posibles modificaciones y ampliaciones en un futuro. La informática es un mundo de constante cambio y cualquier producto ha de ser sensible a futuras mejoras y/o ampliaciones.

RNF6. Tolerancia a errores y acciones incorrectas:

La aplicación tendrá que estar desarrollada de forma que tendrá que tener en cuenta que el usuario que la utilice puede cometer errores durante la entrada o modificación de datos, y realizar acciones de forma diferente a la prevista. Así pues deberá controlar las posibles entradas o acciones incorrectas permitiendo al usuario la corrección de las mismas e intentando minimizar la repetición de entrada de datos incorrectos.

4. Diseño de la aplicación

4.1. Introducción

En el momento de desarrollar una aplicación es importante que anteriormente se haya hecho un buen diseño que facilite su posterior codificación y la comprensión de su estructura y funcionamiento.

Inicialmente explicaremos la tecnología seleccionada a la hora de desarrollar la aplicación y sus características más importantes. Después se presentará un resumen del diseño de la aplicación y de su interfaz de usuario.

4.2. Tecnología de desarrollo

Tal como se ha comentado en el estudio de viabilidad, todos los productos de Unit4 *ekon*, y en concreto el que nos concierne *ekon Salus,* se desarrollan en la plataforma tecnológica *karat*. Esto implica que este producto está construido siguiendo las herramientas que nos proporciona la propia plataforma.

4.2.1. Java y Eclipse

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o la fuga de memoria, a través del colector de basura ("Garbage Collector").

Las principales características de este lenguaje son:

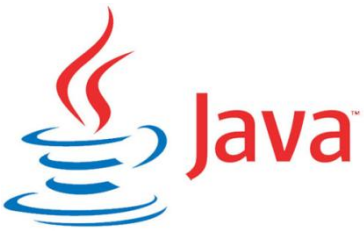

Figura 4: Logotipo de Java

1. Utiliza la metodología de la programación orientada a objetos. Los objetos son entidades que combinan estado (atributo), comportamiento (método) e identidad. Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos que pueden pertenecer a la misma o a diferente clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos.
2. Independencia de plataforma: permite la ejecución de un mismo programa en múltiples sistemas operativos.
3. Incluye por defecto soporte para trabajo en red.

4. Diseñado para ejecutar código en sistemas remotos de forma segura.
5. Relativamente fácil de usar y que coge lo mejor de otros lenguajes orientados a objetos, como C++.

Actualmente, la empresa Unit4 opera con la mayoría de sus productos, entre ellos *ekon Salus,* implementados o construidos en Visual Basic, un lenguaje de programación no orientado a objetos que se ha quedado obsoleto para realizar algunas actividades. Por ello, y por todas las facilidades que acabamos de explicar que nos ofrece Java, la empresa está migrando todos sus productos a Java. Cabe destacar que el escritorio del médico se realizará en este lenguaje, sin que el resto de la aplicación o producto lo esté, y por lo tanto de momento quedará un poco aislado. En un futuro, cuando todo el producto *Salus* esté migrado a Java, podrá enlazarse con nuestro escritorio y así utilizar de manera más óptima todas sus posibilidades.

El entorno de trabajo de Java que utilizaremos es Eclipse.

Eclipse es un entorno de desarrollo integrado de código abierto ("Open Source") multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido". Además, consta de infinidad de herramientas para hacer cómodo el desarrollo de aplicaciones y la gestión de proyectos.

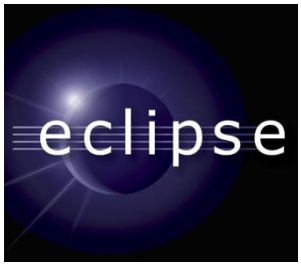

Figura 5: Logotipo de Eclipse

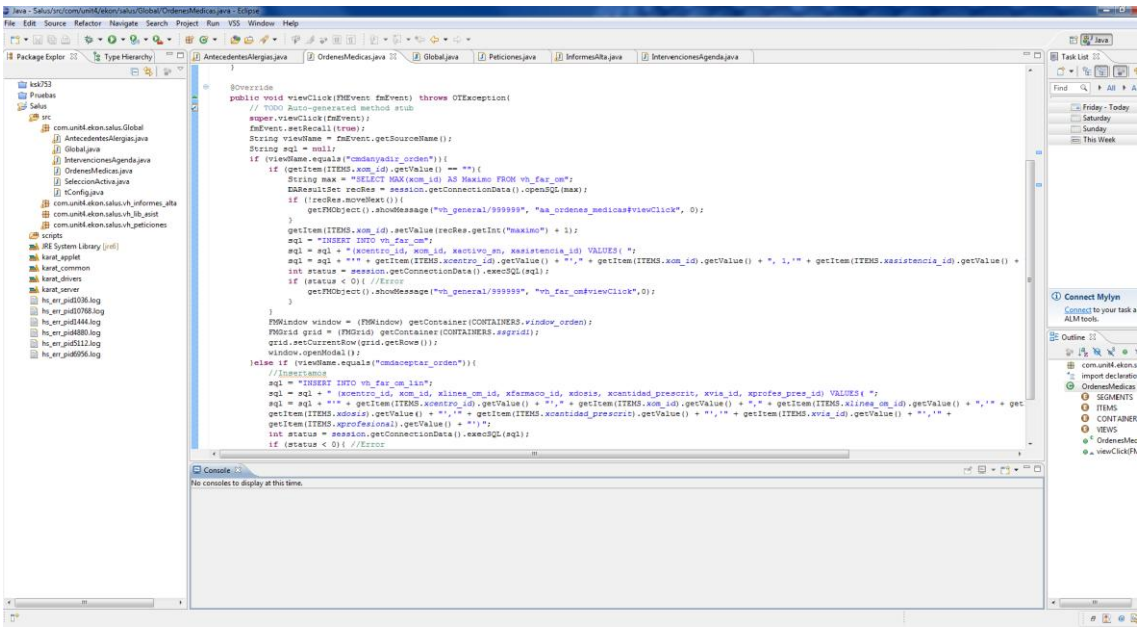

Figura 6: Ejemplo de entorno eclipse

4.2.2. Karat

Con su avanzada plataforma tecnológica *karat*, UNIT4 ofrece un entorno de gestión integral del ciclo de vida para las aplicaciones de gestión *UNIT4 ekon* (desarrollo e instalación, soporte, mantenimiento, evolución, personalización, etc.).

Su principal característica es que permite la ejecución del software desarrollado con ella en múltiples plataformas tecnológicas, sistemas operativos y bases de datos sin necesidad de modificar el código fuente.

Karat es la plataforma tecnológica idónea para el desarrollo, implantación, mantenimiento y evolución continua de aplicaciones. Está basado en un modelo de datos y en una forma de acceder a esos datos que a continuación describiremos.

Cada producto *ekon* dispone de una base de datos o repositorio propio. Este repositorio es un objeto de *karat* donde se almacenan el resto de objetos de *karat* que contienen la información y los datos necesarios de nuestra aplicación. A continuación explicaremos los más relevantes que utilizamos en este proyecto:

1. **Tablas:** Es donde están almacenados los datos físicamente. Con ayuda del resto de objetos *karat,* que explicaremos a continuación, contienen los datos que se mostrarán en la aplicación. Cada tabla está compuesta por un conjunto de campos, que hacen referencia al tipo de datos que hay almacenado, su importancia (si son campos primarios o no) y la relación que tienen con otros campos (mediante claves foráneas).
2. **Consulta Base y Consultas:** Son los objetos *karat* que nos permiten acceder a los datos físicos, agrupados en campos, de las tablas que componen el repositorio y mediante los cuales se puede modificar, borrar, mostrar y agregar datos en dichas tablas. Los accesos pueden afectar a una o más tablas a la vez y por supuesto a uno o más campos. Para esto se utiliza un lenguaje de consultas, como en nuestro caso SQL.
3. **Objeto de negocio:** Quizás el objeto *karat* que más relevancia tiene ya que permite gestionar como una entidad única varios niveles de información de la capa de datos. En él se agrupa toda la lógica de la entidad que implementa, así como sus reglas de validación, y nos permitirá hacer el mantenimiento de los datos asociados a dicha entidad.

   El objeto de negocio está formado por paneles y cada panel está asociado a una consulta. Los paneles se componen de controles, y cada uno de éstos pertenece a un campo de la tabla a la que la consulta asociada hace referencia. Hay un panel, llamado “HEADER” que es el más importante, el que, normalmente, contiene la información de más alto nivel y filtra al resto de paneles, a través de uno o más de sus campos.

   De esta manera, podemos decir que un Objeto de negocio es un agrupador de tablas y consultas a partir del cual gestionamos las diferentes entidades de nuestra aplicación. Cada entidad pertenece a un Objeto de negocio.
4. **Formulario:** Es el objeto de *karat* que implementa la representación visual de un objeto de negocio. En él definimos cómo se visualizará una pantalla de

nuestra aplicación. Cabe destacar que una aplicación estará formada por múltiples formularios, cada uno asociado única y exclusivamente a un Objeto de negocio.

Estos formularios se crean a partir de los controles definidos en el Objeto de negocio. Cada control puede ser de un tipo diferente, los más utilizados son: campos de texto, botones, listas despegables, checks[3] y grids[4]. De esta forma representamos de manera gráfica nuestros datos.

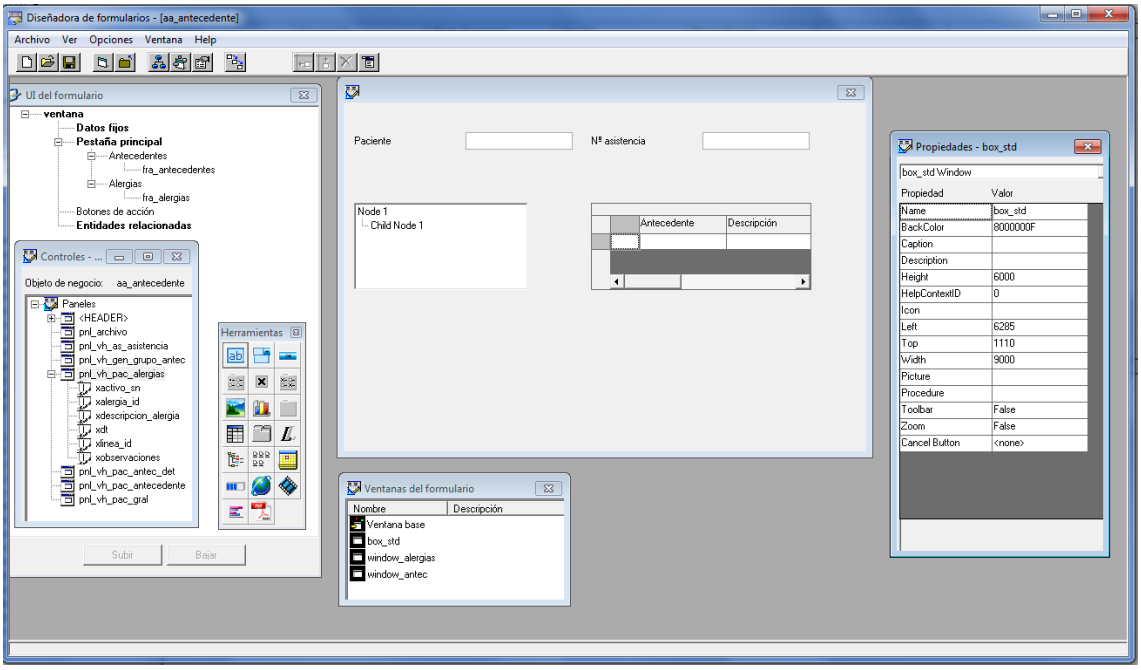

Figura 7: Diseñadora de formularios

5. **Variable de entorno:** es un objeto *karat* donde se guardan y almacenan datos sobre algún campo de una tabla, y al cual podrán acceder el resto de objetos de *karat*. También se denominan como variables globales, ya que se puede acceder a dichas variables desde cualquier objeto independientemente de la tabla o consulta a la que hagan referencia.

[3] Hacen referencia a los botones de selección o verificación que actúan como interruptores en los que según si están apretados o no transmiten una u otra información.
[4] Los grids son cuadrículas para representar datos en forma de tabla.

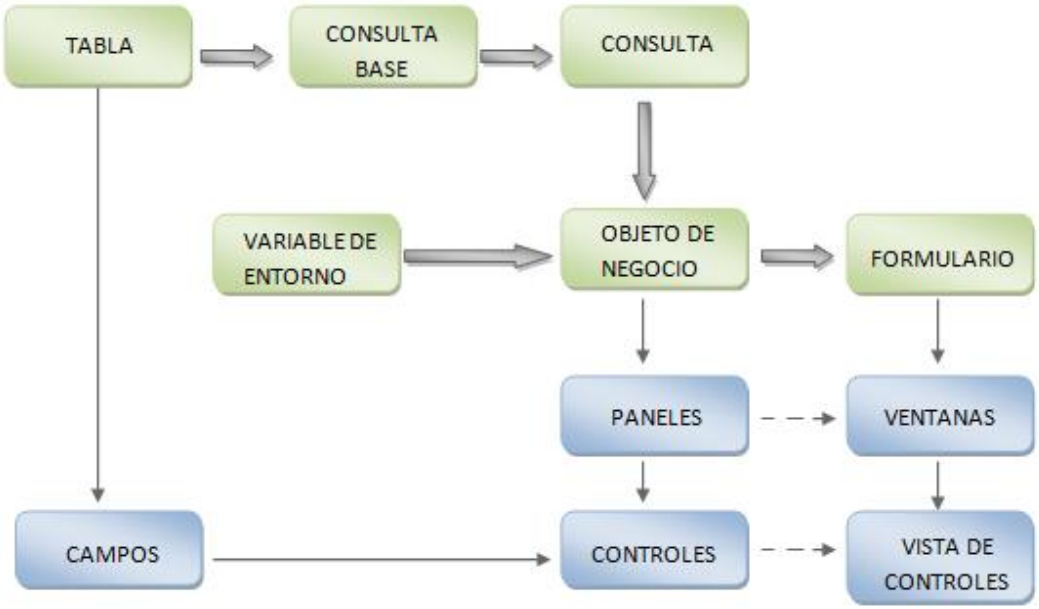

Figura 8: Esquema *Karat*

Estos objetos se definen en *karat* mediante una ayuda interactiva de asistentes o definidoras, donde se añaden los datos, sus valores, las relaciones entre ellos,… al repositorio. En la construcción de nuestro escritorio de trabajo del médico se han creado nuevos objetos *karat*, así como también se han utilizado y aprovechado objetos ya existentes en el repositorio. Más adelante, en la fase de Implementación serán explicados con detalle.

Además, estos objetos, también pueden ser creados y modificamos de forma manual a través de las clases de personalización de Java, que detallaremos a continuación. Esta clase de personalización debe utilizar las librerías de *karat* que son las que contienen las funciones específicas para poder modificarlos.

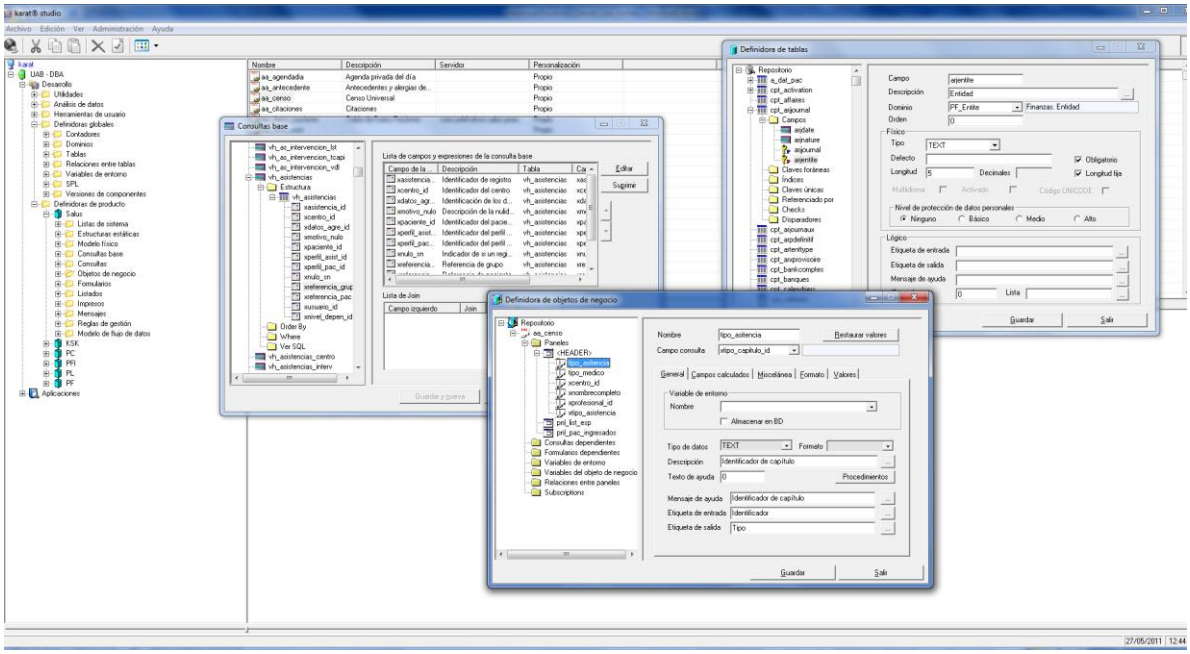

Figura 9: Diseñadora y asistentes de Objetos de negocio, Consultas y tablas.

4.2.3. Clases de personalización

Normalmente, existe una funcionalidad avanzada que no es posible realizar mediante la definición de objetos en el repositorio. Para poder implementarla, *karat* permite personalizar el funcionamiento de la aplicación mediante la generación de eventos programables.

El objetivo de los procedimientos de programación es permitir que los desarrolladores y/o implantadores puedan incluir cierta funcionalidad en las aplicaciones para adaptarlas a sus necesidades (o a las del cliente). Es posible personalizar cualquier objeto *karat* mediante la inclusión de procedimientos de programación en lenguaje *java*.

Existen diversas alternativas viables para la programación de objetos *karat* en función del tipo de objeto. La más utilizada, y la que hemos utilizado en este proyecto, son las clases de personalización a través de las cuales añadimos funcionalidad a cualquier tipo de objeto *karat*, mediante la inclusión de procedimientos de programación, de forma que se pueda personalizar el comportamiento de éste, extendiendo toda la funcionalidad proporcionada en tiempo de diseño.

En la definición de varios objetos *karat,* es posible definir un campo llamado "Servidor" en el cual podemos indicar la dupla librería-clase *java* que contiene el código de personalización del objeto. Las clases de personalización deben distribuirse en todas las máquinas que ejecuten la aplicación.

Cabe destacar que los procedimientos de programación no sólo permiten personalizar el comportamiento de determinado objeto *karat*, sino que además permiten:

- **Programación de subsistemas.** Desde cualquier procedimiento de programación es posible acceder a los diferentes subsistemas de *karat* (acceso a datos, ejecución de procedimientos almacenados, sesión, etc.) mediante la inclusión de un API específico para la programación de dichos subsistemas.
- **Enlazar con el resto de objetos *karat*.** En cualquier procedimiento de programación es posible enlazar con otros objetos propios de los definidos en el repositorio de *karat* (por ejemplo, ventana de diálogo de impresión de listados, visor de registros, etc.).

```
@Override
public void containerDblClick(FMEvent fmEvent) throws OTException {
    // TODO Auto-generated method stub
    super.containerDblClick(fmEvent);
    fmEvent.setRecall(true);
    String value = fmEvent.getSourceName();
    if (value.equals("TreeAntec")){
        FMWindow window = (FMWindow) getContainer(CONTAINERS.window_antec);
        FMGrid grid = (FMGrid) getContainer(CONTAINERS.ssgrid1);
        grid.setCurrentRow(grid.getRows());
        window.openModal();
    }
}
```

Figura 10: Ejemplo de clase de personalización

4.2.4. Base de Datos Oracle y lenguaje SQL

La base de datos por la que nos hemos decidido para almacenar nuestro repositorio ha sido Oracle ya que sigue el sistema de gestión de base de datos objeto-relacional, destacando entre sus características su estabilidad, escalabilidad y soporte multiplataforma. Para ello, nos ayudamos de la herramienta "Open Source" Aqua Data Studio.

Para acceder a nuestra base de datos a través de las clases de personalización utilizamos SQL, un lenguaje de consulta estructurado de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre ella.

Sus características principales son:

- Es un lenguaje de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjunto de registros, permite una alta productividad en la codificación.
- Asume las siguientes funcionalidades:
  - Lenguaje de definición de datos.
  - Lenguaje de definición de vistas.
  - Lenguaje de manipulación de datos.
  - Permite concesión y denegación de permisos.
  - Implementa restricciones de integridad y controles de transacción.
  -

4.3. Interfaz de usuario

4.3.1. Concepto y componentes básicos

Como se ha dicho con anterioridad en la fase de requerimientos, nuestro escritorio de trabajo del médico ha de basarse como premisa principal en el modelo BoxApp.

Para ello, dispondremos de la herramienta de creación de escritorios de *karat* que cumple esta funcionalidad. Esta herramienta permite al usuario añadir información a nuestro escritorio, el cual trataremos como entorno gráfico de la aplicación, a través de mini aplicaciones (o cajitas).

A continuación, presentamos más detalladamente los diferentes componentes y sus características:

4.3.1.1. Definidora de Mini aplicaciones

Como hemos dicho con anterioridad, a partir de los objetos *karat*, creamos formularios individuales sobre cada entidad de nuestra aplicación donde podremos visualizar la información que queremos. Estos formularios se insertan en mini aplicaciones o cajitas de nuestro escritorio.

Las mini aplicaciones son representaciones gráficas de nuestros formularios de pequeño tamaño consistentes en sí mismos y con capacidad de comunicarse con otras mini aplicaciones. Las definimos a través de su definidora (*ka_boxapp*). En ésta, se le asigna a cada mini aplicación un nombre, un formulario al que hace referencia, un tamaño y unos parámetros de entrada y salida que permiten enlazar la información de ésta con otras mini aplicaciones.

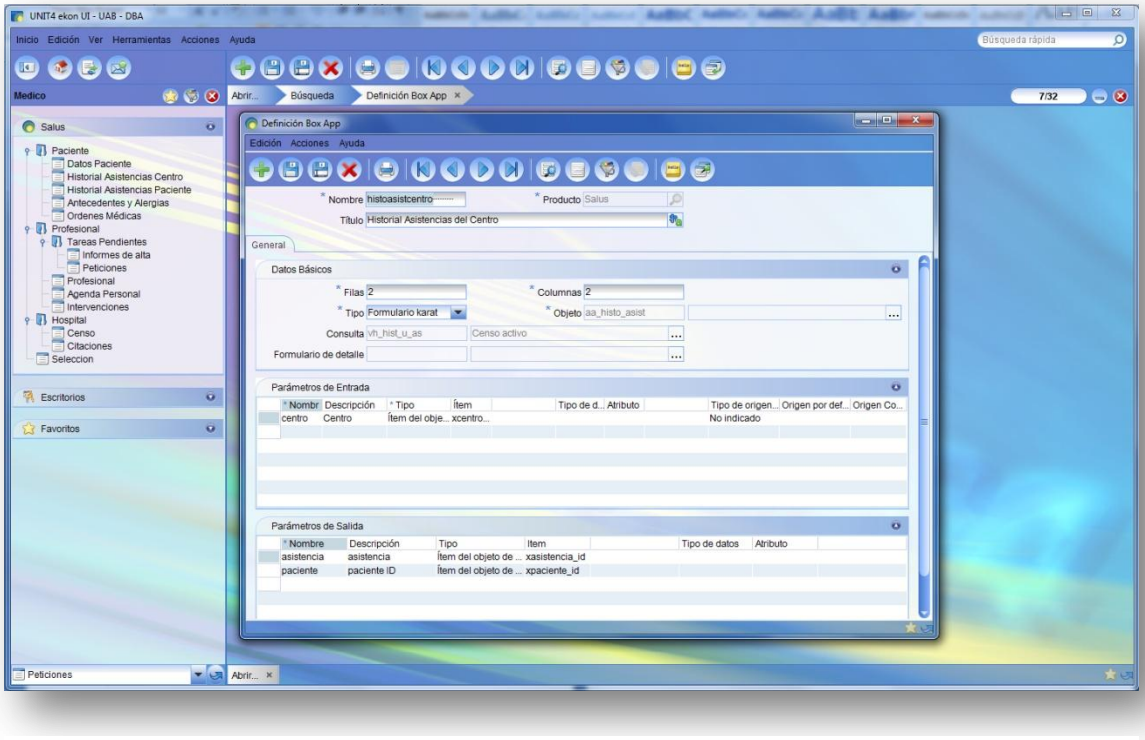

Figura 11: Definidora de miniaplicaciones (BoxApp)

Las mini aplicaciones pueden tener un color (normalmente tienen un mismo color aquellas mini aplicaciones que tienen relación entre sí) y medidas variables, y disponer de un refresco en segundos, para saber si alguna información relevante ha cambiado.

Las mini aplicaciones se distribuyen por el espacio del escritorio ocupando un número de celdas del mismo en función de una posición, altura y anchura.

Las cajas pueden ser de dos tipos:

- *BO Browser:* Muestran un objeto de negocio en modo lista (*BOData*). No necesitan diseño, basta con la definición realizada en la diseñadora de mini aplicaciones (*ka_boxapp*).
- *Formulario:* Muestran un formulario *karat* tal y como ha sido diseñado en la definidora de formularios.

Como bien hemos enumerado en diversas ocasiones, en el caso de cajas de tipo formulario (las que principalmente hemos utilizado en nuestra aplicación) es necesario diseñar previamente el formulario que se va a mostrar en la caja. En este punto indicaremos los puntos básicos a tener en cuenta en el desarrollo de este tipo de formularios.

En tiempo de ejecución, las cajas se pueden visualizar de tres formas distintas:

- Normal o *standard:* Visualización normal de la caja junto al resto de cajas del escritorio. La mini aplicación se muestra de forma compartida con el resto de mini aplicaciones repartiéndose la zona de trabajo en función de lo indicado por el usuario.
- Maximizada: La mini aplicación se muestra ocupando la mayor parte del escritorio de trabajo para facilitar el trabajo del usuario sobre esa mini aplicación, y con el resto de mini aplicaciones del escritorio minimizadas en la parte derecha del mismo.
- Minimizada: La mini aplicación se muestra ocupando una única celda en la parte derecha del escritorio de trabajo mientras otra de las mini aplicaciones se muestra en forma maximizada.

En la mayoría de casos parece conveniente tener tres visualizaciones diferentes del mismo formulario en función de la vista que está activa en cada momento. Para poder diseñar un formulario que soporte diferentes vistas del mismo se deben seguir las siguientes reglas:

- La pantalla completa de nuestro formulario será mostrada en el modo maximizado. Pertenece a la ventana base.
- En el caso de existir una subventana de nombre **box_std**, ésta será la mostrada en la visualización estándar.
- En el caso de existir una subventana de nombre **box_min**, ésta será la mostrada en la visualización minimizada. Esta ventana se deberá diseñar considerando que sólo será visible la primera fila.

Una cuestión a considerar es que el tamaño de las diferentes vistas no es fijo, sino que dependerá de la resolución de pantalla, de si la aplicación está maximizada o

no, etc...

4.3.1.2. Definidora de Escritorios

Los escritorios *karat* son contenedores de mini aplicaciones o cajitas de aplicación configurables por parte del usuario. En estos escritorios se pueden añadir o quitar mini aplicaciones y mover sus posiciones, a través del "Drag & Drop", también conocido como "Arrastrar y Soltar", donde una mini aplicación es arrastrada con el ratón y soltada en la posición requerida. Si en esa posición hay otra mini aplicación se intercambian. Además, estos escritorios incluyen una comunicación entre las diferentes mini aplicaciones que lo componen, a través de unos parámetros de entrada y salida. La salida de una mini aplicación puede ser la entrada de otras.

Los escritorios se dividen en celdas donde se añaden las mini aplicaciones (cada aplicación ocupa un número de celdas determinadas). Un escritorio estándar es de 12x12 celdas, aunque su tamaño exacto depende del tamaño de la pantalla y de su resolución.

Estos escritorios pueden ser modificados y configurados por el usuario para adaptarse a sus propias necesidades, e incluso podrá, si lo considera necesario, crear sus propios escritorios personales. También pueden incluir una gestión de calendario opcional. Se permitirá indicar que un escritorio gestione la fecha actual (Escritorio Calendario), en cuyo caso aparece un calendario en la parte superior que permite modificar la fecha e informar del cambio de fecha a las cajas que lo componen.

Los desarrolladores pueden distribuir plantillas de escritorios con sus respectivas mini aplicaciones, las cuales pueden ser utilizadas y personalizadas inmediatamente por los usuarios del sistema; podrán eliminar mini aplicaciones, recolocarlas o cambiar sus parámetros. De todas maneras, los usuarios finales pueden crear sus propios escritorios adicionales, los cuales se pueden acceder desde el panel asociado del árbol de navegación.

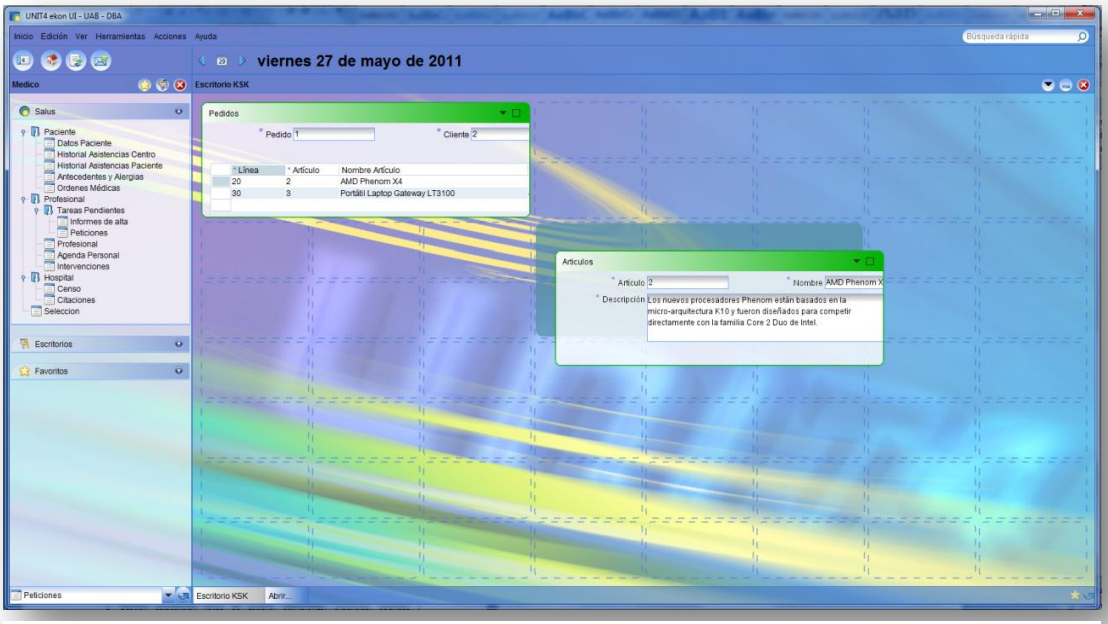

Figura 12: Ejemplo de un *"Drag & Drop"* en un escritorio dividido en celdas.

5. Fase de Implementación

5.1. Introducción

En este apartado, se explicará con detalle cómo se ha realizado nuestro escritorio. En primer lugar, resaltar, como se ha dicho con anterioridad, que para crear nuestras mini aplicaciones o ventanas en el escritorio primero se ha tenido que realizar uno a uno nuestros formularios a partir del resto de objetos de *karat*.

A continuación, se explicarán estos formularios, así como su visualización y funcionalidad final. También comentaremos los aspectos más importantes de su implementación. Por último expondremos en el apartado de "Relaciones entre ventanas", qué ventanas están relacionadas con otras y de cuales reciben la información que necesitan.

Antes de ello, cabe destacar un par de aspectos importantes del escritorio.

En primer lugar recalcar que el escritorio del médico es, por definirlo de alguna manera, una puerta de acceso al resto de módulos de *ekon Salus*. Por lo tanto, está pensado para ser únicamente una herramienta de lectura, para que el profesional vea la información más relevante de su trabajo, pero sin posibilidad de cambiar o modificar dicha información, ya que de estos mantenimientos, se encargan el resto de módulos de *ekon Salus*. Sólo tres ventanas permiten añadir o modificar información: Antecedentes y alergias, Órdenes Médicas y Agenda Personal.

En segundo lugar, es que en nuestro escritorio habrá cierta información global que utilizarán o necesitarán las diferentes ventanas o formularios. Esta información servirá como filtros entre ventanas y se hará a través de las variables de entorno explicadas con anterioridad. El formulario de Selección Activa cobra importancia en este aspecto.

A continuación, podemos ver el árbol de nuestro escritorio, donde se encuentran todos los formularios realizados.

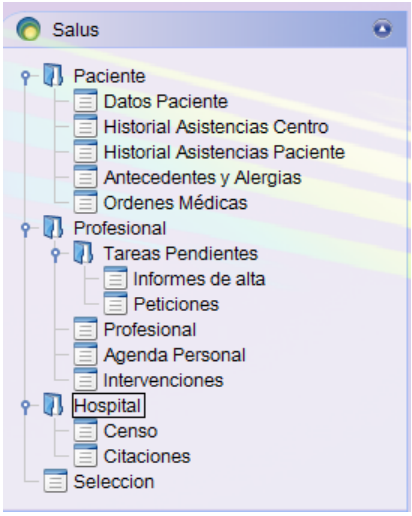

Figura 13: Árbol de formularios en nuestro escritorio

5.2. Formulario de Selección Activa

En primer lugar, cabe decir que como la aplicación de *ekon Salus* no está migrada todavía a Java, no disponemos de los módulos de configuración, a partir de los cuales, un profesional accede a la aplicación mediante un usuario y contraseña y se carga toda la información relativa a este profesional. De momento, en nuestro escritorio sólo se dispone del login inicial de *karat.*

Por ello hemos realizado este formulario de Selección Activa. En éste, el usuario, escoge el profesional que quiere (normalmente el que le corresponda) y el centro en el que realiza sus actividades, a través de una MDQO[5] de dos campos de texto. Esta información se guarda en sus respectivas variables de entorno comunes a todas las ventanas. A continuación podemos abrir el escritorio. Todas las ventanas de éste se filtrarán a través del profesional y el centro escogidos en este formulario. Cuando queramos cambiar esta información sobre las ventanas de nuestro escritorio solo tenemos que abrir este formulario y cambiar los campos.

En la siguiente figura, se observa el formulario de selección activa. Al hacer click[6] en las lupas de los campos de texto, se abren las respectivas listas de profesionales y centros en las que podemos hacer la elección.

Tenemos también un botón que se encarga de cerrar este formulario. De esta manera sabemos que nuestra información ha sido guardada.

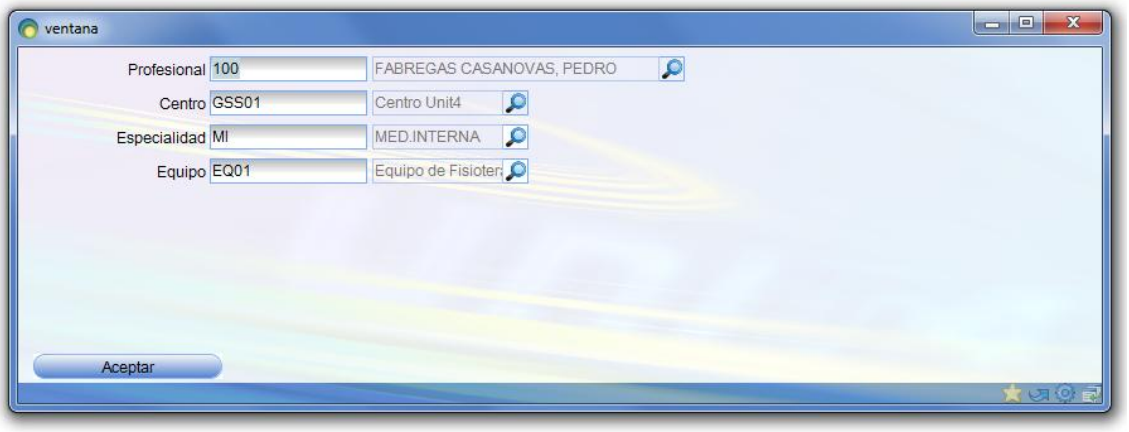

Figura 14: Formulario de Selección Activa

[5] Es una consulta dependiente a un campo y que muestra todos sus registros, para elección del que más nos guste

[6] Hacer click, hace referencia a la acción de de apretar el botón derecho del ratón de nuestro ordenador.

5.3. Ventanas relacionadas con los pacientes

A continuación se explican aquellas ventanas que tienen que ver con los pacientes y sus datos, así como enfermedades o antecedentes que pueden tener y posibles medicaciones que estén tomando.

5.3.1. Datos Paciente

En esta ventana el profesional puede observar los datos de sus pacientes. Seleccionando una id[7] (o expediente) de un paciente se muestran sus datos.

A continuación se explica cómo se ha implementado esta ventana:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_datos_paciente*", está compuesto por diversos paneles. El de más relevancia, es panel HEADER, que hace referencia a la tabla "vh_pac_gral", a través de la consulta con el mismo nombre, en la que están almacenados todos los datos e información de los pacientes. A partir de esta tabla conseguimos la id del paciente (también llamado expediente) que utilizamos para filtrar en el resto de paneles y conseguir sus datos personales (*vh_pac_direcciones* y *vh_pac_archivo*), sus datos de afiliación a la seguridad social (*vh_pac_cober*) y sus datos bancarios (*vh_pac_cob*).
- **Formulario:** el formulario, que es como veremos nuestra ventana, estará formado de la siguiente manera:
  - Cabecera: formada por la id del paciente del campo HEADER, también denominado expediente.
  - Pestaña de Datos Generales: se ven los datos más relevantes del paciente, a través de diversos campos de texto: nombre y apellidos del paciente, domicilio, fecha de nacimiento, estado civil, etc. Esta pestaña corresponde con los paneles HEADER y de direcciones.
  - Pestaña de Datos bancarios: se observan los datos bancarios de nuestros pacientes, siempre respetando la LOPD, como: los códigos y nombres de la entidad bancaria y la oficina, el número de cuenta o la forma de pago.
  - Pestaña de Datos de afiliación: donde vemos los datos de afiliación de los pacientes a la seguridad social, así como la entidad a la que están afiliados, el número de tarjeta sanitaria, etc.
- **Clase de personalización:** La clase de personalización de esta ventana se denomina "*DatosPaciente"*. En ella únicamente realizamos la funcionalidad de recibir el parámetro de entrada de la id paciente.

  Para ello, es necesario destacar que en el escritorio, no deberemos indicar la id del paciente para ver sus datos (aunque también tenemos esta posibilidad) sino que recibiremos esta información a través de otras ventanas, la del Censo del

[7] Es un identificador numérico.

Hospital o la de Citaciones. Para ello, esta ventana tiene dos parámetros de entrada que corresponden con:

- El primero de ellos, "*pacienteid*", proviene de un Ítem del objeto de negocio de Censo Hospital. Al seleccionar cualquier paciente del Censo, se le envía su id a Datos Paciente.
- El segundo, "*pacienteid2*", es un parámetro libre que proviene de Citaciones a través de una clase de personalización de java. Mediante ésta, al seleccionar un paciente de Citaciones, se le pasa a Datos Paciente la id de dicho paciente.

Cabe destacar, que el parámetro recibido es sólo uno, el último que llega. Por lo tanto, si hacemos click en un paciente del Censo, veremos en este formulario sus datos, y si después cambiamos de paciente en Citaciones también se cambia esta información. El método que realiza esta función se denomina "*deskBoxParameter*" y detecta cual es el parámetro que está recibiendo y de donde procede.

A continuación, se puede ver cómo quedan y cómo se visualizan las vistas estándar y maximizada de esta ventana. En la estándar solo vemos la id y el nombre del paciente, así como su número de historia clínica y tarjeta sanitaria, mientras que en la maximizada es donde podremos acceder a toda la información de éste paciente.

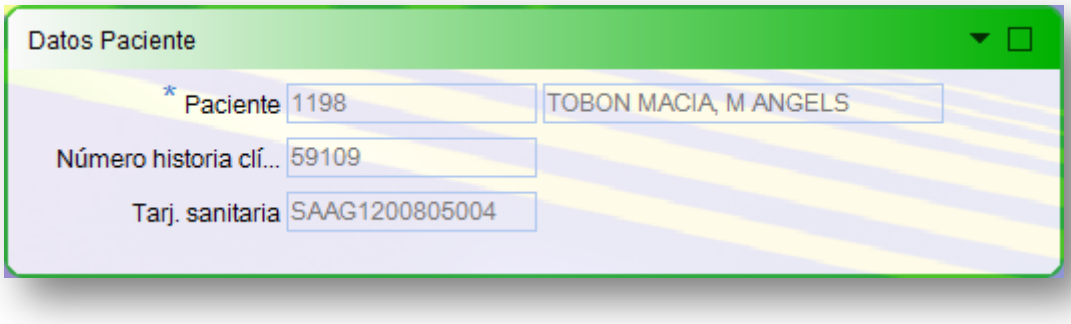

Figura 15: Vista estándar de la ventana de Datos del Paciente

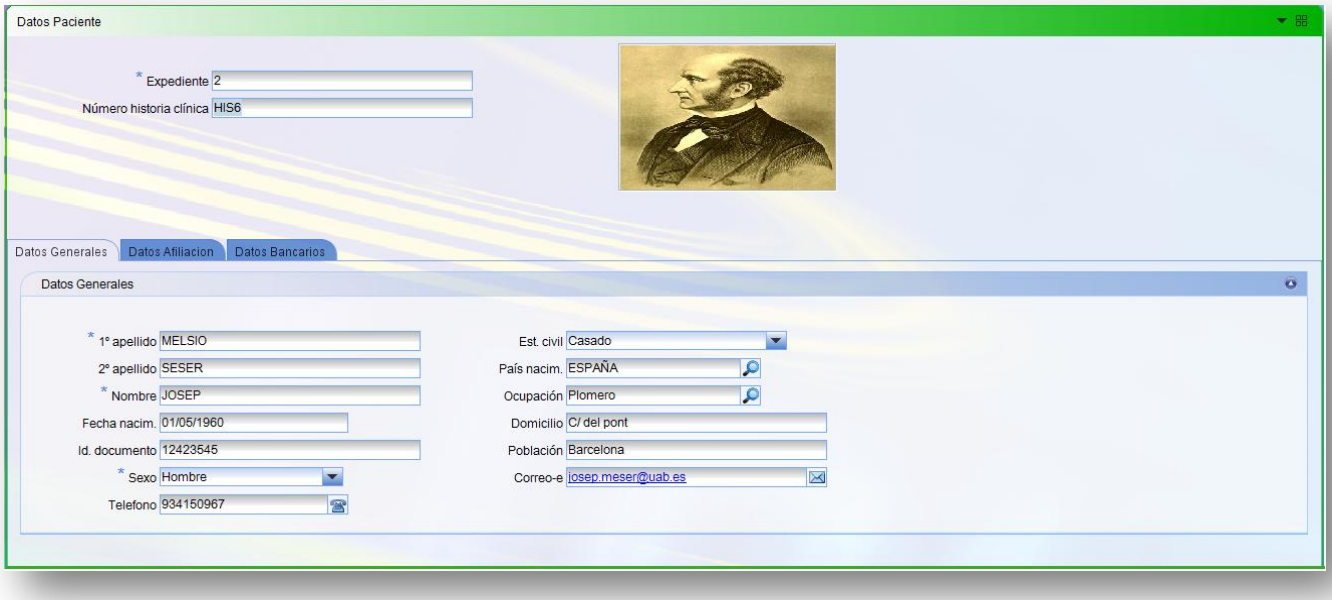

Figura 16: Ventana de Datos del Paciente

5.3.2. Historial de asistencias del Paciente

Esta ventana, como su nombre indica, muestra las asistencias de los pacientes a partir dos parámetros de entrada que recibe de Censo y Citaciones, del mismo modo que hemos explicado en la ventana anterior. Si en cualquiera de estos dos seleccionamos un paciente, inmediatamente pasamos su id al Historial de asistencias.

A continuación se explica cómo se ha implementado esta ventana:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_histo_asist_pac*", está compuesto por dos paneles:
  - Panel HEADER: se obtiene la identificación del paciente (id) y sus datos más relevantes. Se utiliza para filtrar el panel de asistencias.
  - Panel de historial de asistencias: hace referencia a la tabla "*vh_hist_u_as*" de la que obtenemos las diferentes asistencias de un paciente y sus detalles. Viene filtrado por la id del paciente del HEADER.
- **Formulario:** estará formado por:
  - Cabecera: formado por la id del paciente del campo HEADER.
  - Pestaña de Historial del Paciente: vemos todas las asistencias del paciente y los datos más relevantes de éstas. Esta pestaña corresponde con el panel de historial de asistencias.

Nos muestra todas las asistencias que ha tenido este paciente en el hospital, así como una pequeña descripción de cada una. En la figura 16 podemos ver cómo queda la ventana completa:

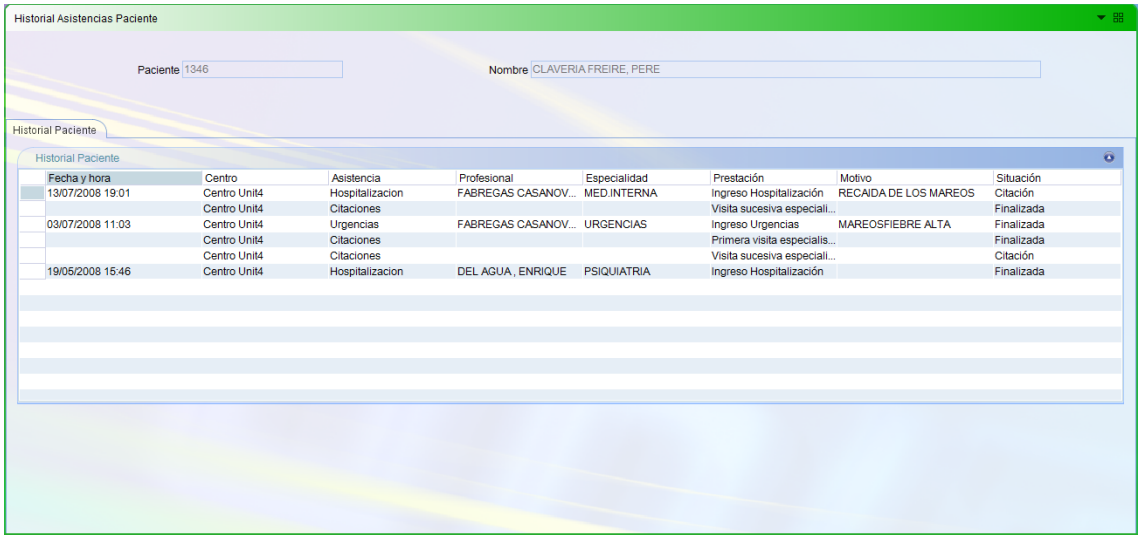

Figura 17: Ventana de Historial de Asistencias

La vista enstándar de esta ventana es muy parecida, al ser una ventana simple de construcción. La podemos ver en la figura 17:

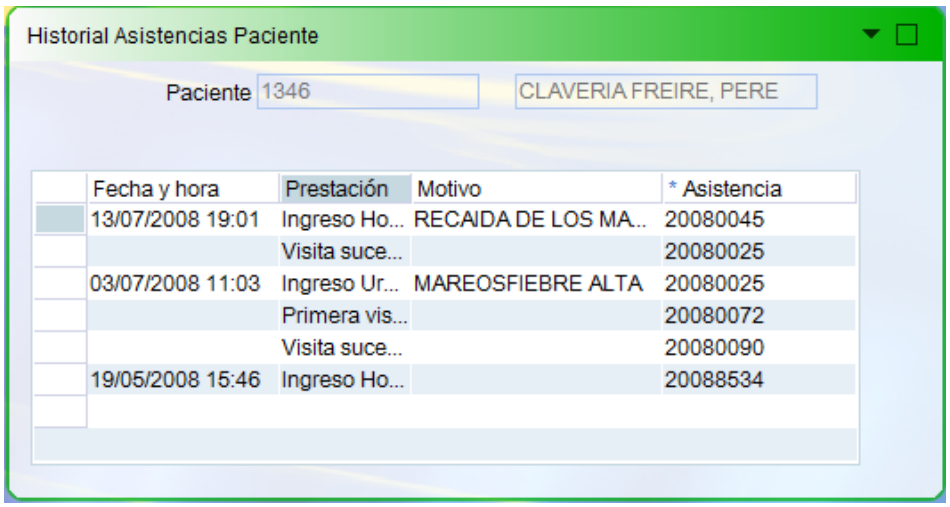

Figura 18: Vista estándar de Historial de Asistencias del paciente

5.3.3. Antecedentes y alergias

Posiblemente esta sea una de las ventanas más completas de nuestro escritorio. A diferencia de la gran mayoría de ventanas, ésta no es simplemente una ventana de lectura, que muestra datos pero no permite su modificación (premisa principal del escritorio), sino que además permite añadir o modificar información.

En esta ventana el profesional puede ver los antecedentes de sus pacientes. Estos antecedentes consisten en un resumen global de la salud general del paciente hasta la fecha, incluyendo las lesiones antiguas, alergias, intervenciones quirúrgicas,

inmunizaciones así como la historia psiquiátrica, hábitos o incluso, enfermedades familiares que puedan haber sido heredadas.

Para ello se necesita saber qué paciente estamos visualizando. Esta ventana recibe como parámetros de entrada la id del paciente y una asistencia, esta última, simplemente informativa. Como en las anteriores ventanas, estos parámetros vienen del Censo del Hospital como Ítem del objeto de negocio o de las Citaciones como parámetro libre, según en qué lista seleccionemos el paciente. A partir de estos parámetros filtramos el resto de información. En la clase de personalización explicaremos cómo los recibe.

A continuación explicaremos como se ha implementado esta ventana:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_antecedentes*", está compuesto por diversos paneles, los más importantes son:
  - Panel HEADER: tenemos los campos de la id del paciente y la id de la asistencia que nos servirán como filtro para el resto de paneles.
  - Panel de grupos de antecedentes: hace referencia a la tabla "*vh_gen_grupo_antec*" y de ella obtenemos los diferentes grupos de antecedentes que puede tener un paciente.
  - Panel de antecedentes: este panel hace referencia a la tabla *"vh_pac_antec_det"* de la que obtenemos el tipo de antecedente y su descripción. Está filtrado por la id del paciente del HEADER y el grupo de antecedente al que pertenece.
  - Panel de alergias: hace referencia a la tabla *"vh_pac_alergias"* de la que obtenemos las alergias del paciente y sus datos más relevantes. Está filtrado por la id del paciente del HEADER.
  - Paneles auxiliares: son los paneles que se utilizan para obtener datos relevantes relacionados con el paciente y la asistencia que se están tratando. Acceden a la tabla de asistencias (*vh_asistencias*) y a la de datos del paciente (*vh_pac_gral*)
- **Formulario:** el formulario, que es como veremos nuestra ventana, estará formado por:
  - Cabecera: estará formada por los campos del HEADER, paciente y asistencia, así como de los datos obtenidos de los paneles auxiliares: nombre y apellidos del paciente, número de historial clínico, centro en el que se encuentra y episodio de la asistencia.
  - Pestaña de Antecedentes: esta pestaña estará formada por:
    - Árbol de Grupos de Antecedentes: disponemos de un campo *"FMTreeView"* done mostraremos en un árbol los diferentes grupos de antecedentes. La creación de este árbol se realiza mediante su clase de personalización de java.
    - Grid y cuadro de texto de Antecedentes: a la derecha del árbol disponemos de un grid donde se muestran los diferentes antecedentes del paciente, y un

campo de texto donde se muestra la descripción de los antecedentes más detalladamente. Señalar que para que se muestren datos en estos campos, primero se ha de seleccionar un grupo de antecedente del árbol, como se explicará en la clase de personalización. Hace referencia al panel de antecedentes del objeto de negocio.

- Botón de actualizar: se dispone de un botón "*Actualizar"* para grabar en su respectiva tabla (*vh_pac_antec_det)* los datos introducidos en el campo de texto.

- Pestaña de Alergias:

  - Grid de Alergias: se dispone de un grid donde se muestran las diferentes alergias que sufre o ha sufrido el paciente, así como los datos más relevantes de éstas (la fecha de detección de la alergia, si aún la padece o no y posibles observaciones). Hace referencia al panel de alergias.

  - Botón de añadir: se dispone de un botón *"Añadir"* donde le damos al profesional la posibilidad de añadir una nueva alergia al paciente. Al pulsar este botón se abre una ventana modal que permite añadir dicha alergia. Estos datos se añaden mediante una sentencia SQL a la tabla *vh_pac_alergias* e inmediatamente podemos observarlos en nuestro grid.

- **Clase de personalización:** La clase de personalización de esta ventana se denomina "*AntecedentesAlergias"*. En ella se realiza la funcionalidad de más relevancia de esta ventana, que se explica a continuación. Empezaremos por la pestaña de antecedentes. En primer lugar hemos de explicar cómo se forma nuestro árbol de grupos de antecedentes. Para ello disponemos de las funciones o métodos "*Antecedentes*" y "*loadAntecedentes*". En la primera de ellas creamos la consulta necesaria mediante SQL para obtener los diferentes grupos y su descripción de la tabla "*vh_gen_antec_det*". En la segunda función, tratamos la sentencia SQL anterior y creamos el árbol asignándolo al campo del formulario denominado *FMTreeView.* De esta manera, ya podemos ver en nuestro árbol los diferentes grupos de antecedentes.

  Una vez creado el árbol, al seleccionar un grupo de este árbol, a la derecha se muestran los antecedentes del paciente (que corresponden a ese grupo), tanto en el grid como en el campo de texto explicados con antelación. La selección del grupo informe se hace mediante la función "*containerNodeClick"* que nos detecta qué nodo (o grupo) hemos seleccionado y nos devuelve un identificador que corresponde con el grupo. De esta manera, con este grupo y el paciente que tenemos en la cabecera, a partir del objeto de negocio, mostramos esta información en la pestaña.

  Otra funcionalidad de este árbol es la de poder añadir nuevos antecedentes al grid. Para ello hemos de hacer doble click sobre uno de los nodos del árbol. Mediante la función *"containerDblClick"* se abre una ventana mo[8]dal donde podemos añadir un nuevo antecedente, la fecha de detección y posibles

[8] Una ventana modal, es una ventana "hija" que no permite ninguna acción en la ventana "madre", la que le ha dado origen, hasta que no se haya tomado una decisión sobre ella misma.

observaciones. A partir de otra función, denominada *"viewClick"*, detectamos si hemos pulsado el botón "*Aceptar"* lo que permite guardar, mediante sentencia SQL, la alergia en la tabla correspondiente, e inmediatamente visualizarlo en el grid. También se dispone de otro botón denominado "*Actualizar"* que guarda de la misma manera que el anterior los datos que hayamos introducido en el campo de texto de antecedentes.

Por último, cabe decir que los parámetros de entrada se reciben del mismo modo que en los datos del paciente, mediante la función "*deskBoxParameter*" que detecta cual es el parámetro que está recibiendo y de donde.

A continuación podemos observar la ventana de antecedentes y alergias con la pestaña de antecedentes activa, donde visualizamos el árbol, grid y campo de texto mencionados antes, y la ventana modal para añadir un antecedente al grid:

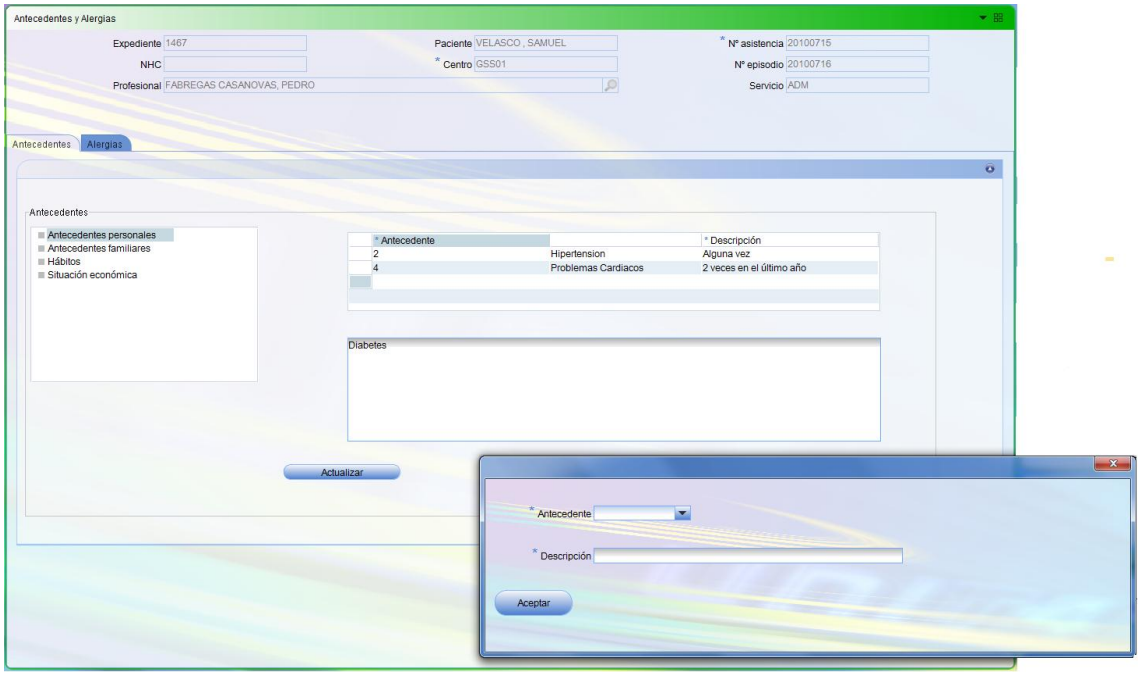

Figura 19: Ventana de Antecedentes y Alergias

Pasamos ahora a la pestaña de alergias. Aquí el profesional también tiene la posibilidad de añadir una nueva alergia al paciente si así lo requiere. Haciendo un click sobre el botón "Añadir" se abre una ventana modal donde podemos añadir una alergia, completando los mismos campos que en el grid anterior.

Esta funcionalidad se le permite a través de la función "*viewClick*". Caundo se pulsa el botón se abre dicha ventana modal y nos permite añadir, a través de una sentencia SQL, nuestros datos sobre la nueva alergia a la tabla que le corresponde (vh_pac_alergias), para luego poder visualizarla en la ventana. Si la alergia no existe la añadimos y si ya existe la actualizamos con los nuevos datos introducidos, ya que un mismo paciente no puede tener dos alergias iguales.

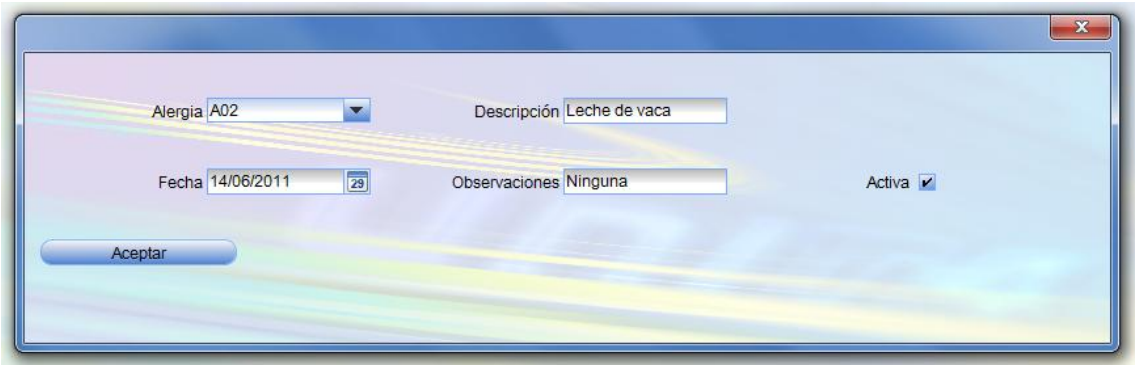

Figura 20: Ventana modal de inserción de alergias

También disponemos de una vista estándar de esta ventana que veremos nada más abrir el escritorio. En esta sólo vemos el paciente y la asistencia que estamos tratando y sus antecedentes en un grid. No se puede añadir ni modificar información. Para ello deberemos abrir la vista ampliada o maximizada.

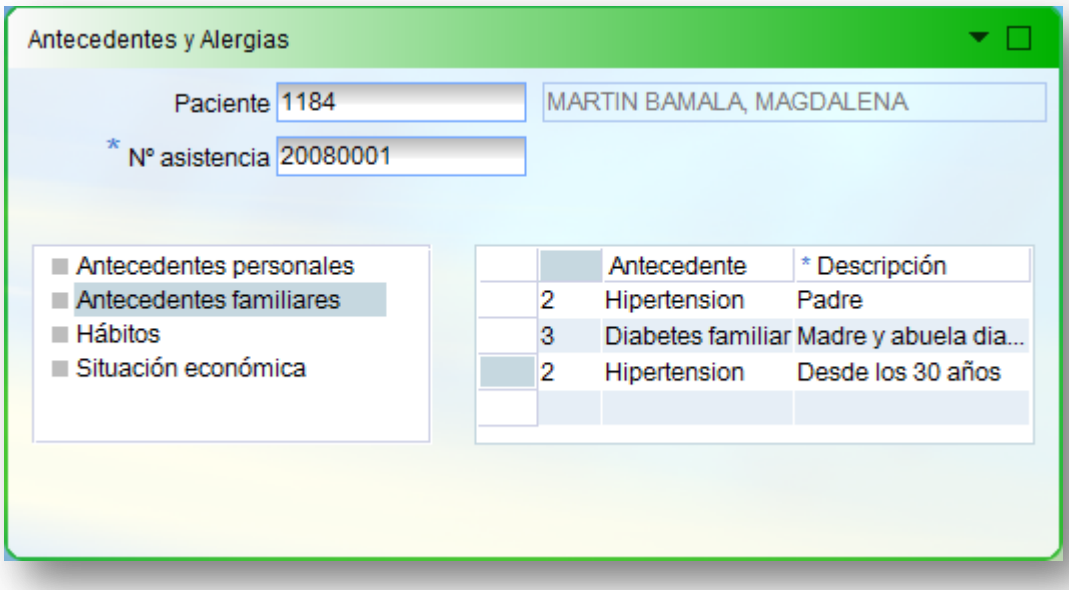

Figura 21: Vista estándar de Antecedentes y Alergias

5.3.4. Órdenes Médicas

Esta ventana es, junto a la de Antecedentes y Alergias y a la de Agenda del Profesional, las únicas ventanas que permiten modificar o añadir información.

Es más fácil y sencilla de tratar que la anterior, puesto que sólo está compuesta por una pestaña. En ella visualizamos todas las órdenes médicas que el paciente ha recibido a lo largo de su vida. Estas órdenes suelen ser fármacos que un profesional le ha hecho tomar al paciente como tratamiento de alguna enfermedad o síntoma.

En primer lugar, cabe decir que esta información depende de una asistencia. Cada asistencia pertenece a un paciente, aunque éste puede tener diversas asistencias, y sólo puede tener una orden médica. Esta asistencia se recibe en la ventana como parámetro de entrada de Ítem de objeto de negocio de Citación o como parámetro Libre de Censo de Hospital, según cual seleccionemos en cada momento. Esto se controla a través de la clase de personalización.

A continuación explicaremos como se ha implementado esta ventana:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_ordenes*", está compuesto por diversos paneles. Los más importantes son:
  - Panel HEADER: tenemos los campos de la id del paciente y la id de la asistencia, que recibimos como parámetros de entrada, y que nos servirán como filtro para el resto de paneles.
  - Panel de Órdenes médicas: hace referencia a la tabla "*vh_far_om*" y de ella obtenemos las id's de las órdenes médicas y el profesional del que dependen, según la asistencia que se le pase desde el panel HEADER.
  - Panel de líneas de órdenes médicas: hace referencia a la tabla *"vh_far_om_lin"* de la que obtenemos todos los detalles de las órdenes médicas, como el fármaco a tomar, la dosis, la frecuencia y las fechas de inicio y fin del tratamiento. Está filtrado por la id de la orden médica que recibe del panel anterior.
- **Formulario:** estará formado de la siguiente manera:
  - Cabecera: estará formada por los campos del HEADER, paciente y asistencia, así como de las variables de entorno profesional y centro.
  - Pestaña de Órdenes Médicas: esta pestaña está compuesta por:
    - Campo de texto: tiene un campo de texto donde se muestra informada la id de la orden médica que tiene la asistencia del paciente con la que se está trabajando.
    - Grid de Órdenes Médicas: donde se muestran las diferentes líneas de órdenes médicas en las que está o ha estado en tratamiento el paciente. Hace referencia al panel de líneas de órdenes médicas.
    - Botón de añadir: con el que damos al profesional la posibilidad de añadir una nueva línea de órden médica a la asistencia del paciente. Al pulsar este botón se abre una ventana modal que permite añadir dicha línea. Estos datos se añaden mediante sentencia SQL a la tabla "*vh_pac_alergias"* e inmediatamente podemos observarlos en nuestro grid. A continuación podemos ver esta ventana modal que será explicada en la clase de personalización.

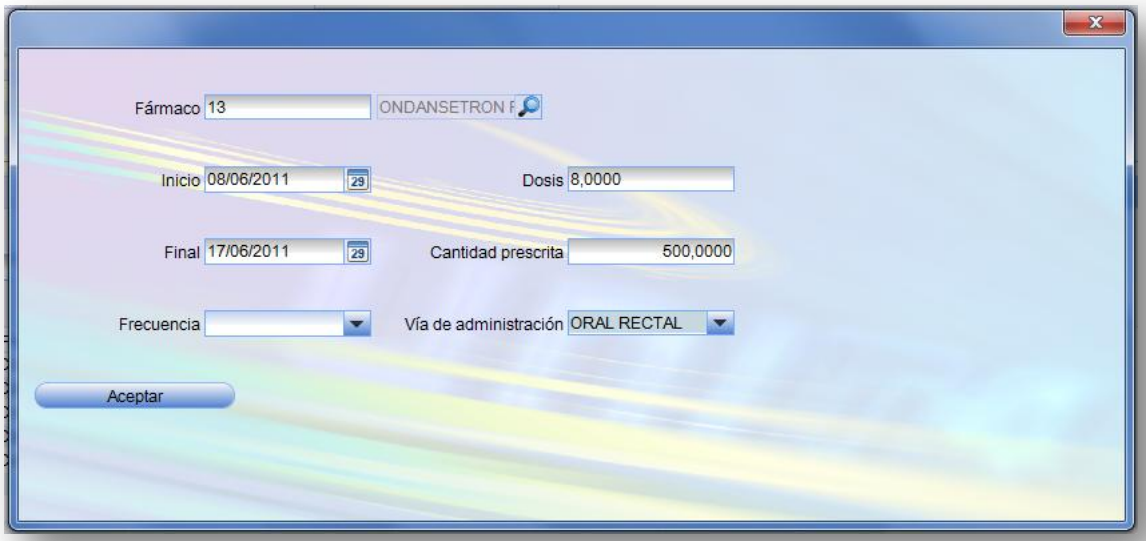

Figura 22: Ventana Modal de inserción de órdenes médicas

- **Clase de personalización:** La clase de personalización de esta ventana se denomina "*OrdenesMedicas"*. La única funcionalidad que implementa esta clase es la de añadir una nueva línea de orden médica.

Para ello deberemos hacer click sobre el botón "Añadir" y mediante el método o función "*viewClick"* se nos abrirá una ventana modal donde introduciremos los datos requeridos y que mediante las sentencias SQL añadiremos a las tablas correspondientes para luego poder visualizarlos. Es importante decir que si la asistencia de algún paciente no tiene ninguna orden médica y aún así deseamos introducir una receta o fármaco a éste paciente, primero se creará dicha orden, buscando la id de la última creada y sumándole uno, para luego poder introducir los datos.

De la misma manera que en Antecedentes recibimos los parámetros de entrada explicados al principio, mediante la función "*deskBoxParameter*" que detecta cual es el parámetro que está recibiendo y de donde.

A continuación podemos ver la ventana de órdenes médicas completa:

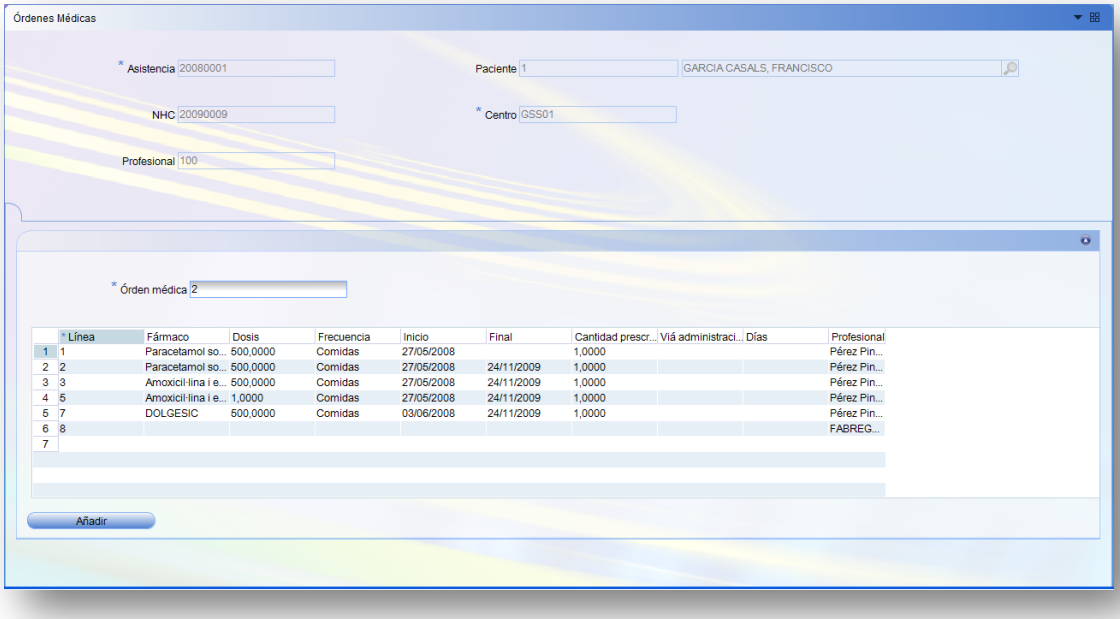

Figura 23: Ventana de Órdenes Médicas

En la vista estándar de esta ventana, como en la de Antecedentes, no se puede añadir ni modificar información. Para ello deberemos abrir la vista ampliada o maximizada. En la vista estándar (figura 23) veremos el paciente y la asistencia que estamos tratando y sus órdenes médicas en un grid.

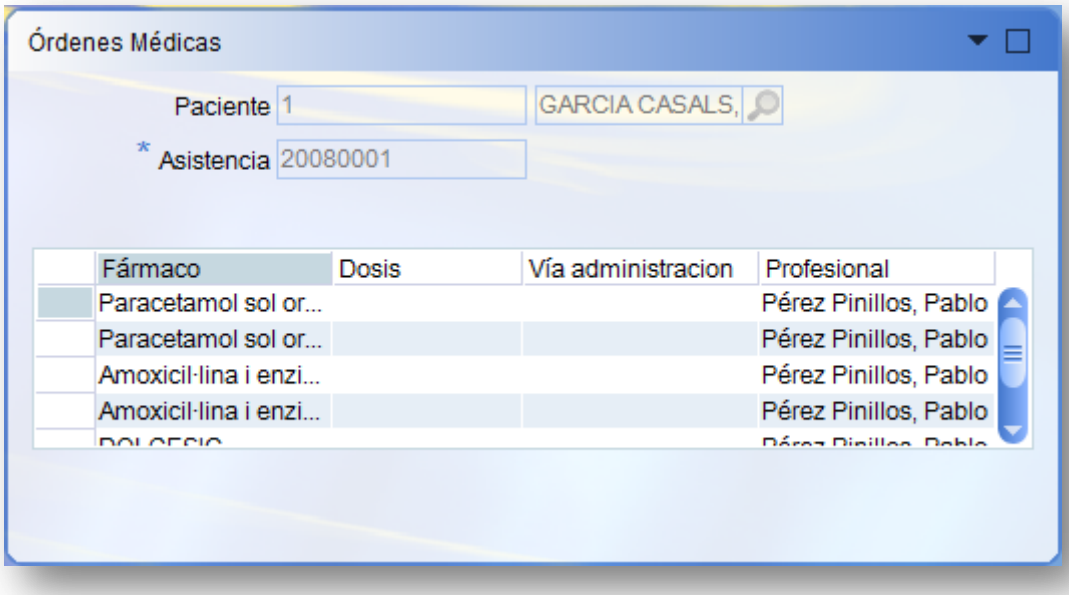

Figura 24: Vista estándar de la ventana de Órdenes Médicas

5.4. Ventanas relacionadas con el profesional

A partir de aquí nos centraremos en las ventanas relacionadas exclusivamente con el profesional, sus datos y sus tareas o actividades.

5.4.1. Datos Profesional

Ventana simple, en la que se muestran los datos del profesional que está trabajando en el escritorio. Sirve como información adicional, para que el usuario sepa con que profesional está trabajando, en el centro en que está actuando, así como su especialidad y servicios.

Esta ventana corresponde al formulario de selección activa pero ahora dentro del escritorio. La respuesta de porque entonces utilizamos este formulario fuera del escritorio si luego lo tenemos también dentro, es que el escritorio no permite un refresco simultáneo de todas sus ventanas. Por lo tanto, si el cambio de información de este formulario se hiciese con el escritorio abierto, habría que cerrarlo y volverlo a abrir.

Por ello se ha decidido hacer el formulario de selección activa explicado al principio de este apartado y luego añadir de manera informativa esta ventana. Sólo utiliza un panel HEADER que recibe los datos del profesional, centro, especialidad y equipo de las variables de entorno del formulario de selección activa (figura 24).

Figura 25: Ventana de Datos del Profesional

5.4.2. Tareas Pendientes

En este apartado veremos las ventanas que corresponden a las tareas que el profesional tiene pendientes de realizar. Está formado por dos ventanas: informes de alta y peticiones.

5.4.2.1. Informes de Alta

Esta ventana es la primera de las tareas pendientes de un profesional. En ella se muestran los Informes de Alta de los pacientes que aún no se han cerrado o no se han finalizado.

Un informe de alta médica es un documento emitido por el médico responsable en un centro sanitario al finalizar cada proceso asistencial de un paciente, que especifica los datos de éste, un resumen de su historial clínico, la actividad asistencial prestada, el diagnóstico y las recomendaciones terapéuticas**.** Por lo tanto, podemos decir que cada informe de alta está asociado con una asistencia de un paciente.

En nuestro caso, al no disponer del módulo de *ekon Salus* de Informes de Alta donde se gestionan y mantienen éstos, se ha decidido mostrar una lista de los informes con una breve explicación de sus detalles.

A continuación explicaremos como se ha implementado esta ventana:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_informesalta*", está compuesto por dos paneles:
  - Panel HEADER: En este panel, tenemos agrupados los campos que nos servirán como filtro para los informes de alta. En primer lugar tenemos el campo del centro como parámetro de entrada, que viene determinado por la variable de entorno correspondiente del formulario de selección activa y por el que filtramos nuestros informes. Cabe decir que esta ventana está filtrada únicamente por el centro sanitario ya que un profesional puede ver todos los Informes de Alta no finalizados del centro donde trabaja, sin necesidad de ser suyos.

    En segundo lugar, tenemos los campos de id y fecha del informe que recibiremos de la clase de personalización y nos servirán para filtrar el panel de Informes de Alta.
  - Panel de Informes de Alta: hace referencia a la tabla "*vh_as_rpt*" y de ella obtenemos los datos de los informes que se están tratando en esta ventana, filtrado por los campos descritos en el HEADER. Estos datos son la asistencia a la que pertenece, la descripción y el tipo de petición, la fecha de creación del informe, el profesional que lo creó y la situación en que se encuentra, así como el paciente a quien pertenece.
- **Formulario:** el formulario, que es como veremos nuestra ventana, estará formado de la siguiente manera:
  - Cabecera: formada por el campo del centro con el que estamos trabajando y por el que se filtra esta ventana.

- Pestaña de Informes de Alta:
  - Árbol de Informes de Alta: disponemos de un campo *"FMTreeView"* done mostraremos los informes de alta del profesional en un árbol. La carga y pintada de los informes se realizará en la clase de personalización de java. Se sitúa a la izquierda de la pestaña.
  - Datos de los Informes: situados a la derecha de la pestaña, está formada por diversos campos de texto y un grid donde mostramos los datos de los informes tal y como hemos descrito en el Panel de Informes de Alta del objeto de negocio. Podemos decir que esta parte se corresponde con el panel citado con anterioridad.
- Botonera: aquí es donde disponemos de un botón "*Consultar Altas desde"* y un campo de fecha que serán tratados en la clase de personalización, donde explicaremos sus funciones.

- **Clase de personalización:** La clase de personalización de esta ventana se denomina "*InformesAlta"*. En ella se realiza la funcionalidad de más relevancia de esta ventana. En primer lugar hemos de explicar cómo se forma nuestro árbol de informes. Para ello disponemos de las funciones o métodos "*informesAlta*" y "*loadInformes*". En la primera de ellas creamos la consulta necesaria mediante SQL para acceder a las diversas tablas donde están los datos que necesitamos: las más importantes son las de asistencias (*vh_as_asistencias* y *vh_asistencias)*, las de informes de alta (*vh_as_rpt* y *vh_rpt_informes*) y la de datos de los pacientes (*vh_pac_gral*). En la segunda función, tratamos la sentencia SQL anterior y creamos el árbol asignándolo al campo del formulario denominado *FMTreeView.* De esta manera ya podemos ver en nuestro árbol dichos informes formados con una cabecera: fecha de creación del informe + nombre del paciente + versión del informe de alta; y agrupados según el tipo de asistencia (hospitalización, urgencias, farmacia, ambulatoria,…). Cabe destacar que estas funciones se llaman justo cuando se inicia o abre la ventana. En la figura 25 podemos ver cómo queda nuestro árbol.

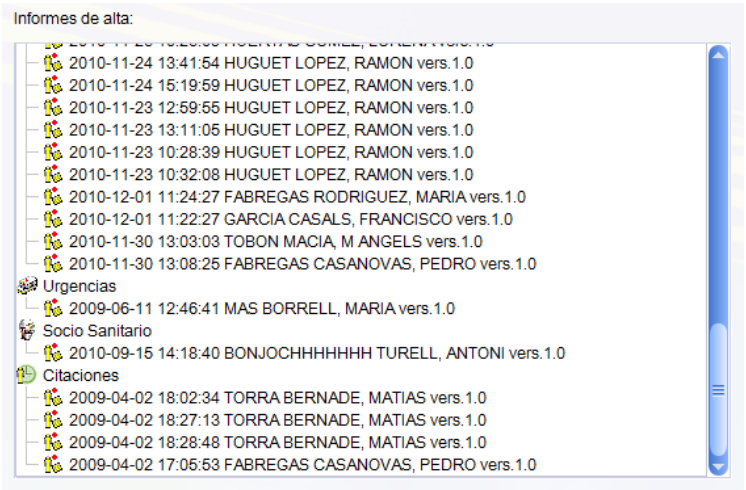

Figura 26: Árbol de informes de la ventana Informes de Alta

Una vez creado el árbol, esta ventana también permite seleccionar un informe de alta de este árbol y ver a la derecha, en la pestaña de Informes de Alta, los datos más relevantes e interesantes para el profesional. La selección del informe se hace mediante la función *"containerNodeClick"* que nos detecta qué nodo hemos seleccionado y nos devuelve una cadena de caracteres a tratar que corresponden con el informe, asistencia e id del paciente. Tratamos esta información y, a partir de los objetos *karat* (objeto de negocio y formulario), mostramos esta información en la pestaña.

El profesional también dispone de un filtro de fecha, que se encuentra en la Botonera, a través del cual podemos ver sólo los informes a partir de la fecha indicada en adelante. Utilizando la función *"viewClick"* detectamos cuando presionamos este botón, y volvemos a llamar a las funciones que crean y pintan el árbol, esta vez con una restricción más, la de la fecha introducida.

En la figura 26 podemos ver cómo queda la ventana completa.

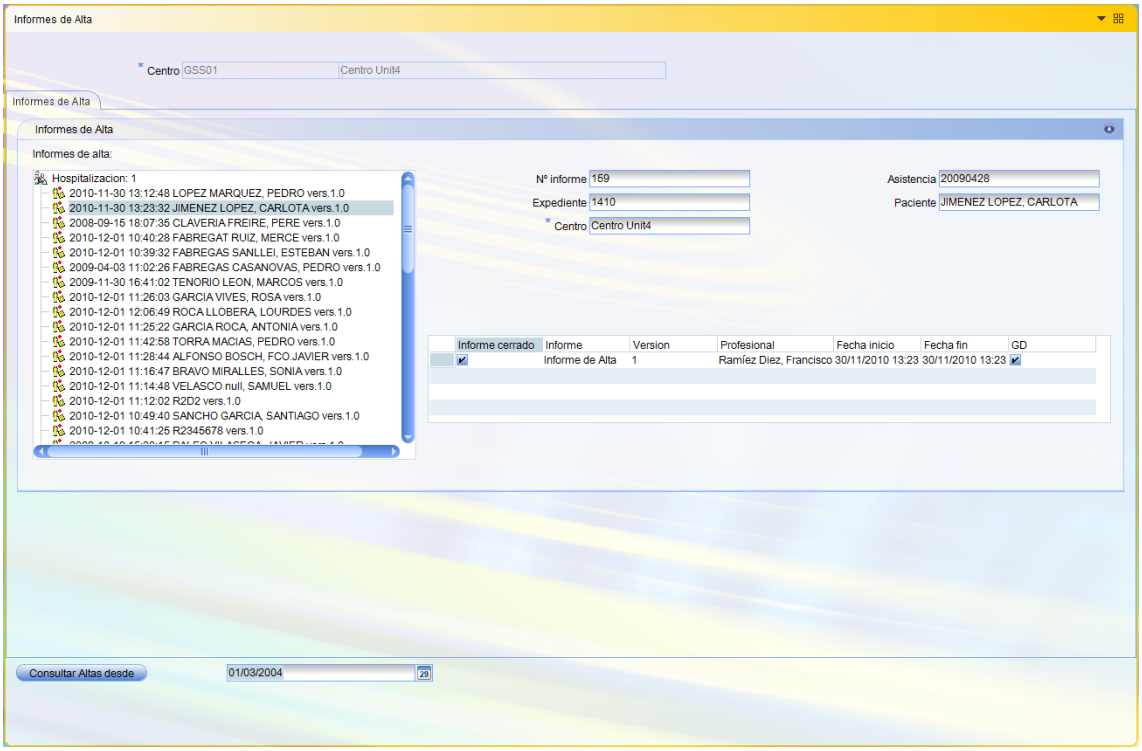

Figura 27: Ventana completa de Informes de Alta

En la figura 27 podemos ver la vista estándar de nuestra ventana en el escritorio. En él sólo vemos una pequeña cabecera donde se nos informa sobre qué centro estamos trabajando y el árbol de informes.

Figura 28: Vista estándar de Informes de Alta

5.4.2.2. Peticiones

En esta ventana veremos las peticiones de los profesionales. Las peticiones son solicitudes de pruebas para un paciente, como por ejemplo unas analíticas o unas placas. Por ejemplo un profesional con perfil cirujano puede pedir una prueba solicitando una extracción de sangre para preparar el preoperatorio del paciente. Hay tres tipos de peticiones:

- Solicitadas: peticiones que el profesional ha solicitado a otros profesionales especialistas en la prueba requerida.
- Recibidas: peticiones que el profesional ha recibido de otros profesionales.
- Otras: como su nombre indica, pruebas o peticiones que no corresponden a ningún grupo anterior.

Esta ventana sigue la misma estructura que la de Informes de Alta, sólo cambia la información que presenta. Así pues su implementación es muy parecida. Aún asi, una de las diferencias, es que esta ventana está filtrada por profesional, ya que sólo éste puede ver sus peticiones privadas, sean del tipo que sean (evidentemente, una misma petición la puede ver tanto el profesional que la solicita como el que la recibe).

A continuación explicaremos como se ha implementado esta ventana:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_peticiones*", está compuesto por dos paneles:
  - Panel HEADER: tenemos agrupados los campos que nos servirán como filtro para las peticiones. En primer lugar tenemos los campos del centro y profesional como parámetros de entrada, que vienen determinados por la variable de entorno del formulario de selección activa, y por el que filtramos nuestras peticiones. En segundo lugar tenemos el campo de la id de la petición

que recibiremos de la clase de personalización y que nos servirá para filtrar el panel de peticiones.

- Panel de Peticiones: Este panel hace referencia a la tabla "*vh_as_peticiones*" y de ella obtenemos los datos de las peticiones que se están tratando en esta ventana: la asistencia a la que pertenece, la descripción y el tipo de la petición, la fecha, la situación en que se encuentra, así como el profesional que ha enviado la petición o al que va destinada.

- **Formulario:**
  - Cabecera: formada por los campos del centro y profesional con los que estamos trabajando y por los que se filtra esta ventana.
  - Pestaña de Peticiones:
    - Árbol de Peticiones: disponemos de un campo *"FMTreeView"* done mostraremos las peticiones pendientes del profesional en un árbol. La carga y pintada de las peticiones se realizará en la clase de personalización de java. Se sitúa a la izquierda de la pestaña.
    - Datos de Peticiones: Situada a la derecha de la pestaña. Está formada por diversos campos de texto y un grid donde mostramos los datos de las peticiones tal y como hemos descrito en el Panel de Peticiones del objeto de negocio. Esta parte se corresponde con el panel citado con anterioridad.

- **Clase de personalización:** La clase de personalización de esta ventana se denomina "*Peticiones"*. En ella es donde se realiza la mayoría de la funcionalidad de esta ventana, que intentaremos explicar lo mas sencillamente posible:

  En primer lugar hemos de explicar cómo se forma nuestro árbol de peticiones. Para ello disponemos de las funciones "*peticionesTipoProf*" y "*petiProf*". La primera crea una consulta, mediante SQL, que accede a las tablas donde se encuentran los datos que necesitamos, siempre filtrados por el profesional que tenemos en la cabecera. Las tablas más importantes son las de asistencias (*vh_asistencias* y *vh_as_asistencia*), la de peticiones (*vh_as_peticiones*) y la de datos del paciente (*vh_pac_gral*). La segunda de ellas trata los datos a los que hemos accedido y monta y pinta el árbol en el campo *"FMTreeView.* El árbol queda de la siguiente manera.

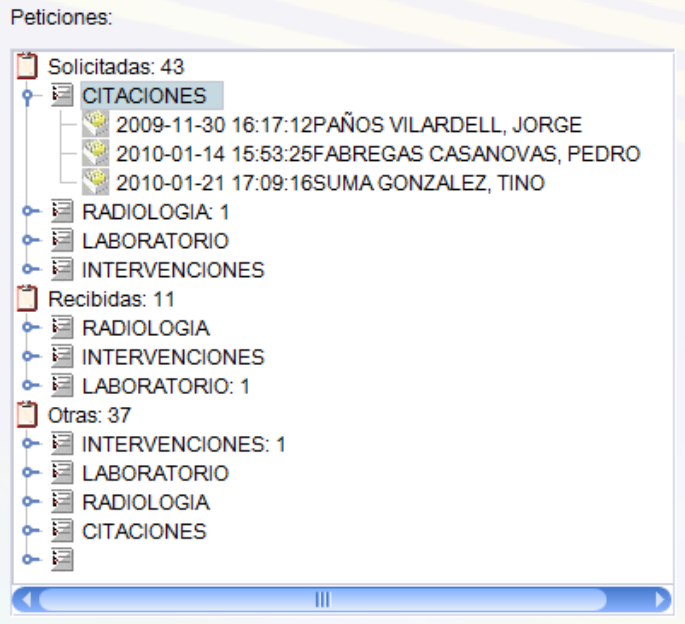

Figura 29: Árbol de peticiones

Una vez creado el árbol, se desea poder seleccionar cualquier petición de éste y ver sus datos con más detalle. Para ello disponemos de la función *"containerNodeClick"*. A partir de esta función se detecta qué nodo se está seleccionando y éste nos devuelve su identificador, que es una cadena de caracteres que hemos creado a la hora de montar el árbol (con la asistencia, petición, datos del paciente, etc.). Tratamos esta información y se la pasamos a los diversos campos mediante el objeto de negocio. De esta manera el profesional, ya puede ver los datos detallados de cada petición.

A continuación podemos ver cómo queda la ventana completa:

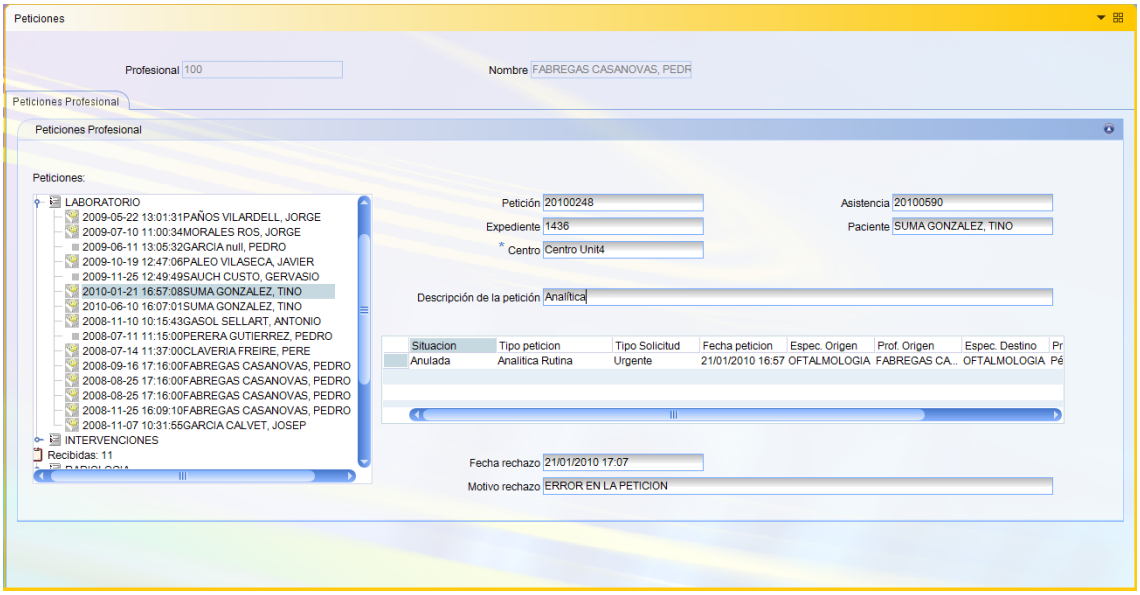

Figura 30: Ventana completa de Peticiones

En la vista estándar de nuestra ventana en el escritorio (figura 30), sólo vemos una pequeña cabecera donde se nos informa de qué profesional está trabajando, y el árbol de peticiones. Para más detalles debe de abrir la ventana completa.

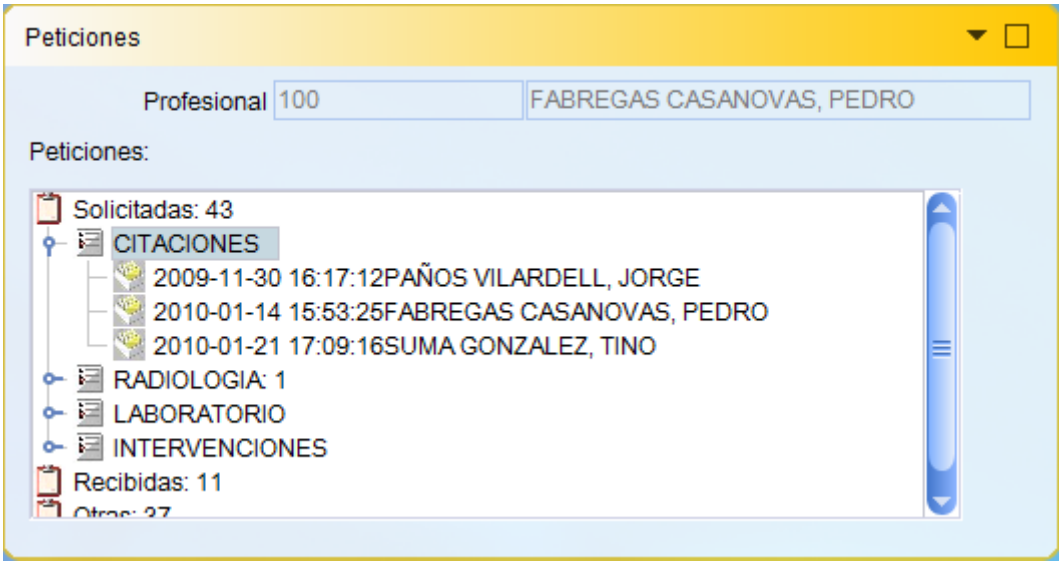

Figura 31: Vista estándar de Peticiones

5.4.3. Agenda Personal

Otra de las facilidades con las que cuenta el profesional es una pequeña ventana donde dispone de su agenda personal. En ella puede añadir cualquier nota o tarea que tenga que realizar y ver en cada momento las que ya tiene programadas.

A continuación explicaremos como se ha implementado esta ventana:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_agendadia*", está compuesto por dos paneles:
  - Panel HEADER: tenemos agrupados los campos que nos servirán como filtro para la agenda. En primer lugar, tenemos el campo de profesional como parámetro de entrada, que viene determinado por la variable de entorno del formulario de selección activa, y por el que filtramos la agenda. En segundo lugar tenemos dos campos de fecha (*fecha_inicio y fecha_fin)* que le servirán al profesional para filtrar, si lo desea, las tareas de la agenda. Estas fechas serán tratadas en la Clase de personalización y se filtrarán en el panel de la agenda del profesional.
  - Panel de la Agenda del profesional: Este panel hace referencia a la tabla "*vh_etm_prof_age*" y de ella obtenemos los datos que este profesional quiere mostrar sobre su agenda personal en esta ventana: la tarea a realizar y la fecha en que se ha de realizar.
- **Formulario:**

- Cabecera: formada por el campo del profesional por los que se filtra esta ventana.
- Pestaña de Agenda Personal: formada por dos campos de texto de las diferentes fechas (de inicio y fin) y un check (día actual), que el profesional puede utilizar para filtrar información. Además incluy un grid donde el profesional podrá ver la lista de las tareas que ha realizado o tiene que realizar.
- Botonera: aquí disponemos de los botones "*Actualizar*" y"*Añadir"* que serán tratados en la clase de personalización.

- **Clase de personalización:** La clase de personalización de esta ventana se denomina "*IntervencionesAgenda"*. En ella disponemos de la función "*viewClick*" que permite detectar cuando el usuario, o profesional, pulsa un botón. En este caso disponemos de 2 botones:
  - Actualizar: una vez el profesional ha añadido los campos de fecha por los cuales quiere filtrar el grid puede pulsar este botón. Si lo hace, mediante sentencia SQL, el grid es actualizado a partir de los datos introducidos. Si el check de "día actual" está informado, sólo mostrará las intervenciones del día actual en el que se encuentre el profesional, independientemente de las fechas introducidas en los otros dos campos.
  - Añadir: Con este botón le damos al profesional la posibilidad de añadir una nueva tarea o nota a la lista de su agenda, para poder visualizarla. Al pulsar este botón se abre una ventana modal que permite añadir dicha tarea y la fecha en que se ha de realizar. Estos datos se añaden mediante sentencia SQL a la tabla *vh_etm_prof_age* e inmediatamente podemos observarlos en nuestro grid.

A continuación podemos como queda estructurada la ventana completa, con la ventana modal de inserción de una tarea:

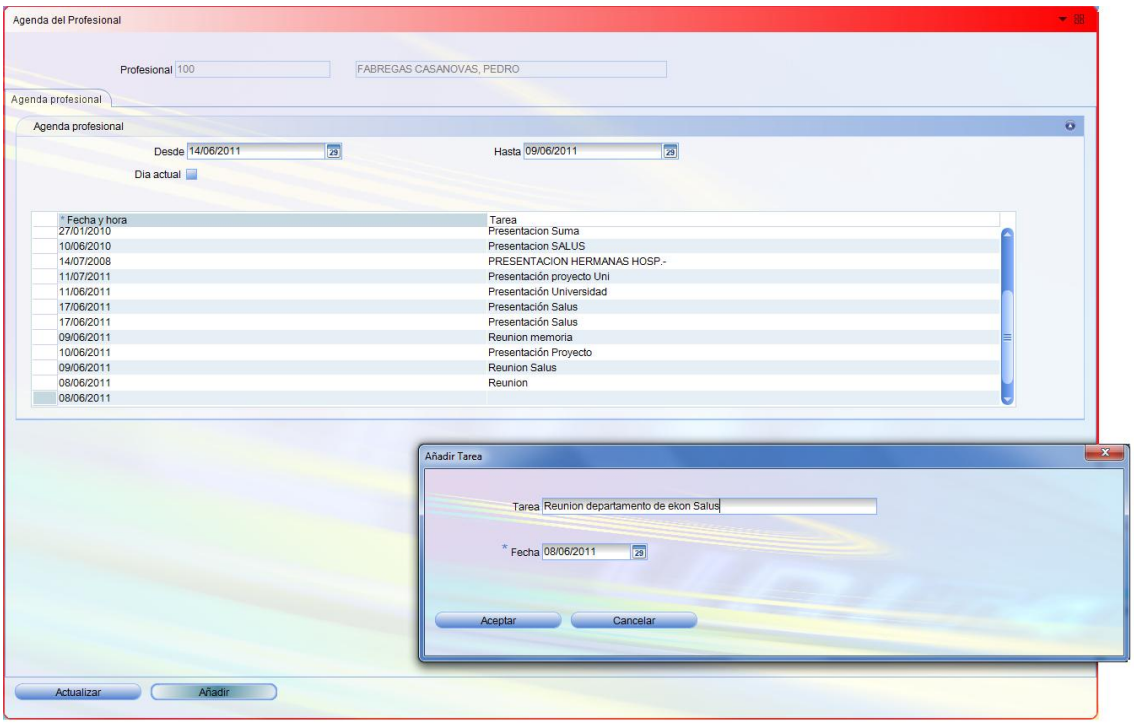

Figura 32: Ventana de Agenda Personal

En esta figura, podemos observar todo lo explicado con anterioridad. La ventana se filtra a través del parámetro del profesional y en el grid puede ver las tareas o notas que tiene. A partir de los campos de fecha puede filtrar esta información e incluso se le da la posibilidad de añadir nuevas tareas si lo requiere pulsando los botones correspondientes.

En su vista estándar (figura 32), observamos únicamente un grid con las tareas o notas que el profesional tiene asignadas por fecha.

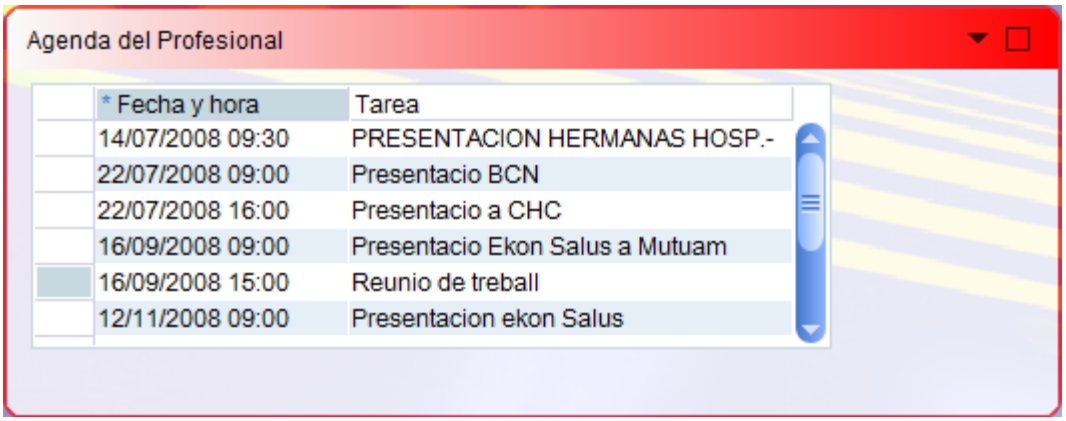

Figura 33: Vista estándar de la ventana de Agenda Personal

5.4.4. Intervenciones

En esta ventana el profesional podrá ver las intervenciones que aún tiene pendientes, tanto del día actual, como de fechas pasadas o venideras. Las intervenciones son actividades operatorias que el profesional ha de realizar, según su especialidad.

Para la construcción de esta ventana utilizaremos la misma estructura que en la anterior de Agenda Personal del profesional. La única diferencia, aparte de mostrar y acceder a diferentes tablas e información como es evidente, es que el profesional no podrá agregar ninguna intervención. Para ello deberá abrir el módulo de *Salus* que corresponda a las intervenciones y la añadirá, puesto que este es un proceso más complejo.

A continuación explicaremos qué objetos hemos utilizado y como, para la implementación de nuestra ventana:

- Objeto de negocio: El objeto de negocio de esta ventana, llamado "*aa_intervenciones*", está compuesto por dos paneles:
  - Panel HEADER: tenemos concentrados todos los campos que nos servirán como filtro para las intervenciones. En primer lugar, tenemos los campos de profesional y centro, que vienen determinados por las variables de entorno del formulario de selección activa, y por los que filtraremos las intervenciones. En segundo lugar, tenemos tres campos de fecha (*fecha_inicio, fecha_final y fecha_dia*), que servirán al profesional para filtrar, si lo desea, las intervenciones. Estas fechas serán tratadas en la Clase de personalización.
  - Panel de Intervenciones: Este panel hace referencia a la tabla "*vh_as_intervenciones*" y de ella obtenemos los datos de las intervenciones que queremos mostrar en esta ventana: como la fecha de la intervención, el paciente al que se le realiza, el tipo de intervención o la prioridad.
- Formulario: el formulario, que es como veremos nuestra ventana, estará formado por:
  - Cabecera: formada por los campos del panel HEADER, por los que se filtra esta ventana: el centro y el profesional.
  - Pestaña de Intervenciones: formada por dos campos de texto de las diferentes fechas (de inicio y fin) y un check (*fecha_dia*), que el profesional puede utilizar para filtrar información, y por un grid donde el profesional podrá ver la lista de las intervenciones que tiene pendientes de realizar.
  - Botonera: aquí añadiremos nuestro botón "*Actualizar*".
- Clase de personalización: Esta ventana comparte clase de personalización con la Agenda personal, llamada "*IntervencionesAgenda"*, puesto que las dos ventanas utilizan la misma función "*viewClick*". Esta función permite detectar cuando el usuario o profesional aprieta un botón. En este caso tenemos un solo botón:
  - Actualizar: una vez el profesional ha añadido los campos de fecha por los cuales quiere filtrar el grid pulsa este botón. Entonces, mediante sentencia SQL, el grid es actualizado a partir de los datos introducidos. Si el check de "día actual" está

informado, sólo mostrará las intervenciones del día actual independientemente de las fechas introducidas en los otros dos campos.

Por todo lo descrito antes, la composición de nuestra ventana queda así:

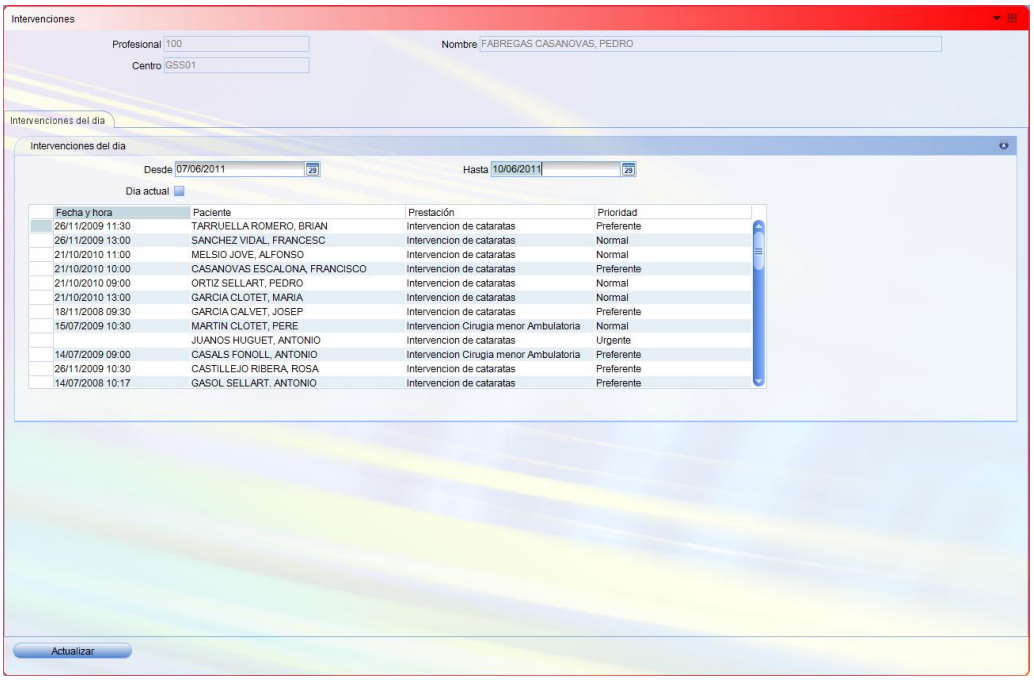

Figura 34: Ventana de Intervenciones

Como podemos observar en la imagen anterior, el profesional se encuentra en primer lugar con la cabecera, que le indica sobre qué centro y profesional se está trabajando, y en la pestaña que incluye el grid, y por lo tanto la información que quiere ver. Si lo desea puede filtrar este grid introduciendo las fechas en los campos correspondientes.

En la vista estándar (figura 34) de nuestra ventana solo veremos el grid de las intervenciones y no podremos realizar ninguna de las acciones anteriores.

Figura 35: Vista estándar de la Ventana de Intervenciones

5.5. Ventanas relacionadas con el centro hospitalario

Estas ventanas, Censo del Hospital y Citaciones, son las principales ventanas de nuestro escritorio, de donde se cogen la id de los pacientes y las asistencias que necesitan las otras ventanas.

5.5.1. Censo del Hospital

En esta ventana, como su nombre indica, el profesional puede ver el censo de pacientes del centro sanitario u hospital donde está trabajando. Un censo, es la relación de pacientes y asistencias que se encuentran registradas en dicho centro. En él, se pueden ver tanto los pacientes que ya están ingresados como los pacientes que están en Lista de espera.

Cabe decir que para que esta ventana muestre la relación de pacientes y asistencias que luego se utilizarán como parámetro de salida hacia otras ventanas, primero se ha de informar sobre qué profesional y centro estamos trabajando. Para ello utilizamos las variables de entorno del formulario de selección activa.

A continuación explicaremos cómo se ha implementado esta ventana:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_censo*", está compuesto por tres paneles:
  - Panel HEADER: tenemos agrupados los campos que nos servirán como filtro para el resto de paneles. En primer lugar tenemos los campos de profesional y centro que provienen de las variables de entorno. En segundo lugar tenemos dos campos, tipo de asistencia y tipo de médico que utilizaremos para filtrar los respectivos paneles. Estos dos campos serán tratados en la Clase de personalización.
  - Panel de Pacientes Ingresados: hace referencia a la tabla "*vh_as_censo_activo*" y de ella obtenemos la lista de pacientes ingresados que hay en el centro de trabajo. Este panel está filtrado por el mencionado centro, por el profesional y por el tipo de asistencia, todos ellos provienen del HEADER.
  - Panel de Lista de espera: hace referencia a la tabla *"vh_h_lista_espera"* de donde obtenemos la relación de pacientes que se encuentra en lista de espera del centro. Este panel está filtrado por centro y por tipo de asistencia. El profesional no es necesario, ya que todos los profesionales de un centro pueden ver la Lista de espera.
- **Formulario:** el formulario, que es como veremos nuestra ventana, estará formado por:
  - Cabecera: formada por los campos de texto del profesional y centro por los que se filtra esta ventana, de manera informativa. Además dispone de un campo de lista desplegable donde se escoge el tipo de asistencia del que se quiere consultar el censo.

- Pestaña de Pacientes Ingresados: Esta formada por un grid o tabla donde el profesional podrá ver la lista de los pacientes ingresados así como la asistencia que tienen relacionada. Hace referencia al Panel de pacientes ingresados.
- Pestaña de Lista de espera: formada por un grupo de botones de tipo check, en el que se escoge el tipo de médico, si un médico ha solicitado el ingreso del paciente o no. Una vez escogido, muestra un grid con todos los pacientes en lista de espera con el motivo de ingreso.

- **Clase de personalización:** La clase de personalización de esta ventana se denomina "*Global"* pues como hemos dicho antes es la que utilizamos para enviar la id de los pacientes a otras ventanas, junto con Citaciones que comparte la misma clase.

  En primer lugar, explicaremos como pasamos estos datos al resto de ventanas. Hay dos maneras: mediante Ítem del objeto negocio o mediante parámetro Libre. Si son ítems es mucho más sencillo, pues solo hay que relacionar una ventana con otra mediante sus parámetros de entrada y salida. Si son por parámetro libre se realiza mediante código java. Para ello disponemos de un método que se llama *"containerRowChange"* que detecta cuando seleccionamos una fila de nuestro grid o cambiamos a otra diferente. En ese momento, enviamos la asistencia o la id del paciente (según se requiera en cada momento) al resto de ventanas a través de la función *"setDeskBoxParameter"* y que las ventanas reciben con su "*deskBoxParameter*" que hemos explicado en cada una de ellas. Ya tenemos relacionadas las ventanas. En la sección de "Relaciones entre ventanas" explicaremos porque todos los parámetros no se pueden pasar mediante Ítem de negocio.

  A partir de aquí, explicaremos el resto de funcionalidad que ofrece esta clase de personalización.

  Una vez abierta esta ventana, hemos de seleccionar el tipo de asistencia para poder ver los pacientes ingresados o en lista de espera. Para ello ya hemos dicho que disponíamos de una lista desplegable donde la seleccionábamos. La creación, pintado y carga de esta lista se hace mediante las funciones *"cargaComboAsistencias"* y *"pintaComboAsistencias"* respectivamente. La primera crea la sentencia SQL para acceder a la tabla *"vh_aux_tipo_capi"* de donde obtendremos los diferentes tipos de asistencia. La segunda trata la sentencia SQL y la pinta en el combo correspondiente mediante una lista dinámica.

  Una vez cargada la lista, ya podemos seleccionar una para cargar los grids mediante los objetos *karat* descritos con anterioridad.

  La última funcionalidad de esta ventana se encuentra en la pestaña de Lista de espera. Una vez cargada (habiendo elegido el tipo de asistencia), tenemos la posibilidad de elegir, mediante el grupo de check, si el ingreso a esta lista ha sido por parte del médico o no (solicitante o atención). Mediante la función *"formItemSelect"* detectamos qué opción del grupo de checks hemos seleccionado y tratamos mediante SQL la nueva restricción que padecerá el grid de lista de espera.

  A continuación podemos como queda estructurada la ventana completa:

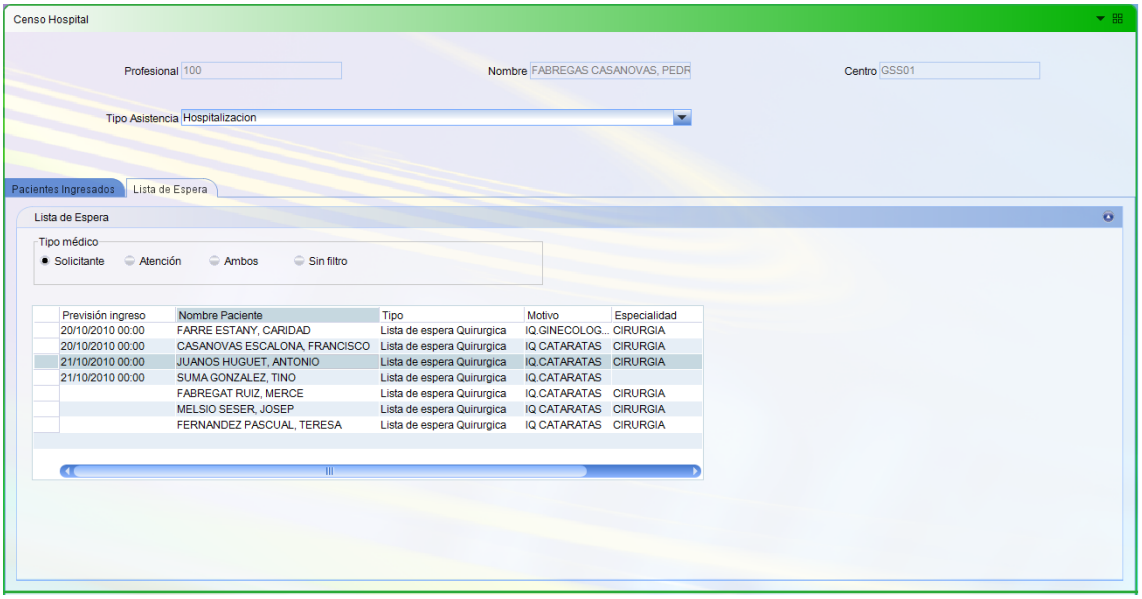

Figura 36: Ventana de Censo del Hospital

En la figura anterior, podemos observar: la cabecera con los campos de filtro y el tipo de asistencia y las diferentes pestañas con sus grids. A continuación podemos ver cómo queda nuestra vista estándar en el escritorio de esta ventana. En él solo vemos el tipo de asistencia a elegir por el profesional y el grid de los pacientes ingresados:

Figura 37: Vista estándar de la ventana de Censo del Hospital

5.5.2. Citaciones

En esta ventana el profesional puede ver las citas médicas de los pacientes de un centro sanitario u hospital donde está trabajando. La diferencia entre esta ventana y la del Censo es la finalidad de la asistencia, ya que las dos dependen de ella. La asistencia del Censo corresponde a la solicitud de ingreso de un paciente en un Centro por el motivo que sea. Sin embargo un paciente registrado en la lista de espera de citaciones, está pendiente de la confirmación de fecha y hora de una citación. Las citaciones están gestionadas desde las agendas de cada profesional.

Al igual que en el Censo, para que esta ventana muestre la relación de pacientes y asistencias, primero se ha de informar sobre qué profesional y centro estamos trabajando. Para ello utilizamos las variables de entorno del formulario de selección activa.

La construcción o implementación de esta ventana es muy parecida a la de Censo. A continuación explicaremos cómo se ha hecho:

- **Objeto de negocio:** El objeto de negocio de esta ventana, llamado "*aa_citaciones*", está compuesto por cuatro paneles:
  - Panel HEADER: tenemos agrupados los campos que nos servirán como filtro para el resto de paneles. En primer lugar tenemos los campos de profesional y centro que provienen de las variables de entorno. En segundo lugar tenemos tres campos, tipo de agenda, tipo de médico y fecha que utilizaremos para filtrar los respectivos paneles. Estos tres campos serán tratados en la Clase de personalización.
  - Panel de Visitas del día: Este panel hace referencia a la tabla "*vh_ce_citas*" y de ella obtenemos la lista de visitas a pacientes que hay programadas en el centro de trabajo.
  - Panel de Pendientes de reprogramar: hace referencia a la tabla *"vh_ce_reprogramar"* de la que obtenemos la lista de pacientes que tenían una vista prevista y la han cancelado. Por lo tanto están pendientes de reprogramar. Viene filtrado por el tipo de agenda.
  - Panel de Lista de espera: hace referencia a la tabla *"vh_ce_lista_espera"* de donde obtenemos la relación de pacientes que se encuentra en lista de espera de citaciones.
- **Formulario:** el formulario, que es como veremos nuestra ventana, estará formado de la siguiente manera:
  - Cabecera: formada por los campos de texto del profesional y centro por los que se filtra esta ventana, así como el equipo médico, de manera informativa. Además dispone de un campo de lista desplegable donde se escoge la agenda de la que se quiere consultar las citaciones.
  - Pestaña de Visitas del día: Esta formada por un grid o tabla donde el profesional podrá ver la lista de las citaciones a los pacientes que tiene que visitar así como datos importantes de dicha citación. Además se dispone de un campo de texto

de tipo fecha y dos cheks (día actual y todos) para filtrar las citaciones que se ven en el grid. Esto se hará mediante la clase de personalización. Hace referencia al panel de visitas del día. También dispone de un botón *"Actualizar"*

- Pestaña de Pendientes de reprogramar: se muestra un grid con los pacientes que están pendientes de reprogramar. Hace referencia al panel de pendientes de reprogramar.
- Pestaña de Lista de espera: formada por un grupo de botones de tipo check, en el que se escoge el tipo de médico, si un médico ha solicitado el ingreso del paciente o no. Una vez escogido muestra un grid con todos los pacientes en lista de espera con el motivo de ingreso.

- **Clase de personalización:** Como hemos dicho comparte clase de personalización con Censo, y se denomina "*Global"*. La funcionalidad es parecida.

  La manera en que lo datos se pasan al resto de ventanas se hace de la misma manera que en el Censo, mediante ítems del objeto de negocio o parámetros libres. Estos últimos mediante la función "*containerRowChange"* y *"setDeskBoxParameter"* que ya hemos explicado.

  A partir de aquí, explicaremos el resto de funcionalidad que ofrece esta clase de personalización.

  De la misma manera que en Censo, una vez abierta esta ventana, hemos de seleccionar, esta vez, el tipo de agenda para poder ver los diferentes datos en las pestañas correspondientes. Para ello disponemos de una lista desplegable donde seleccionamos la agenda. La creación y pintado de esta lista se hace mediante la función *"loadComboAgendas"* que abre la SQL donde están guardadas las agendas, en la tabla *"vh_ce_agendas"* y las pinta en la lista. Esta función filtra las agendas por equipo médico (profesional y centro).

  Una vez cargada la lista ya podemos hacer la selección una para cargar los grids mediante los objetos *karat*. El profesional puede filtrar las citaciones que ve en la pestaña de visitas del día a partir de la fecha indicada en el campo de texto correspondiente o seleccionando los checks. Al apretar el botón *"Actualizar"*, mediante la función *"viewClick"* miramos que checks están activos y la fecha que ha sido introducida para actualizar el grid mediante sentencia SQL. Si el check "día actual" esta activado muestra las citaciones del día actual. Si el check activado es el de "todos" muestra las citaciones de todos los días. Si ninguno de estos dos está activado, coge la información de la fecha que le hayamos asignado mostrando las citaciones del día elegido.

  La última funcionalidad de esta ventana se encuentra en la pestaña de Lista de espera y es la misma que en el Censo. Una vez cargada (habiendo elegido el tipo de agenda), tenemos la posibilidad de elegir, mediante el grupo de checks, si el ingreso a esta lista ha sido por parte del médico o no (solicitante o atención). Mediante la función *"formItemSelect"* detectamos qué opción del grupo de checks hemos seleccionado y tratamos mediante SQL la nueva restricción que padecerá el grid de lista de espera.

En la figura 37 podemos como queda estructurada la ventana completa, con la ventana modal de inserción de una tarea:

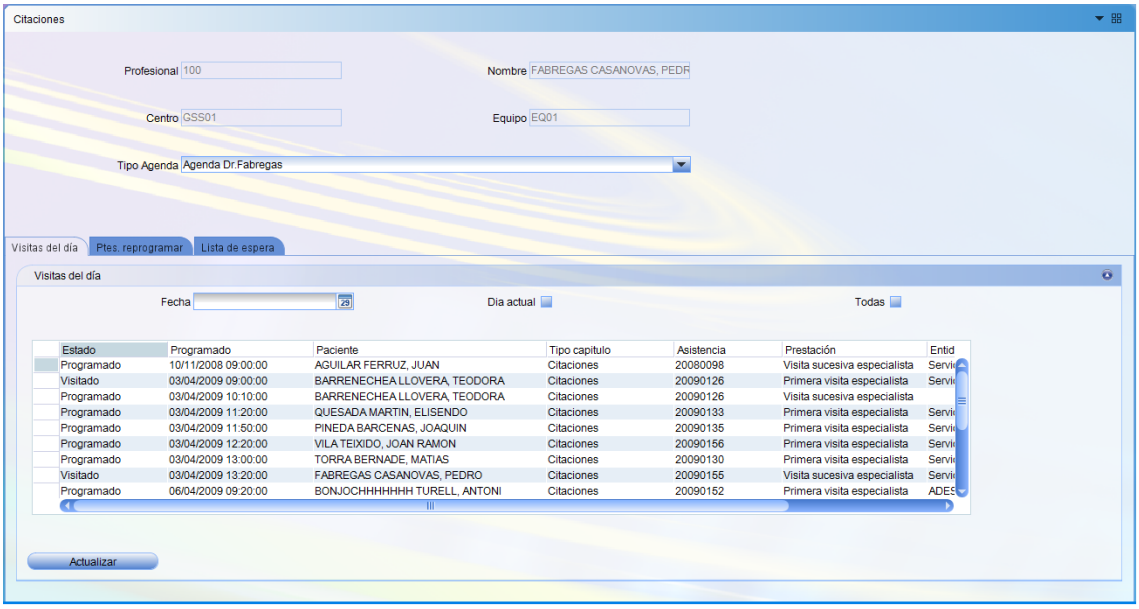

Figura 38: Ventana de Citaciones

En la vista estándar de esta ventana (figura 38) sólo observaremos la lista desplegable con la agenda a escoger y el grid de las citaciones médicas. A continuación podemos verlo:

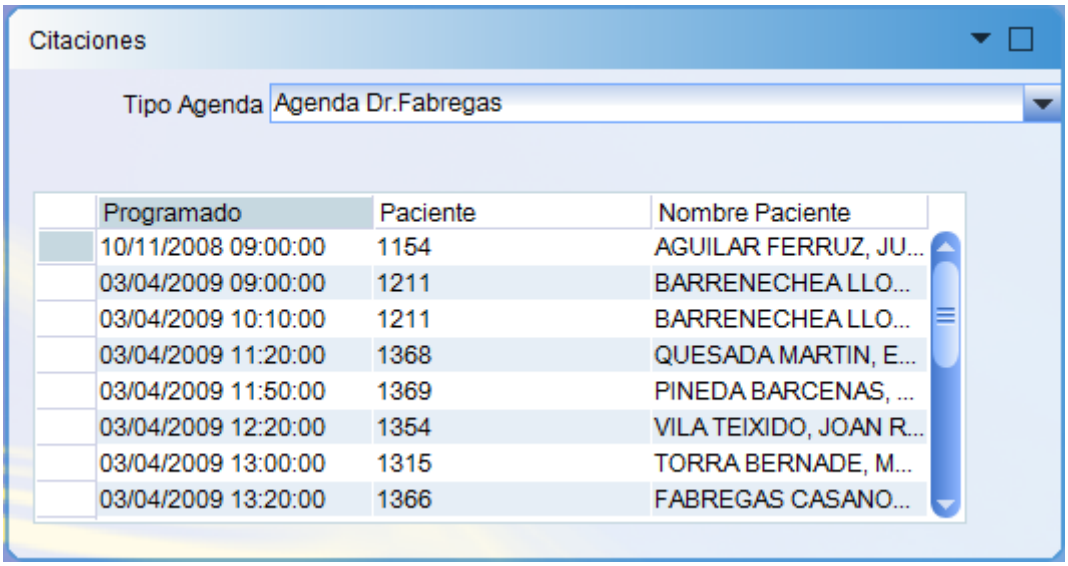

Figura 39: Vista estándar de Citaciones

5.6. Resumen de las relaciones entre ventanas

Como hemos explicado en las ventanas anteriores, las relaciones entre ventanas se hacen por parámetros de entrada y salida que se definen en la definidora de mini aplicaciones y están relacionados uno a uno. Hemos dicho que hay de dos tipos:

- Ítem de objeto de negocio: se relacionan en la misma herramienta de creación de escritorios (definidora *ka_boxapp)*.
- Libre: mediante la clase de personalización detectamos qué campo está seleccionado y debe enviarse a otras ventanas.

Como ítem es mucho más sencillo, el problema es que el escritorio sólo admite que una ventana tenga un único parámetro de entrada por el mismo ítem. El resto se han de hacer por parámetro libre en la clase de personalización. Pondremos de ejemplo a la ventana Datos Paciente: ésta recibe dos parámetros de entrada iguales, la id del paciente, pero uno de ellos proviene de Censo y el otro de Citaciones, según cual estemos seleccionando en ese momento. Por lo tanto, uno de estos parámetros puede ser por Ítem pero el otro ha de ser Libre. Y así ocurre con Antecedentes y Alergias y con Órdenes Médicas.

Una de las posibilidades visuales que nos da el escritorio para ver las ventanas que se relacionan entre sí son los colores de éstas. Normalmente las ventanas con un mismo color, significa que están relacionadas entre sí.

A continuación veremos un esquema de cómo queda nuestro escritorio en cuanto a relación entre sus ventanas. También vemos la relación con las variables globales.

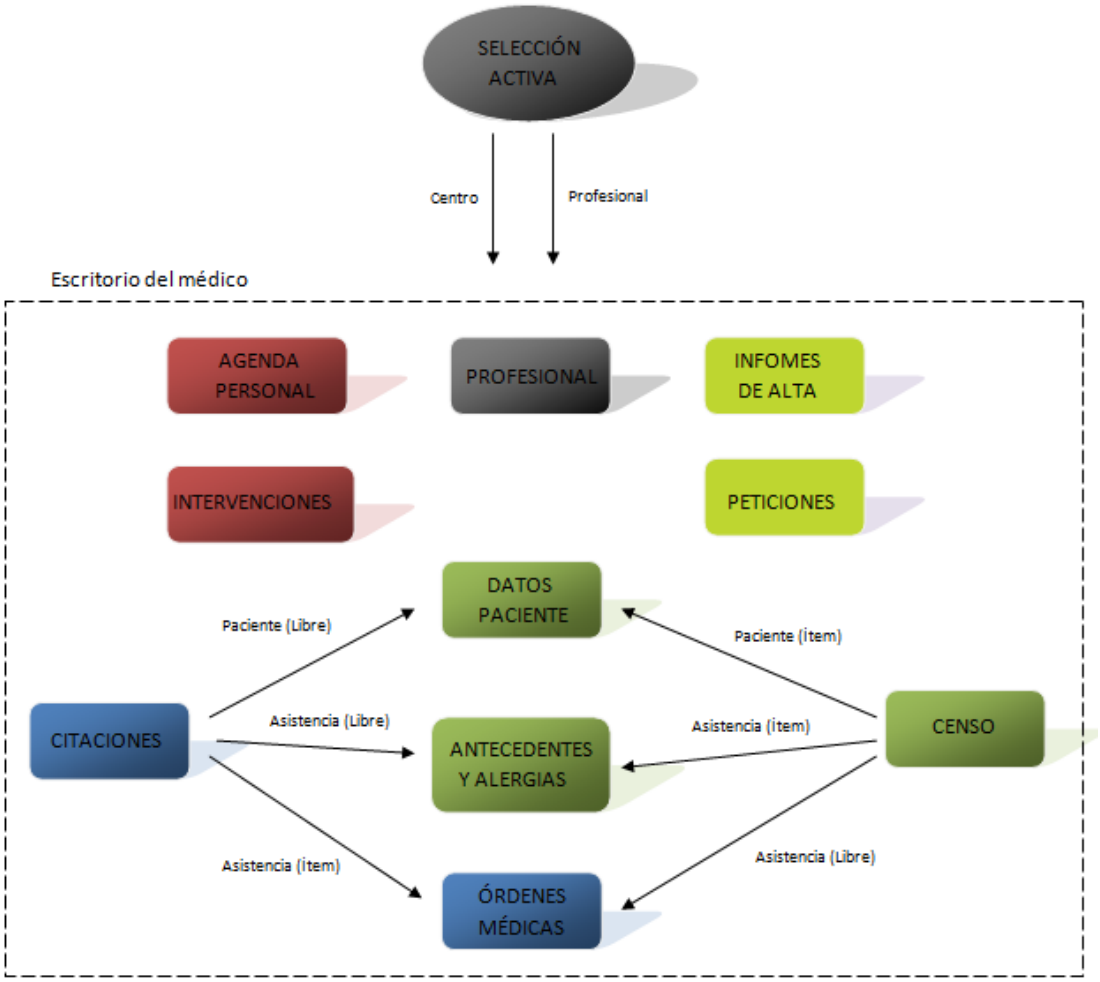

Figura 40: Esquema de la relación entre ventanas en el escritorio

Como hemos explicado en el párrafo anterior, en esta figura podemos observar cómo las ventanas que reciben dos parámetros de entrada iguales han de alternar el modo de ser enviados (ítem o Libre). En la figura 41, podemos ver cómo queda nuestro escritorio en su totalidad.

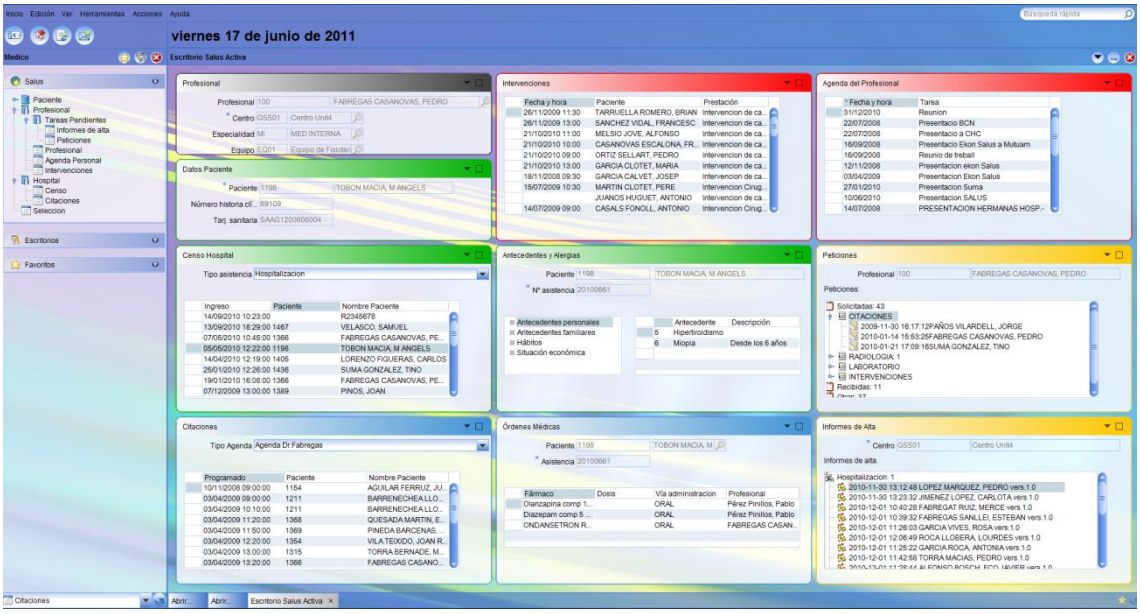

Figura 41: Vista completa y detallada del escritorio de trabajo del médico

5.7. Autoconfiguración del usuario

En este apartado hablaremos un poco de las gestiones que ha de realizar el usuario o profesional para configurarse o adaptarse el escritorio a sus necesidades.

En nuestro escritorio de trabajo del médico hemos creado una serie de ventanas (relacionadas con un formulario creado previamente). El usuario puede distribuirse las ventanas a través del escritorio de la manera que más le guste o le sea más fácil reconocer. Puede cambiarles el tamaño e incluso el color.

Otra posibilidad que tiene es cambiar las relaciones entre ventanas. Para ello deberá acudir a la diseñadora de mini aplicaciones y añadir parámetros de entrada y salida a éstas para luego poder enlazarlas, aunque para ello necesitará una formación previa por parte del administrador.

En un futuro se pretende crear multitud de ventanas o mini aplicaciones para que el usuario pueda escoger cuales quiere ver en su escritorio.

6. Fase de Pruebas

6.1. Introducción

Para completar el ciclo de desarrollo de una aplicación es necesario garantizar en la medida de lo posible su correcto funcionamiento y la integridad de los datos. Hay muchos factores diferentes a tener en cuenta en el ciclo de desarrollo que minimizan la incidencia de errores y facilitan su detección.

Normalmente, esta fase de pruebas se realiza al finalizar la aplicación, aunque en nuestro caso las pruebas se han ido intercalando a medida que se desarrollaba la aplicación. Se han ido encontrando errores tanto propios como de las herramientas que hemos utilizado a la hora de desarrollar nuestra aplicación. Cabe destacar que para el desarrollo de nuestra aplicación hemos utilizado la herramienta de creación de escritorios de *karat*, que ha sido desarrollada hace poco tiempo y que aún se encuentra en una versión beta. Por lo tanto, han ido surgiendo problemas que han tenido que ser subsanados a medida que avanzaba nuestro proyecto.

A continuación detallaremos las diferentes pruebas que se han realizado y cómo se han llevado a cabo.

6.2. Pruebas Unitarias

Las pruebas unitarias son aquellas que nos permiten asegurar que un determinado componente de la aplicación devuelve una salida correcta por una determinada entrada.

En nuestra aplicación se ha hecho una llamada a cada una de las acciones de los módulos varias veces, modificando los valores de entrada y comprobando que los valores de salida eran los esperados.

Las diferentes pruebas que se han realizado en todos nuestros formularios dentro de esta categoría son las siguientes:

- Validación de los campos obligatorios: Comprobar que en todos los formularios con campos de entrada obligatoria no permiten continuar sin la complementación de estos campos. Se comprueba que la acción proporciona el correspondiente mensaje de error.
- Validación del formato correcto de los datos de entrada: Comprobar que en cada campo únicamente se permite introducir datos con el formato que pide. Esto se ha conseguido mediante máscaras de entrada y salida.

- Validación de la información actualizada en la base de datos: Para cada acción que actualiza información en la base de datos se ha comprobado que esta información era la esperada.
- Comprobación de los resultados a consultas de datos: Se ha comprobado que cualquier consulta a la base de datos devolvía los valores esperados.

6.3. Pruebas Funcionales

Las pruebas funcionales son aquellas que nos permiten asegurar y comprobar que los diferentes componentes de nuestra aplicación realizan las funciones que se requieren.

A continuación veremos las pruebas funcionales en las diferentes ventanas de nuestro escritorio:

- **Ventana de Antecedentes y Alergias:**
  - Se ha comprobado, en primer lugar, que si no están informados los campos de asistencia y paciente, el usuario no puede añadir ningún antecedente o alergia. Se le informa mediante un mensaje de error.
  - Se ha comprobado que al informar los campos mencionados anteriormente, se cargan el resto de datos de la cabecera y los datos de los grids de antecedentes y alergias filtrados por estos campos de manera correcta.
  - Al añadir una alergia o antecedente a un paciente, se controla que los campos obligatorios sean rellenados. Si no lo están se muestra un mensaje de error. Una vez comprobados estos campos, se comprueba que han sido añadidos correctamente a la base de datos. Se muestra un mensaje de inserción correcta en la base de datos. Inmediatamente se muestran en el grid correspondiente.
- **Ventana de Órdenes Médicas:**
  - En primer lugar se comprueba que si no están informados los campos de asistencia y centro, el usuario no puede añadir ninguna orden médica. Se le informa mediante un mensaje de error.
  - Una vez rellenados estos campos se comprueba que se cargan el resto de datos de la cabecera y los datos del grid de órdenes médicas filtradas por estos campos de manera correcta.
  - Al añadir una línea de orden médica a un paciente, se ha de controlar que exista previamente una orden médica a la que añadir dicha línea. Si no existe, se añade. Al añadir los datos de la línea de orden médica, se comprueba que los campos obligatorios son rellenados, sinó se muestra un mensaje de error. Una vez añadidos, se muestra un mensaje de inserción

correcta en la base de datos. Inmediatamente se muestran en el grid correspondiente.

- **Ventana de Agenda Personal:**
  - Comprobamos que al tener informado un profesional, se cargan todas sus tareas en el grid correspondiente. Por lo contrario, si no tenemos ningún profesional informado el grid estará vacio.
  - Al querer filtrar estas tareas mediante los campos de fecha disponibles, primero se comprueba el correcto formato de estas fechas mediante máscaras de entrada, y en segundo lugar, que los datos mostrados a continuación en el grid se hayan actualizado correctamente.
  - Al añadir una nueva tarea, se controla que los campos obligatorios sean rellenados. Si no lo están se muestra un mensaje de error. Una vez comprobados estos campos, se comprueba que han sido añadidos correctamente en la base de datos. Se muestra un mensaje de inserción correcta en la base de datos. Inmediatamente se muestran en el grid correspondiente.
- **Ventana de Intervenciones:**
  - Comprobamos que al tener informado un profesional, se cargan todas sus tareas en el grid correspondiente. Por lo contrario, si no tenemos ningún profesional informado el grid estará vacio.
  - Al querer filtrar estas tareas mediante los campos de fecha disponibles, primero se comprueba el correcto formato de estas fechas mediante máscaras de entrada, y en segundo lugar, que los datos mostrados a continuación en el grid se hayan actualizado correctamente.
- **Ventana de Peticiones:**
  - Comprobamos que al tener informado un profesional, se cargan las diferentes peticiones en el árbol correspondiente. Por lo contrario, si no tenemos ningún profesional informado el árbol estará vacio.
  - Se comprueba que al seleccionar una petición del árbol, se cargan y muestran los datos de éste en los campos de texto correspondientes.
- **Ventana de Informes de Alta:**
  - Comprobamos que si tenemos un centro informado, se cargan los diferentes informes de alta en el árbol correspondiente. Por lo contrario, si no tenemos ningún centro informado el árbol estará vacio.
  - Se comprueba que al seleccionar un informe de alta del árbol, se cargan y muestran los datos de éste en los campos de texto correspondientes.
  - Al intentar filtrar las consultas por fecha, se comprueba que el formato introducido en el campo de fecha es el correcto, mediante una máscara de entrada. Una vez introducida comprobamos que la información del árbol vuelve a ser cargada filtrada por esta fecha.

- **Ventana de Censo del Hospital:**
  - Se comprueba que al tener informado un profesional y un centro de trabajo, se carga y se forma la lista desplegable de los diferentes tipos de asistencia. Por lo contrario, si no tenemos ningún profesional informado esta lista estará vacía.
  - Al seleccionar un elemento de esta lista, se comprueba que se cargan lo datos correspondientes a los pacientes censados en los grids correspondientes filtrados por el elemento seleccionado.
  - En la pestaña de lista de espera, se comprueba que al seleccionar los diferentes checks disponibles la información del grid va cambiando correctamente.
- **Ventana de Citaciones:**
  - Se comprueba que al tener informado un profesional y un centro de trabajo, se carga y se forma la lista desplegable de los diferentes tipos de agendas. Por lo contrario, si no tenemos ningún profesional informado esta lista estará vacía.
  - Al seleccionar un elemento de esta lista, se comprueba que se cargan lo datos correspondientes a los pacientes citados en los grids filtrados por el elemento seleccionado.
  - En la pestaña de visitas del día comprobamos que al seleccionar una fecha, los datos del grid se filtran correctamente.
  - En la pestaña de lista de espera, se comprueba que al seleccionar los diferentes checks disponibles la información del grid va cambiando correctamente.
- **Ventana Datos Paciente:**
  - Al informar el número de expediente de un paciente se comprueba que se carguen correctamente todos sus datos en los campos correspondientes filtrados por éste expediente.
- **Ventana de Selección Activa:**
  - Se comprueba que al informar los diversos campos, éstos son guardados en las variables de entorno correctamente.
- **Ventana de Historial de Asistencias:**
  - Se comprueba que al informar el campo del paciente se cargan en el grid correspondiente todos los datos de asistencias filtrados por este campo.

6.4. Pruebas de Integración

Las pruebas de integración tienen como finalidad probar el funcionamiento conjunto de la aplicación. Esto quiere decir asegurar que los módulos y acciones

probadas en el punto anterior funcionan correctamente de forma conjunta. Los procesos verificados se detallan en los siguientes puntos:

- Prueba 1: Variables de entorno.
  - Se ha comprobado que al seleccionar los diferentes datos del formulario de selección activa éstos son pasados correctamente, a través de las variables de entorno, al resto de ventanas del escritorio para filtrar sus datos.

- Prueba 2: Relación entre ventanas.
  - Se ha comprobado que al seleccionar una fila de los grids de Censo del Hospital o Citaciones, pasamos los diferentes parámetros de entrada a las ventanas de Datos de Paciente, Antecedentes y Alergias y Órdenes Médicas, dependiendo de cuál sea la ventana seleccionada.

7. Conclusiones

7.1. Objetivos adquiridos

Una vez llegados a este punto, es el momento de hacer balance y revisar si la aplicación desarrollada cubre los objetivos inicialmente fijados. A grandes rasgos se puede afirmar que el producto resultante de este proyecto es una solución que encaja perfectamente con los objetivos que se enumeraron durante la definición del proyecto.

Nuestro escritorio de trabajo del médico permite, de manera visual, realizar al profesional la mayoría de las actividades necesarias en su día a día en su puesto de trabajo. Además la aplicación se ha desarrollado de manera que el usuario se sienta cómodo utilizándola y entienda rápidamente todas las funcionalidades. Este era uno de los principales objetivos ya que esta aplicación está pensada para profesionales del sector sanitario los cuales no tienen por qué tener unos conocimientos avanzados en informática.

Finalmente hay que destacar que la aplicación se ha desarrollado siguiendo el programa de *karat* y su nueva interfaz *Walnut*, y de manera que se pueda adaptar perfectamente al producto *ekon Salus*.

7.2. Desviaciones de la planificación

Podemos decir, que la planificación inicial de nuestro proyecto se ha seguido casi en su totalidad. Todos los requerimientos iniciales se han completado en mayor o menor medida.

De todas maneras, algunos de nuestros requerimientos han sufrido, a medida que avanzaba el proyecto, recortes en su funcionalidad o complejidad. Esto ha sido debido a que nuestra incorporación a la empresa fue en noviembre, fecha a partir de la cual empezamos con los análisis de la aplicación actual así como de la que queríamos proponer, pero dependíamos de la realización, por parte del Departamento de *platform*, de la herramienta de creación de escritorios en la que habíamos de implementar nuestra aplicación. Esta aplicación nos fue facilitada a finales de febrero, por lo cual nos pasamos bastante tiempo analizando y diseñando sobre cómo hacer nuestro escritorio sin saber a ciencia cierta si se podría llevar a cabo todo en dicha herramienta.

Por ello, una vez proporcionada esta herramienta y estando ya en la fase de implementación, tuvimos que volver en muchas ocasiones hacia atrás, hablar con

nuestro tutor y diseñar las cosas de diferente manera, pues la herramienta no permitía algunas de nuestras ideas.

Al final, todo lo previsto se ha podido realizar, pero con esto quería reseñar que nuestra planificación fue variando a medida que avanzaba la implementación de nuestra aplicación.

La siguiente tabla muestra las modificaciones temporales que se han producido respecto a la planificación inicial.

Nombre de tarea	Duración planificada	Duración real
Investigación y formación sobre el escritorio del médico actual	7días	12 días
Investigación y formación sobre el escritorio del médico actual	20 días	19 días
Diseño de la Aplicación	34 días	28 días
Desarrollo de la Aplicación	46 días	56 días
Test y pruebas	17 días	12 días
Generación de la Documentación	5 días	10 días

Tabla 10: Desviación de la planificación inicial

7.3. Ampliaciones

Como se ha explicado con anterioridad durante los diferentes apartados de la memoria, nuestra aplicación, el escritorio de trabajo del médico, forma parte del extenso producto *ekon Salus.* El escritorio es una puerta de acceso sencilla y visual a este producto, donde se realiza la mayor parte de las funciones asistenciales necesarias.

La idea es poder, en un futuro cuando el producto esté completamente migrado a Java, enlazarlo con nuestro escritorio y así poder aumentar las prestaciones de éste, accediendo a los módulos de dicho producto a través de nuestro escritorio y disponiendo de todas sus funcionalidades.

Cuando esto esté hecho podrá implementarse un acceso a la aplicación más sofisticado, siguiendo la configuración de los empleados o profesionales.

Otras posibles ampliaciones vienen determinadas por las posibles mejoras de la herramienta de creación de escritorios de *karat.* En este momento es una versión beta producida hace poco tiempo y que necesita ser mejorada y ampliada. A partir de aquí nuestra aplicación podría incorporar nuevas funcionalidades.

Un ejemplo de posible mejora de la herramienta, es la posibilidad de cambiar la propiedad del color de las ventanas de manera dinámica. Es decir, las ventanas pueden tener un color y unos parámetros de entrada y salida. Normalmente, como se ha explicado en la Implementación, las ventanas enlazadas por parámetros comparten el mismo color. Pero cabe la posibilidad, como en nuestro caso, que una ventana reciba parámetros de diversas ventanas que no tienen mucho en común, Censo y Citaciones, según cual tengamos seleccionada. A partir de esta propiedad (de cambio de color) según cuál de éstas seleccionemos y enviemos el parámetro a nuestra ventana, éste recibiría también su color. De esta manera quedaría más claro de que ventana está recibiendo el valor.

7.4. Valoración personal

El desarrollo de este proyecto me ha parecido una experiencia muy enriquecedora por varios motivos. Uno de ellos, ha sido poder aprender en profundidad un lenguaje de programación como Java y poder trabajar con él en un proyecto de gran alcance. Hasta ahora, en la universidad, habíamos aprendido diversos lenguajes de programación pero no habíamos entrado en gran profundidad en ninguno de ellos.

También ha sido importante el hecho de poder desarrollar una aplicación desde cero y realizar un ciclo completo en el desarrollo de software. De esta manera, he podido darme cuenta de la dificultad, trabajo y dedicación que conlleva un proyecto de esta tamaño y envergadura. Aunque el trabajo es bien compensado con la satisfacción que produce ver los resultados que uno va viendo.

En relación al proyecto, al principio me resulto un poco complejo y difícil familiarizarme con los conceptos médicos y sanitarios necesarios para poder desarrollar la aplicación. A medida que pasaba el tiempo adquiría mayor conocimiento y al final pude realizarlo todo sin demasiadas complicaciones.

Cabe decir, que estoy contento de haber podido realizar este convenio con la empresa UNIT4. Esto me ha dado la oportunidad de realizar el proyecto en una gran empresa, saber cómo funciona, adaptarme a su metodología de trabajo y darme cuenta de cómo funciona el mundo de la informática a nivel laboral.

Por todos estos motivos, y a pesar que el número de horas realizadas posiblemente haya sido superior a los proyectos convencionales, mi valoración personal del proyecto es muy positiva y gratificante. Estoy muy contento de haber tomado esta decisión.

Bibliografía

A continuación se detallan las referencias bibliográficas que se han usado para realizar el proyecto, se han incluido las más usadas pero cabe indicar, que la mayor parte de documentación para la realización de nuestra aplicación se ha encontrado en documentos internos de la empresa y del producto *karat.*

- Documentación de la herramienta de creación de escritorios.

- Documentación karat.

- http://es.wikipedia.org
  Es una enciclopedia libre y políglota de la Fundación Wikimedia (una organización sin ánimo de lucro).

- http://www.java2s.com/
  Web muy completa sobre tutorías y ejemplos en Java.

Daniel Caliz Vilchez
Sabadell, 23 de Junio de 2011