Facturación de servicios

Memoria del proyecto
de Ingeniería Técnica en
Informática de Gestión
realizado por
David Manzano Barba
y dirigido por
Jordi Pons Aróztegui

Escola d'Enginyeria

Sabadell, junio de 2012

El abajo firmante, **Jordi Pons Aróztegui**,
profesor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria
ha sido realizado bajo su dirección por
David Manzano Barba
Y para que conste firma la presente.
Sabadell, **junio** de **2012**

Firma: **Jordi Pons Aróztegui**

El abajo firmante, **Ramon Torres Arús**,
de **UNIT4**,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria
ha sido realizado bajo su supervisión por
David Manzano Barba
Y para que conste la firma presente.
Sabadell, **junio** de **2012**

Firma: **Ramon Torres Arús**

HOJA DE RESUMEN – PROYECTO FIN DE CARRERA DE L'ESCOLA D'ENGINYERIA

Título del proyecto: Facturación de servicios	
Autor: David Manzano Barba	**Fecha:** Junio de 2012
Tutor: Jordi Pons Aróztegui, Ramon Torres Arús	
Titulación: Ingeniería Técnica en Informática de Gestión	
Palabras clave (mínimo 3) • Castellano: facturación, gestión, servicios, albaranes. • Catalán: facturació, gestió, serveis, albarans. • Inglés: invoicing, management, services, bill.	
Resumen del proyecto (extensión máxima 100 palabras) • Castellano: Este proyecto se ha realizado mediante un convenio de prácticas con la empresa UNIT4 (Barberà del Vallès). Se basa en el desarrollo de un nuevo módulo enfocado hacia compañías distribuidoras de productos, que será incorporado en el software de gestión que vende la empresa. Este módulo consistirá en facturar algunos servicios ofrecidos en los envíos de paquetes, que se incluirán en el albarán de entrega de los mismos. • Catalán: Aquest projecte s'ha realitzat mitjançant un conveni de pràctiques amb l'empresa UNIT4 (Barberà del Vallès). Es basa en el desenvolupament d'un nou mòdul enfocat cap a companyies distribuïdores de productes, que serà incorporat en el software de gestió que ven l'empresa. Aquest mòdul consistirà en facturar alguns serveis oferts en els enviaments de paquets, que s'inclouran en l'albarà de lliurament. • Inglés: This project was realized using a working agreement with the company UNIT4 (Barberà del Vallès). It is based on developing a new module for distribution companies that will be incorporated into the management software that sells UNIT4. This module will be to invoice some services offered in packet delivery to be included in the bill of these.	

Tabla de contenido

Índice de tablas

Índice de figuras

Introducción

1.1 Marco de trabajo

1.1.1 Convenio de colaboración entre la UAB y UNIT4

El siguiente proyecto final de carrera se ha realizado en colaboración con UNIT4, siguiendo el convenio estipulado entre la Universitat Autònoma de Barcelona y la empresa. De esta forma el alumno puede empezar a introducirse en el sector laboral del desarrollo de software, en este caso software ERP.

El tiempo acordado para la elaboración del proyecto son 560 horas, desde noviembre de 2011 hasta junio de 2012, y éste se desarrollará en el departamento de R+D (Research and Development) de las oficinas de Barberà del Vallès (Barcelona).

1.1.2 La empresa

UNIT4 tiene una experiencia durante más de cuarenta años en el mundo de la ingeniería del 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.

La empresa dispone actualmente de 4.230 trabajadores y cuenta 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.

Como principales objetivos tienen mantener su posición de liderazgo como proveedor global de soluciones y servicios TIC y estar por delante en la investigación de nuevas tecnologías y desarrollo de nuevas soluciones.

Hablando del proyecto, éste se realizará dentro del producto *ekon* (ERP), sostenido por la plataforma karat para su desarrollo y en modo gráfico *Walnut*. A continuación, una breve explicación de cada uno de ellos.

UNIT4 Ibérica desarrolla y ofrece la gama de soluciones *ekon*, tanto ERP como para la gestión del negocio, que se adaptan especialmente a las organizaciones que desarrollan su actividad en un clima de cambios frecuentes y dinámicos (especialmente las centradas en servicios o las del sector público). *Ekon* se trata de una suite ERP totalmente integrada, ágil, transparente y orientada a

resultados. Entre sus principales funcionalidades destacan: *Finanzas*, *Logística*, *Proyectos*, *RRHH*, *CRM* y *Análisis*.

La empresa aplica a sus soluciones la tecnología karat, una completa plataforma tecnológica para la gestión de las empresas, que aporta un nuevo concepto de soluciones basado en la independencia total y real de entornos. 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.

Hablando del proyecto tecnológico *Walnut*, éste ofrece al usuario una experiencia en el manejo de las herramientas *ekon* mucho más satisfactoria y gratificante, repercutiendo directamente en la calidad de su trabajo diario.

Concluyendo, la misión de UNIT4 es proporcionar una auténtica ventaja competitiva a sus clientes mediante la implantación de soluciones informáticas de gestión.

1.2 Objetivos del proyecto

La finalidad del proyecto es crear un nuevo módulo llamado “Facturación de servicios”. Éste irá instalado en el apartado de logística de cualquier empresa distribuidora.

Estas empresas ofrecen unos servicios específicos en cada envío de paquetes, como por ejemplo, el servicio de “Urgencias”, el servicio de “Envío de paquetes frágiles”, etc. Éstos se incluirán por albarán de entrega, pudiéndose incluir más de uno en cada pedido.

La herramienta estará formada por distintos formularios de inserción y mantenimiento de datos, también por una generación de consumos, que añadirá servicios a los albaranes y calculará el importe a facturar por cliente según unos parámetros de entrada especificados.

Inicialmente, el módulo se ha desarrollado en ekon NDISFAR (en modo gráfico ekon 6, *Walnut*), aplicación que referencia a una empresa de distribución farmacéutica que ha demandado este módulo. Hablando del entorno de ejecución, este proyecto se creará bajo karat 8, plataforma tecnológica idónea para el desarrollo, implantación, mantenimiento y evolución continua de aplicaciones.

A continuación enumeraré todos los objetivos a cumplir en este trabajo, tantos los objetivos

marcados por la empresa, como mis objetivos a nivel personal, indicando qué beneficios puedo obtener con el desarrollo del proyecto.

Los objetivos marcados por la empresa son:

- Gestionar de una forma más detallada este nuevo módulo.
- Conseguir una herramienta altamente intuitiva, fácil de manipular para cualquier usuario.
- Conseguir una aplicación fácilmente accesible, con vistas al mantenimiento de tablas.
- Adaptarlo de la mejor forma posible a la aplicación *ekon*.

A nivel personal, mis objetivos son:

- Mejorar mi nivel de lenguaje en Java y SQL.
- Adquirir experiencia en una empresa multinacional de desarrollo de software.
- Aprender qué es y cómo funciona uno de los ERP más conocidos a nivel internacional.
- Conseguir desarrollar un proyecto de principio a fin, pasando por las diferentes etapas y actividades que éste conlleva.

1.3 Contenido de la memoria

La memoria del proyecto constará de las siguientes partes:

- **Introducción**: Se expone dónde y bajo qué condiciones se va a desarrollar el proyecto, los objetivos marcados.
- **Estudio de viabilidad**: Apartado dónde se presentarán los objetivos, situación actual, requisitos funcionales y no funcionales, alternativas y finalmente conclusiones acerca de los beneficios o pérdidas.
- **Plan de proyecto**: Se hace una planificación de las tareas y recursos necesarios para el desarrollo del proyecto, también se evalúan los riesgos y el presupuesto estimado.
- **Análisis y diseño**: Aquí se examinarán las funcionalidades del sistema, la arquitectura del sistema y un diagrama de la base de datos.
- **Desarrollo**: Punto donde se describirá la tecnología utilizada en la implementación.
- **Codificación y pruebas**: Se demostrarán las pruebas unitarias, funcionales, integrales y de rendimiento. También se enseñará cuál ha sido el estilo de codificación utilizado.
- **Conclusiones**: Se discutirán los objetivos conseguidos del proyecto, posibles ampliaciones y una valoración personal acerca del mismo.
- **Bibliografía**: Detalle de las fuentes utilizadas para el posible desarrollo del módulo.
- **Contenido del CD**: Se podrá ver de qué archivos estará compuesto el CD entregado junto con esta memoria.
- **Agradecimientos**: Último apartado, donde se reconocerá toda la ayuda recibida para la elaboración del proyecto.

Estudio de viabilidad

2.1 Introducción

2.1.1 Objetivos del proyecto

Los objetivos a cumplir del módulo "Facturación de servicios" son los siguientes:

- **Objetivo 1**: Inserción y mantenimiento de servicios en la base de datos.
- **Objetivo 2**: Relacionar clientes con determinados servicios.
- **Objetivo 3**: Establecer datos generales por empresa y sección.
- **Objetivo 4**: Generación de consumos de los servicios seleccionados.
- **Objetivo 5**: Mantenimiento de los consumos generados.
- **Objetivo 6**: Generación de listados de los consumos anteriores.
- **Objetivo 7**: Cargo de servicios en albaranes (fuera del proyecto).

Este último objetivo quedó fuera del proyecto porque necesitaba de otro módulo que todavía está desarrollándose. Por motivos cronológicos este proyecto finalizó antes que dicho módulo en construcción, por lo que se desestimó el cumplimiento de este objetivo.

2.1.2 Catalogación y priorización de los objetivos

A continuación, una tabla con la prioridad de cada objetivo analizando si debe ser crítico, prioritario o secundario.

	Crítico	Prioritario	Secundario
Objetivo 1	**X**		
Objetivo 2	**X**		
Objetivo 3	**X**		
Objetivo 4	**X**		
Objetivo 5		**X**	
Objetivo 6			**X**

Tabla 1: Catalogación y priorización de los objetivos

2.1.3 Partes interesadas del proyecto

A continuación una tabla donde figuran los *stakeholders* y el equipo de proyecto. Los *stakeholders* son las unidades organizativas que afectan o son afectadas por el proyecto, ya sea de forma positiva o negativa. Una buena planificación de proyectos debe involucrar la identificación y clasificación de los interesados, así como el estudio y la determinación de sus necesidades y expectativas. Mientras que el equipo de proyecto, como su propio nombre indica, lo forma la composición de personas que participarán en el desarrollo del mismo.

NOMBRE	DESCRIPCIÓN	RESPONSABILIDAD
Stakeholders		
Departamento R&D, UNIT4, Barberà del Vallès.	Departamento de la empresa dedicado al desarrollo y mantenimiento de software.	Desarrollo y mantenimiento de software.
Equipo de proyecto		
Jordi Pons Aróztegui	Tutor del proyecto (UAB)	Supervisa el trabajo del alumno, le ayuda y lo evalúa.
Ramon Torres Arús	Tutor del proyecto (UNIT4)	Supervisa el trabajo del becario y lo evalúa.
Francisco Javier Marina Angulo	Jefe de proyecto (UNIT4)	Hace un seguimiento de todo el proyecto y define su aprobación.
Marc Bardina Gutiérrez	Analista	Asesoramiento en el análisis de la aplicación y asesoramiento técnico a lo largo del proyecto.
David Manzano Barba	Analista	Realiza el estudio de viabilidad y la planificación. Analiza los requisitos de la aplicación y también participa en el diseño y la validación.
David Manzano Barba	Programador	Desarrolla la aplicación de acuerdo con el análisis y la planificación prevista. Participa en el proceso de validación e implantación.
David Manzano Barba	Diseñador	Diseña la arquitectura de la aplicación, tanto la interfaz gráfica, como la estructura de la base de datos.

Tabla 2: Partes interesadas del proyecto

2.2 Estudio de la situación actual

2.2.1 Contexto

Actualmente, la empresa no dispone de un módulo de facturación de servicios para ninguno de sus productos. Esta necesidad surge cuando UNIT4 vende su software ERP a una distribuidora farmacéutica y se dispone a adaptar la nueva tecnología *ekon* a esta empresa. La distribuidora en cuestión reclama una herramienta capaz de facturar servicios, ya que disponía de ella en su anterior sistema de gestión.
Se integrará esta herramienta en el producto de logística del software *ekon*, para otras posibles empresas distribuidoras que deseen utilizarla.

2.2.2 Descripción física

El nuevo módulo se desarrollará en ekon Distribución farmacéutica, en modo gráfico ekon 6 (Java, *Walnut*). Únicamente es necesario tener instalado la consola de Java, ya que ekon puede ser ejecutado en cualquier sistema operativo, ya sea Windows, Mac OS X o cualquier GNU de Linux.
Hablando de bases de datos, *ekon* acepta cualquier sistema de almacenamiento: MySQL, Oracle, Cache Telnet, etc. Y finalmente, para el desarrollo del producto, también se necesitará de la plataforma karat.
En el caso de este proyecto, se utilizará *ekon* bajo un sistema operativo Windows (versión 7) y una base de datos Oracle.

2.2.3 Usuarios y/o personal del sistema

Esta aplicación va dirigida a usuarios expertos y no expertos de la unidad de logística de la empresa. El tipo de usuario cambiará dependiendo de la parte del módulo a la que se acceda.
El usuario no experto se encargará del mantenimiento de la información en la base de datos, mientras que el experto facturará los servicios, tarea de alta responsabilidad ya que se puede estar operando con mucho efectivo. De todas formas se desea conseguir una herramienta altamente intuitiva para que ésta sea fácil de manipular para cualquier usuario.

2.2.4 Diagnóstico del sistema

Las carencias del sistema que utilizaba la distribuidora farmacéutica, son las mejoras que se incluirán en el nuevo módulo. Éstas son:

- Relacionar clientes con determinados servicios de albaranes.
- Mantenimiento de los consumos generados por la aplicación.
- Generación de listados de los consumos anteriores.

2.3 Requisitos

2.3.1 Requisitos funcionales

Los requisitos funcionales en este caso son secciones o partes de los objetivos principales marcados para el proyecto:

Objetivo 1: Inserción y mantenimiento de servicios en la base de datos.
La finalidad de este objetivo es poder crear un nuevo servicio, o de los ya creados tener acceso a su modificación en la base de datos.

- **RF1**. Crear un formulario de mantenimiento llamado “Servicios”.
- **RF2**. Definir tipos de cliente a tratar del servicio.
- **RF3**. Definir intervalos horarios del servicio.
- **RF4**. Definir series de albarán a no tratar del servicio.
- **RF5**. Definir documentos de gestión a tratar del servicio.
- **RF6**. Definir subtipos de gestión a tratar o a no tratar del servicio

Objetivo 2: Relacionar clientes con determinados servicios.
Ofrecer la posibilidad de definir algunos parámetros a nivel de cliente o a nivel de cliente por servicio.

- **RF7**. Crear un formulario de mantenimiento llamado “Servicios por cliente”.
- **RF8**. Definir un importe exento por cliente.
- **RF9**. Definir si el cliente está exento de todos los servicios.
- **RF10**. Definir un importe exento por cliente y servicio.
- **RF11**. Definir si el cliente está exento de un servicio en concreto.

Objetivo 3: Establecer datos generales por empresa y sección.
La intención de este objetivo es implantar las características genéricas de los servicios por empresa. Así se evitará definir estos datos a nivel de servicio, ya que se cogerá la información de la empresa activa seleccionada. También se podrán establecer según que propiedades por sección de la empresa.

- **RF12**. Crear un formulario de mantenimiento llamado “Empresa y sección”.
- **RF13**. Definir series de albarán a no tratar de la empresa activa.
- **RF14**. Definir secciones a tratar de la empresa y el posible importe exento de cada una.

Objetivo 4: Generación de consumos de los servicios seleccionados.
Se pretende crear un consumo a partir de los servicios que se hayan seleccionado y el intervalo de fechas y clientes establecido. El resultado final será un importe a facturar por cliente.

- **RF15**. Crear un formulario de diálogo llamado “Generación de consumos”.
- **RF16**. Generar consumos entre un intervalo de fechas y clientes.
- **RF17**. Generar consumos para todos los clientes.
- **RF18**. Generar consumos para un determinado número de servicios.

Objetivo 5: Mantenimiento de los consumos generados.
A partir de la generación anterior, aquí se mostrarán los resultados obtenidos, pudiendo modificar según que parámetros a gusto del cliente.

- **RF19**. Crear un formulario de mantenimiento llamado “Consumos”.
- **RF20**. Cálculo de algunos campos del formulario en tiempo real.

Objetivo 6: Generación de listados de los consumos anteriores.
Hacer un listado consiste en formar una vista de impresión de cualquier formulario. Se creará un listado del formulario de mantenimiento anterior de consumos, éste se llamará “Listado de consumos”.

- **RF21**. Mostrar en vista de impresión los consumos generados.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales especifican los criterios que se deben validar para una correcta actividad del sistema, sin tener en cuenta las funcionalidades desarrolladas en el módulo. Estos requisitos son los siguientes:

- **RNF1**. El proyecto tendrá que estar correctamente documentado.
- **RNF2**. La aplicación tendrá que seguir las normativas marcadas por la empresa.
- **RNF3**. Los recursos utilizados por la herramienta tendrán que estar a medida de la entidad.
- **RNF4**. El módulo ha de ser ampliable para poder instalar nuevas mejoras.
- **RNF5**. El diseño ha de seguir las directrices de *Walnut*.

2.3.3 Restricciones del sistema

Las restricciones de sistema son aquellas limitaciones derivadas de las tecnologías que se disponen para que el software de la herramienta se ejecute sin problemas. Estos requisitos son los siguientes:

- La fecha límite de finalización del proyecto ha de ser el 26 de junio de 2012.
- La duración del proyecto ha de ser de 560 horas como máximo.
- El módulo se ha de integrar correctamente en el producto *ekon*.
- El desarrollo de la aplicación ha de ser en lenguaje Java.

2.3.4 Catalogación y priorización de los requisitos

Clasificación de los requisitos funcionales y no funcionales según su importancia en este proyecto. Estos requisitos pueden ser: esenciales (prácticamente obligatorios), condicionales (relevantes) y opcionales (innecesarios).

Requisitos	Esencial	Condicional	Opcional
RF1	X		
RF2	X		
RF3	X		
RF4	X		
RF5	X		
RF6	X		
RF7	X		
RF8		X	
RF9		X	
RF10	X		
RF11	X		
RF12	X		
RF13	X		
RF14	X		
RF15	X		
RF16	X		
RF17		X	
RF18	X		
RF19	X		
RF20		X	
RF21			X
RNF1	X		
RNF2	X		
RNF3	X		
RNF4		X	
RNF5	X		

Tabla 3: Catalogación y priorización de los requisitos

2.4 Alternativas a la solución

Nos encontramos con un nuevo módulo del producto de logística con funcionalidades muy específicas, su definición es exacta, ya que anteriormente la empresa había buscado otras alternativas. Estas fueron analizadas y finalmente se decidieron por el desarrollo de esta aplicación.

2.5 Conclusiones

Como conclusión del estudio de viabilidad, un análisis de los beneficios y las desventajas del proyecto:

Beneficios

- Gestión más eficaz de este nuevo módulo.
- Herramienta altamente intuitiva, fácil de manipular para cualquier usuario.
- Aplicación fácilmente accesible, con vistas al mantenimiento de tablas.
- Posibilidad de ampliar el módulo para introducir nuevas mejoras.

Desventajas

- La ejecución de la generación de consumos es relativamente más lenta en *ekon* que en el anterior sistema de gestión de la distribuidora farmacéutica.
- Necesita de un usuario experto para ejecutar la generación de consumos.

Concluyendo el estudio de viabilidad, podemos constatar que los beneficios superan los inconvenientes, por lo que consideramos que el proyecto es **VIABLE**.

Plan de proyecto

3.1 Introducción

En este punto se muestra el conjunto de actividades que permiten desarrollar, ejecutar y controlar el proyecto. Incluye las tareas y los puntos de control, los recursos, el calendario y la evaluación de riesgos y el presupuesto del proyecto.

Se han seguido los procesos descritos en el PMBOK (2008) y se ha utilizado una metodología lineal en la elaboración del plan.

3.2 WBS

3.2.1 Fases y actividades del proyecto

Fases en las que se dividirá el proyecto y una breve descripción de éstas:

Fases	Descripción
Iniciación	Definición, asignación y matriculación del proyecto.
Planificación	Estudio de viabilidad y Plan de proyecto.
Análisis	Análisis de requerimientos funcionales y no funcionales.
Diseño	Arquitectura de sistema e Interfaz de usuario.
Desarrollo	Fase de implementación, su entorno y el desarrollo de Java.
Codificación y pruebas	Pruebas unitarias, funcionales y de integración.
Generación de documentos	Manuales de usuario y Memoria del proyecto.
Cierre del proyecto	Aceptación del proyecto por parte del director y cierre.
Defensa del proyecto	Defensa del proyecto ante la empresa y la universidad.

Tabla 4: Fases y actividades del proyecto

3.2.2 Diagrama WBS

Un diagrama WBS (Work Breakdown Structure) es una descomposición jerárquica que recoge los objetivos y tareas del proyecto. Cada nivel descendente representa una definición con un detalle del nivel superior. El diagrama del proyecto es el siguiente:

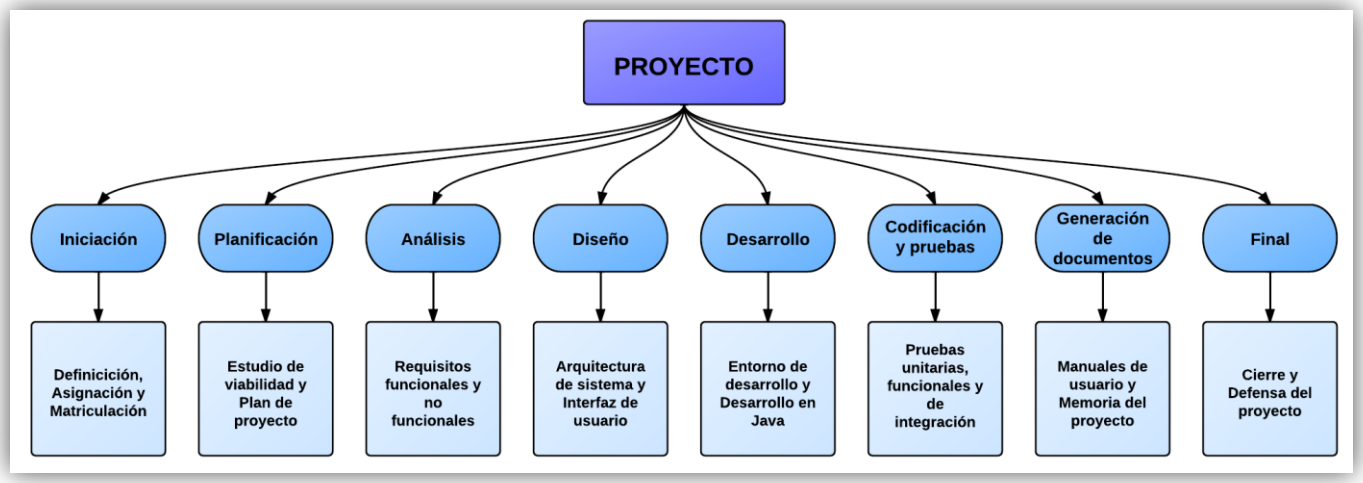

Figura 1: Diagrama WBS

3.2.3 Milestones

Puntos de control del proyecto, descripción de su proceso y fechas en las cuales debe finalizar cada uno de estos.

Fases	Descripción	Fecha
Iniciación	Matriculación, Asignación	14/11/2011
Planificación	Aprobación	24/11/2011
Análisis	Aprobación	07/12/2011
Diseño	Aprobación	10/01/2012
Desarrollo	Aprobación	24/02/2012
Test y pruebas	Aceptación	18/04/2012
Cierre del proyecto	Aceptación	25/06/2012
Defensa del proyecto	Evaluación	26/06/2012

Tabla 5: Milestones del proyecto

3.3 Recursos del proyecto

Recursos necesarios para la elaboración del proyecto, tanto humanos como materiales y también licencias. En la siguiente tabla cada recurso con su respectiva valoración económica.

RECURSOS	VALORACIÓN
Recursos humanos	
Jefe de proyecto	30 €/hora
Analista	15 €/hora
Programador	10 €/hora
Técnico de pruebas	8 €/hora
Recursos materiales	
PC programador	500 €
Licencias	
Karat 7.5.6002	Licencia interna UNIT4 (0€)
Karat 8.0	Licencia interna UNIT4 (0€)
Eclipse IDE for Java Developers	Licencia pública GNU (0€)
Oracle SQL Developer	Licencia OTN (0€)
Notepad++ v5.9.6.2	Licencia pública GNU (0€)
Skype 5.8.0.158	Licencia pública Skype (0€)
Microsoft Office 2010	Licencia de Microsoft Office (450€)

Tabla 6: Recursos del proyecto

3.4 Calendario del proyecto

3.4.1 Dependencias

Todas las fases del proyecto se desarrollan utilizando un modelo lineal, por lo que cada tarea no se empieza hasta que no se ha completado la anterior.

Los apartados de Análisis, Desarrollo y Test, a excepción de las fases anteriores, seguirán un modelo iterativo mejorando así la elaboración del proyecto.

La generación de documentos será la última tarea a realizar antes del cierre y la defensa, ya que contendrá todos los documentos realizados durante el progreso del proyecto.

3.4.2 Cuadro de tareas del proyecto

En este cuadro se informará de las tareas necesarias para el desarrollo del proyecto, la duración de cada una de ellas en días y las fechas de comienzo y fin. También se añaden los recursos humanos necesarios para la elaboración de cada actividad.

Nombre de la tarea	Duración	Comienzo	Finzzzzzzz	Recursos
FACTURACIÓN DE SERVICIOS	**162 días**	**14/11/2011**	**26/06/2012**	
Inicio del proyecto: asignación y matriculación	1 día	14/11/2011	14/11/2011	JP
Cursos de formación	10 días	15/11/2011	28/11/2011	P
Planificación	**6 días**	**29/11/2011**	**06/12/2011**	
Estudio de viabilidad	2 días	29/11/2011	30/11/2011	JP[50%], A[50%]
Aprobación Estudio de viabilidad	1 día	01/12/2011	01/12/2011	JP
Plan del proyecto	2 días	02/11/2011	05/12/2011	JP[50%], A[50%]
Aprobación Plan del proyecto	1 día	06/12/2011	06/12/2011	JP
Análisis	**24 días**	**07/12/2011**	**09/01/2012**	
Reunión con el Jefe de proyecto	1 día	07/12/2011	07/12/2011	JP[50%], A[50%]
Análisis del sistema actual y los requisitos	10 días	08/12/2011	21/12/2011	JP[50%], A[50%]
Análisis de las bases de datos	10 días	22/12/2011	04/01/2012	A
Documentación del análisis	2 días	05/01/2012	06/01/2012	A
Aprobación del análisis	1 día	09/01/2012	09/01/2012	JP
Diseño	**33 días**	**10/01/2012**	**23/02/2012**	
Diseño modular de la aplicación	10 días	10/01/2012	23/01/2012	A[50%], P[50%]
Diseño de la base de datos	10 días	24/01/2012	06/02/2012	A[50%], P[50%]
Diseño de la interfaz	5 días	07/02/2012	13/02/2012	A[50%], P[50%]
Diseño de test y pruebas	5 días	14/02/2012	20/02/2012	A[50%], TP[50%]
Documentación del diseño	2 días	21/02/2012	22/02/2012	A
Aprobación del diseño	1 día	23/02/2012	23/02/2012	JP
Desarrollo	**38 días**	**24/02/2012**	**17/04/2012**	
Preparación del entorno de desarrollo	2 días	24/02/2012	27/02/2012	P
Configuración de las librerías a importar	1 día	28/02/2012	28/02/2012	P
Desarrollo de las clases	5 días	29/02/2012	06/03/2012	P
Desarrollo de las funcionalidades	30 días	07/03/2012	17/04/2012	P
Test y pruebas	**22 días**	**18/04/2012**	**17/05/2012**	
Pruebas unitarias	3 días	18/04/2012	20/04/2012	TP
Pruebas funcionales	3 días	23/04/2012	25/04/2012	TP
Pruebas de integración	3 días	26/04/2012	30/04/2012	TP

Documentación de las pruebas	2 días	01/05/2012	02/05/2012	P[50%], TP[50%]
Aprobación de las pruebas	1 día	03/05/2012	03/05/2012	JP
Corrección de errores	10 días	04/05/2012	17/05/2012	P
Implantación	**6 días**	**18/05/2012**	**25/05/2012**	
Implantación del módulo al producto	5 días	18/05/2012	24/05/2012	A[50%], P[50%]
Generación del archivo JAR	1 día	25/05/2012	25/05/2012	P
Generación de documentos	20 días	28/05/2012	22/06/2012	A[50%], P[50%]
Cierre del proyecto	1 día	25/06/2012	25/06/2012	JP
Defensa del proyecto	1 día	26/06/2012	26/06/2012	JP

Tabla 7: Cuadro de tareas del proyecto

3.4.3 Calendario temporal

Como calendario temporal, a continuación el diagrama de Gantt del proyecto, herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para las tareas o actividades del proyecto.

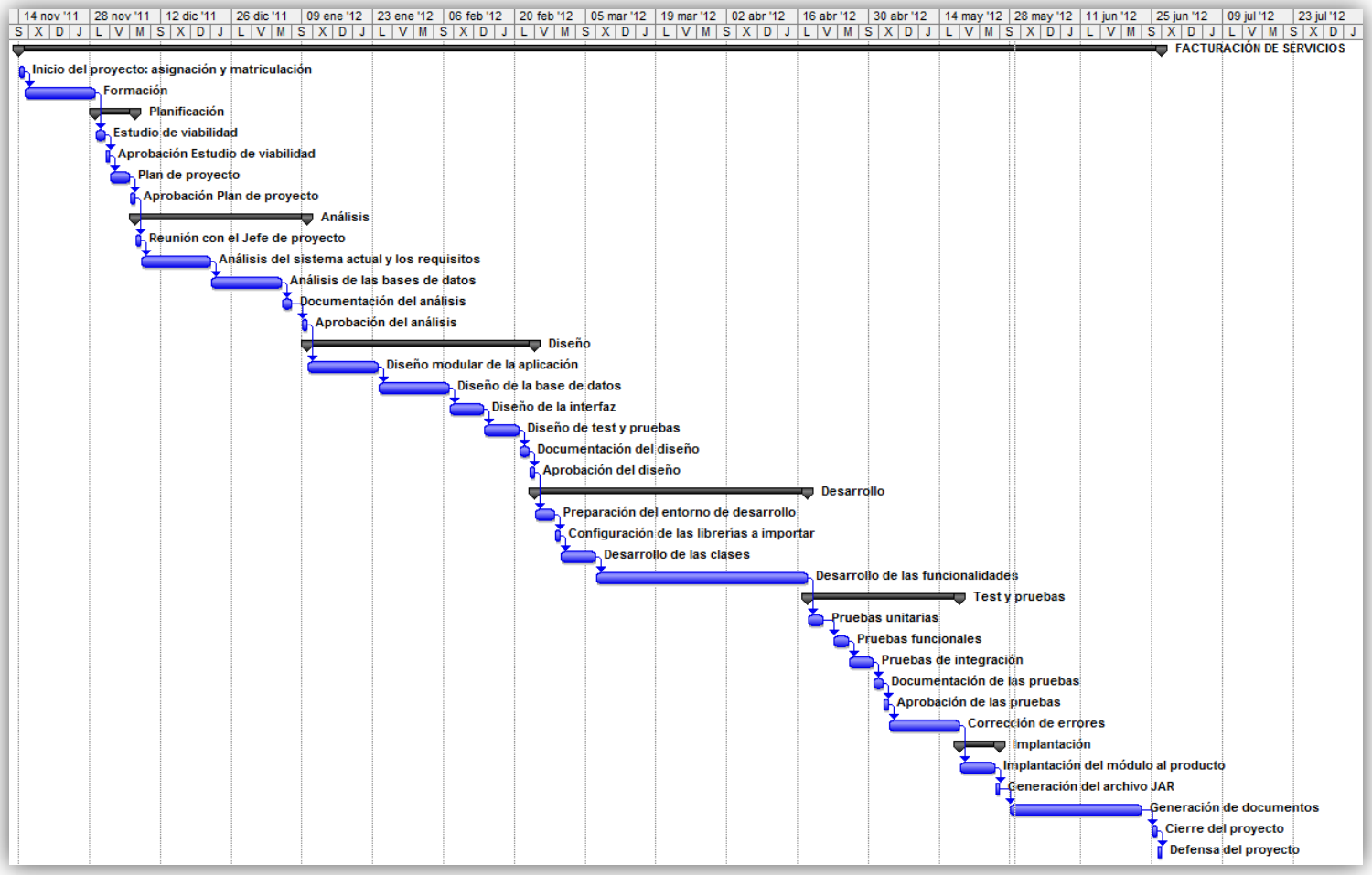

Figura 2: Calendario temporal

3.5 Evaluación de riesgos

3.5.1 Lista de riesgos

La lista de riesgos o peligros que pueden afectar al proyecto, son:

- **R1**. Planificación temporal optimista: no se finaliza el proyecto en la fecha prevista
- **R2**. Falta alguna tarea necesaria: no se cumplen los objetivos.
- **R3**. Cambio de requisitos del Jefe de proyecto: retraso en el desarrollo.
- **R4**. Dificultad para contactar con Jefe de proyecto, ya que está en Zaragoza: faltan requisitos o son inadecuados.
- **R5**. No se hace correctamente la fase de test y pruebas: deficiencias en el módulo.
- **R6**. Incumplimiento de alguna norma o legislación: no se cumplen los objetivos.
- **R7**. Abandono del proyecto antes de la finalización: pérdidas económicas, frustración, fracaso absoluto.

3.5.2 Catalogación de riesgos

En la siguiente tabla, la probabilidad de que los riesgos puedan ocurrir y el impacto si lo hicieran.

Riesgo	Probabilidad	Impacto
R1	Mediana	Crítico
R2	Mediana	Crítico
R3	Alta	Relevante
R4	Alta	Relevante
R5	Mediana	Crítico
R6	Baja	Crítico
R7	Baja	Catastrófico

Tabla 8: Catalogación de riesgos

3.5.3 Plan de contingencia

Soluciones propuestas a los posibles riesgos:

- **SR1**. Aplazar alguna funcionalidad.
- **SR2**. Revisar el plan del proyecto.
- **SR3**. Mejorar el contacto con el Jefe de proyecto.
- **SR4**. Forzar más videoconferencias con el Jefe de proyecto.
- **SR5**. Ir haciendo pruebas durante la fase de Desarrollo.
- **SR6**. Revisar las normas y la legislación vigentes.
- **SR7**. Sin solución, simplemente no debe pensarse nunca que pueda ocurrir.

3.6 Presupuesto

3.6.1 Estimación coste de personal

A continuación, el coste por hora de cada recurso humano y el coste total que implicará cada trabajador en el progreso del proyecto.

Función	Coste/hora	Horas	Coste total
Jefe de proyecto	30 €/hora	58 horas	1740 €
Analista	15 €/hora	196 horas	2940 €
Programador	10 €/hora	340 horas	3400 €
Técnico de pruebas	8 €/hora	50 horas	400 €
TOTAL			**8480 €**

Tabla 9: Estimación coste de personal

3.6.2 Estimación coste de los recursos

Como en el apartado anterior, el coste de los recursos materiales y las licencias necesarias para el desarrollo del proyecto, también su amortización.

RECURSOS	COSTE UNITARIO	PERÍODO DE AMORTIZACIÓN	PERÍODO DE UTILIZACIÓN	COSTE DE AMORTIZACIÓN
Recursos materiales				
PC programador	500 €	36 meses	7,5 meses	104,17 €
Licencias				
Karat 7.5.6002	0 €	36 meses	7,5 meses	0 €
Karat 8.0	0 €	36 meses	7,5 meses	0 €
Eclipse IDE for Java Developers	0 €	36 meses	7,5 meses	0 €
Oracle SQL Developer	0 €	36 meses	7,5 meses	0 €
Notepad++ v5.9.6.2	0 €	36 meses	7,5 meses	0 €
Skype 5.8.0.158	0 €	36 meses	7,5 meses	0 €
Microsoft Office 2010	450 €	36 meses	7,5 meses	93,75 €
TOTAL				**197,92 €**

Tabla 10: Estimación coste de los recursos

3.6.3 Estimación coste de las actividades

De las tareas programadas para el desarrollo del proyecto, aquí el coste de cada una, dependiendo de los recursos utilizados para dicha actividad.

Nombre de la tarea	Duración	Recursos	Coste
FACTURACIÓN DE SERVICIOS	**162 días**		
Inicio del proyecto: asignación y matriculación	1 día	JP	120 €
Cursos de formación	10 días	P	400 €
Planificación	**6 días**		
Estudio de viabilidad	2 días	JP[50%], A[50%]	180 €
Aprobación Estudio de viabilidad	1 día	JP	120 €
Plan del proyecto	2 días	JP[50%], A[50%]	180 €
Aprobación Plan del proyecto	1 día	JP	120 €
Análisis	**24 días**		
Reunión con el Jefe de proyecto	1 día	JP[50%], A[50%]	90 €
Análisis del sistema actual y los requisitos	10 días	JP[50%], A[50%]	900 €
Análisis de las bases de datos	10 días	A	600 €
Documentación del análisis	2 días	A	120 €
Aprobación del análisis	1 día	JP	120 €
Diseño	**33 días**		
Diseño modular de la aplicación	10 días	A[50%], P[50%]	500 €
Diseño de la base de datos	10 días	A[50%], P[50%]	500 €
Diseño de la interfaz	5 días	A[50%], P[50%]	250 €
Diseño de test y pruebas	5 días	A[50%], TP[50%]	230 €
Documentación del diseño	2 días	A	120 €
Aprobación del diseño	1 día	JP	120 €
Desarrollo	**38 días**		
Preparación del entorno de desarrollo	2 días	P	80 €
Configuración de las librerías a importar	1 día	P	40 €
Desarrollo de las clases	5 días	P	200 €
Desarrollo de las funcionalidades	30 días	P	1200 €
Test y pruebas	**22 días**		
Pruebas unitarias	3 días	TP	96 €
Pruebas funcionales	3 días	TP	96 €
Pruebas de integración	3 días	TP	96 €
Documentación de las pruebas	2 días	P[50%], TP[50%]	72 €

Aprobación de las pruebas	1 día	JP	120 €
Corrección de errores	10 días	P	400 €
Implantación	**6 días**		
Implantación del módulo al producto	5 días	A[50%], P[50%]	250 €
Generación del archivo JAR	1 día	P	40 €
Generación de documentos	20 días	A[50%], P[50%]	1000 €
Cierre del proyecto	1 día	JP	120 €
Defensa del proyecto	1 día	JP	120 €
TOTAL			**8480 €**

Tabla 11: Estimación coste de las actividades

3.6.4 Resumen y análisis coste beneficio

El coste real de este proyecto ya está fijado en el convenio de prácticas establecido entre la empresa y la universidad, es de 2500 €.

Volviendo a la simulación de este Plan de proyecto, los costes serán la suma de los recursos materiales, las licencias y el coste de las actividades, coste que depende del sueldo del personal. Afortunadamente, la mayoría de licencias son públicas o proporcionadas por la empresa y su coste es de 0 €.

Detalle de coste	
Coste de los recursos materiales	104,17 €
Coste de las licencias	93,75 €
Coste de las actividades	8480 €
TOTAL	**8677,92 €**

Tabla 12: Resumen y análisis coste beneficio

Analizando estos costes, el proyecto puede parecer un poco caro, pero se deben de considerar todos los beneficios que se obtendrán con el desarrollo de este nuevo módulo.
Aparte de mejorar la gestión en el campo de la facturación de servicios, no solamente es un proyecto que vaya a ser demandado por la distribuidora farmacéutica, sino que cualquier empresa dedicada al sector va a necesitar de esta nueva herramienta.

Aunque el coste de la aplicación no se vaya a recuperar de forma directa, si el producto se vende correctamente se irá rescatando esta inversión.

Análisis y diseño

5.1 Introducción

En este apartado se explicarán el análisis y el diseño del proyecto. La arquitectura del sistema y la interfaz de usuario, por lo que al diseño se refiere y después las funcionalidades del sistema dentro del análisis, que vendrían a ser los requisitos funcionales de la aplicación.

5.2 Arquitectura del sistema

5.2.1 Plataforma tecnológica karat

La herramienta se elaborará bajo la plataforma tecnológica karat, a continuación una completa descripción acerca del desarrollo de aplicaciones en esta arquitectura.

El desarrollo en karat consiste en crear definiciones lógicas de todos los objetos que conformarán una determinada aplicación de gestión. Estas definiciones se almacenan en una serie de tablas específicas en la base de datos que tiene la función de repositorio. Veamos el siguiente gráfico:

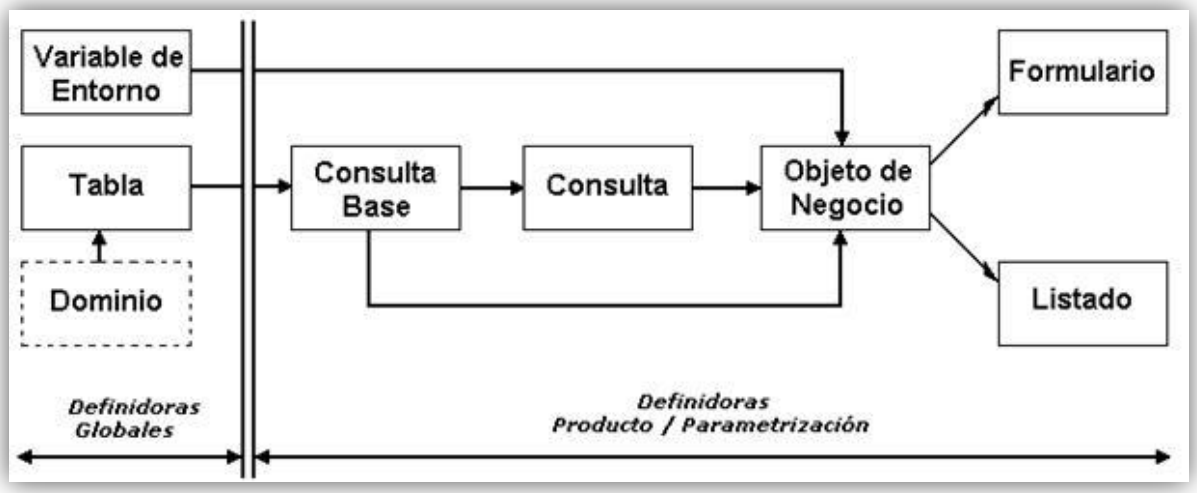

Figura 3: Plataforma tecnológica karat

Aquí una breve explicación del funcionamiento de esta arquitectura con un ejemplo real:

Surge la necesidad de gestionar una parte del área de distribución, el mantenimiento de 'Albaranes'. Primeramente, se deberá de crear un objeto de tipo 'Tabla' donde guardaremos toda la información de estos albaranes. Al crear un objeto 'Tabla' será necesario consultar los registros

que contiene, por lo que se creará otro objeto de tipo ‘Consulta Base’. Este objeto es una sentencia SQL que nos extrae la información de la tabla. Una vez creada la ‘Consulta Base’ se debe definir un objeto llamado 'Objeto de Negocio' y es a partir de este objeto cuando se podrán diseñar 'Formularios', 'Listados' o ambos.

5.2.2 Definición de objetos karat

Una breve descripción de cada objeto de esta arquitectura karat:

- **Tabla**: Definición lógica que tiene como objetivo guardar información en la base de datos para su posterior mantenimiento.
- **Consulta Base**: Sentencia en SQL a una o varias tablas donde se puede restringir la información que se desea mostrar, a nivel de filas o a nivel de columnas.
- **Consulta**: Sentencia en SQL donde se puede restringir la información que se desea mostrar única y exclusivamente a nivel de registros.
- **Objeto de negocio**: Definición de una entidad lógica de la cual se basan otros objetos: Formularios, Listados, Informes Navegables, etc. La mayor parte de la funcionalidad y por tanto del desarrollo que se requiere en una aplicación de gestión se define a nivel de 'Objeto de Negocio'. Lo que se diseña a este nivel se hereda de forma automática en todos los objetos dependientes. Podríamos decir que el 'Objeto de Negocio' es la matriz del desarrollo en karat. Un 'Objeto de Negocio' está compuesto por lo que se conoce como 'Paneles'. El primer panel es obligatorio y se conoce como 'Panel de Cabecera'.
- **Panel**: Se compone de unos elementos llamados 'Controles'. Estos 'Controles' pueden o no estar ligados a 'Campos' de la 'Consulta Base' o 'Consulta' si es que dependen de alguno de ellos.
- **Formulario**: Definición de un objeto gráfico que se usa para mostrar información por pantalla e interactuar con ella. Un 'Formulario' se compone de lo que se conoce como 'Ventanas'. Todo 'Formulario' se compone como mínimo de una ventana llamada 'Ventana Base'. Dado que un 'Formulario' siempre depende de un 'Objeto de Negocio' en las ventanas de dicho 'Formulario' se muestran representaciones de los 'Controles' en lo que se conoce como 'Vistas de Controles'.
- **Listado**: Definición de un objeto que se usa para volcar información, ya sea en formato digital o mediante impresoras. Un 'Listado' se compone de lo que se conoce como 'Secciones'. Todo 'Listado' se compone como mínimo de una sección llamada 'Sección Raíz'. Las secciones de un 'Listado' normalmente están formadas por representaciones de los controles del 'Objeto de Negocio' del cual dependen y se conocen como 'Campos'.

Existe relación entre los objetos karat y los elementos que los componen. De una tabla compuesta por campos puede crearse un objeto de negocio compuesto por controles y de éste se pueden establecer formularios o listados con vistas de control o campos. Véase en la siguiente figura:

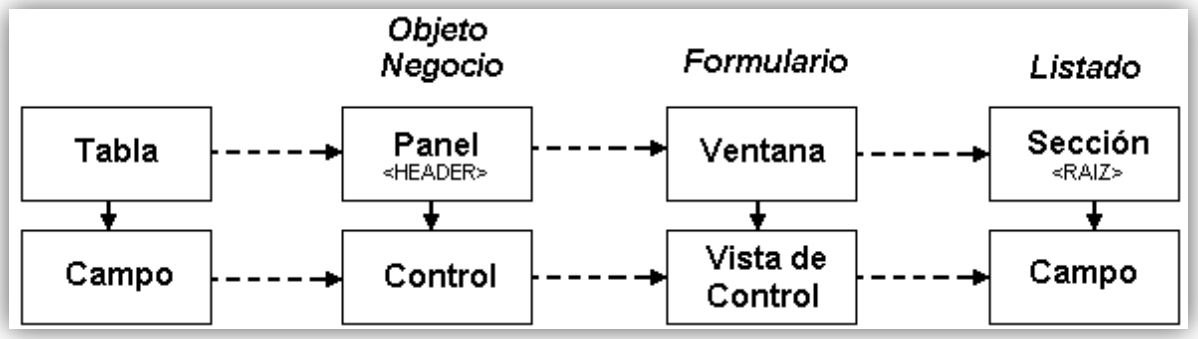

Figura 4: Definición de objetos karat

La forma de crear estos distintos objetos es mediante definidoras de producto (de tablas, de consultas, de objetos de negocio, etc.) alojadas en el software Karat Studio. Más adelante se explicará el funcionamiento de las definidoras y de Karat Studio en el entorno de desarrollo.

5.3 Interfaz de usuario

La forma que el cliente tiene de interactuar con la aplicación es a través de Karat Escritorio. Es el software ERP donde se visualizarán objetos como formularios o listados, ya que el resto de objetos no serán visibles, solamente formarán parte del entorno de desarrollo. Desde Karat Estudio se crearán formularios y listados siguiendo el estilo *Walnut*, modo gráfico explicado anteriormente.
En la siguiente figura se puede observar la interfaz que ofrece Karat Escritorio.

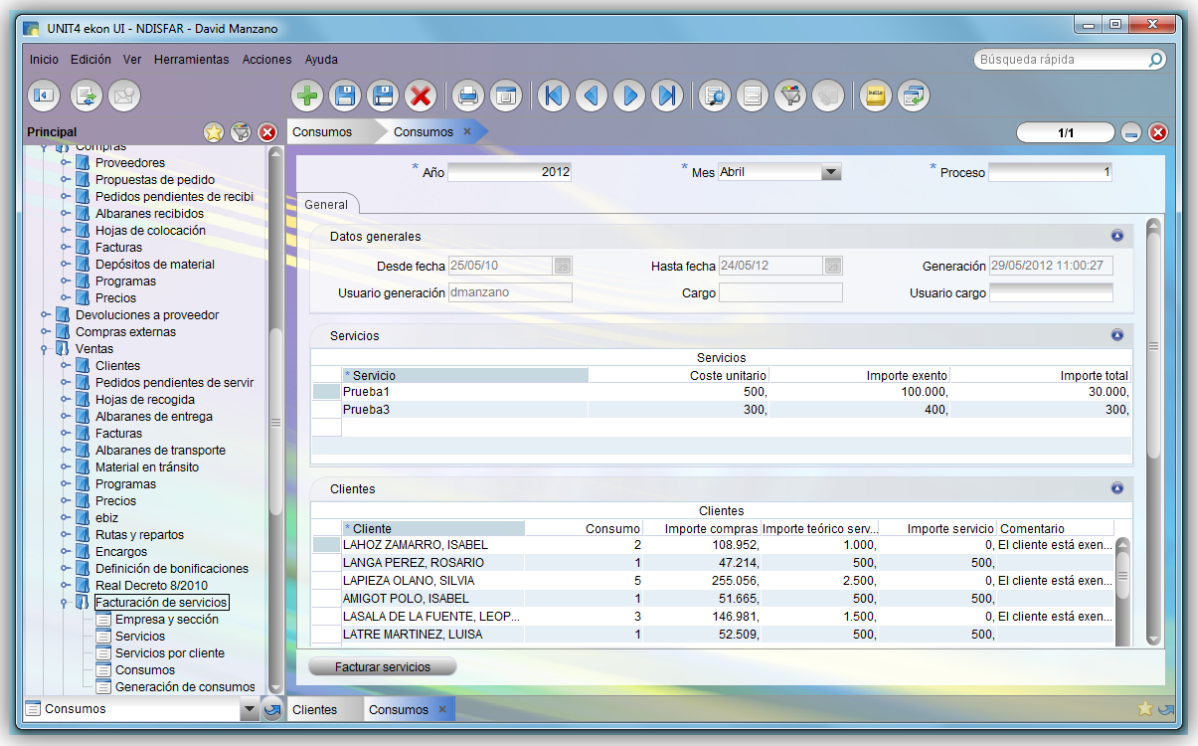

Figura 5: Interfaz de usuario

5.4 Diagrama de la Base de Datos

En este diagrama solamente se mostrarán las tablas que se han creado para el desarrollo del módulo, no se contemplarán las tablas existentes previamente a las que se han hecho claves foráneas. A continuación una enumeración de éstas clasificadas por el formulario en el que se utilicen:

Formulario de mantenimiento “Servicios”:

- **vf_servicios_alb**: tabla con información genérica de los servicios.
- **vf_servicios_tct**: tabla con los tipos de cliente a tratar de los servicios.
- **vf_servicios_snt**: tabla con las series de albarán a no tratar de los servicios.
- **vf_servicios_int**: tabla con los intervalos horarios de los servicios.
- **vf_servicios_docu**: tabla con los documentos de gestión a tratar de los servicios.
- **vf_servicios_sub**: tabla con los subtipos de gestión a tratar y a no tratar de los servicios.

Formulario de mantenimiento “Empresa y sección”:

- **vf_sec_serv**: tabla con información genérica de los servicios por empresa.
- **vf_sec_imp**: tabla con las secciones a tratar de la empresa, para la generación posterior de consumos.
- **vf_sec_snt**: tabla con las series de albarán a no tratar de los servicios por empresa.

Formulario de mantenimiento “Servicios por cliente”:

- **vf_cli_serv**: tabla con información relacionada con el módulo por cliente.
- **vf_cli_por_serv**: tabla con información relacionada con el módulo por cliente y servicio.

Formulario de mantenimiento “Consumos”:

- **vf_mant_resu_cab**: tabla con información genérica de los consumos creados.
- **vf_mant_resu_serv**: tabla con los servicios que se han consumido.
- **vf_mant_resu_cli**: tabla con los servicios consumidos, los clientes y su consumo correspondiente.
- **vf_mant_resu_alb**: tabla con los servicios consumidos, los clientes, su consumo y albaranes correspondientes.

En la figura 6 vemos el diagrama de bases de datos que se ha detallado.

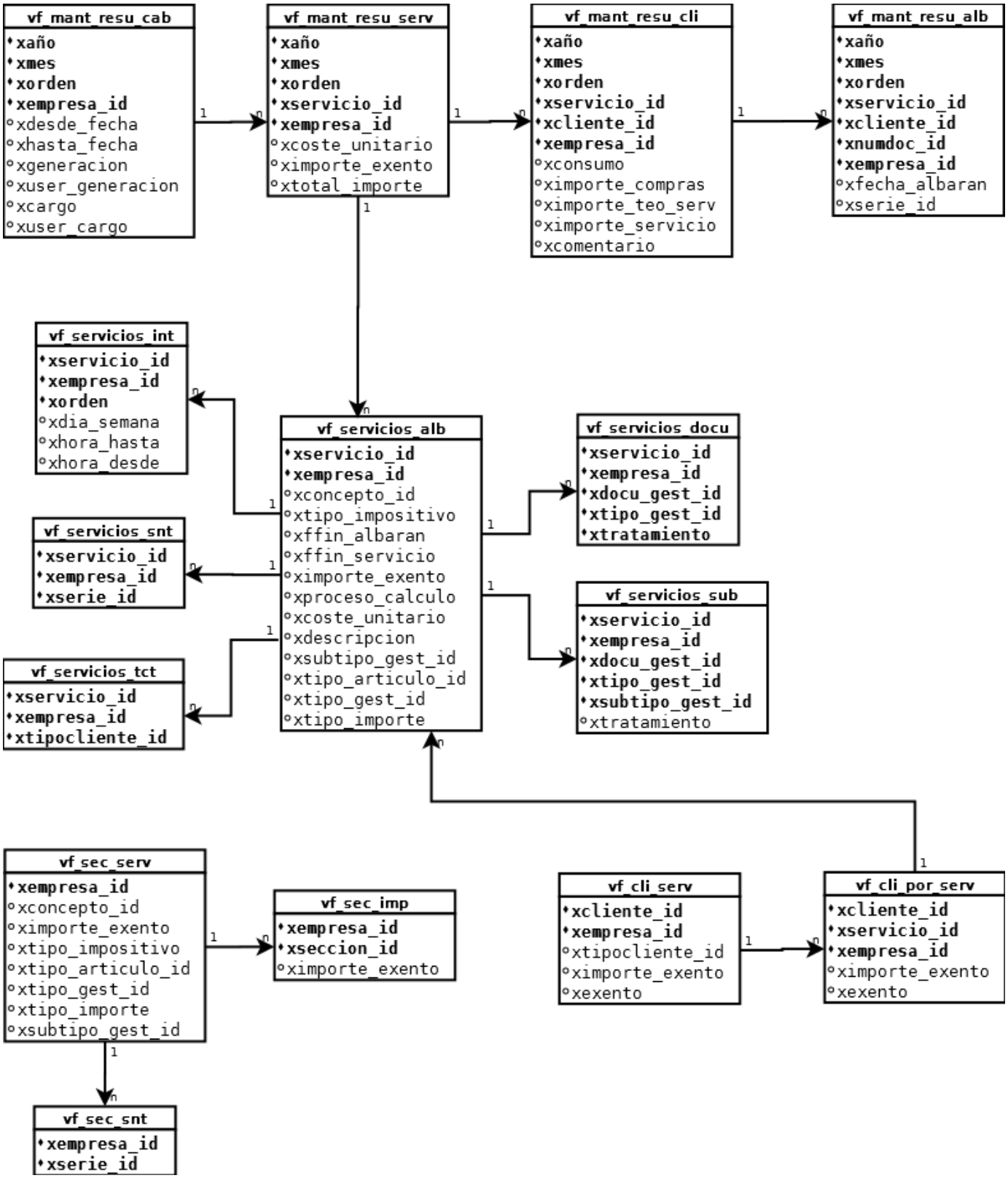

Figura 6: Diagrama de la Base de Datos

5.5 Funcionalidades del sistema

En este apartado se explicarán las diferentes funcionalidades de la aplicación, que son principalmente una breve explicación de cada uno de los requisitos funcionales especificados anteriormente en la sección 2.3.1.

Objetivo 1: Inserción y mantenimiento de servicios en la base de datos.

- **RF1. Crear un formulario de mantenimiento llamado "Servicios".**
  El objetivo será crear un formulario para insertar, modificar o eliminar registros en la base de datos. En cada registro se guardará información acerca de un servicio en concreto. Para la creación de este formulario se seguirá todo el procedimiento establecido por la arquitectura de karat: creación de tabla, de consulta base, de objeto de negocio y finalmente de formulario.
- **RF2. Definir tipos de cliente del servicio.**
  Se debe poder escoger qué clase de clientes aceptará cada servicio. Esto se verá reflejado en el registro donde se guarde el servicio. Como solución, se ha creado otro panel dentro del objeto de negocio con acceso a otra tabla y otra consulta base donde se guardarán estos clientes. Dentro del formulario se verá como un *colapsible* llamado "Tipos de cliente a tratar", este *colapsible* acogerá la información del panel creado en el objeto de negocio.
- **RF3. Definir intervalos horarios del servicio.**
  Cada servicio tendrá sus horarios de disponibilidad: día de la semana y un intervalo de horas. Como en la funcionalidad anterior, se creará un *colapsible* llamado "Intervalos horarios" y se guardará esta información en otra tabla de la base de datos.
- **RF4. Definir series de albarán a no tratar del servicio.**
  No todos los servicios podrán aceptar cualquier albarán. Esto lo controlará la serie del albarán, que es un número identificativo que engloba un grupo de albaranes. Se seguirá exactamente el mismo procedimiento que en la sección anterior, pero con un *colapsible* llamado "Series de albarán a no tratar".
- **RF5. Definir documentos de gestión a tratar del servicio.**
  Se trata de clasificar los albaranes según su tipo de gestión. En este caso puede ser de cargo, de abono o de devolución. En el siguiente *colapsible* llamado "Documentos de gestión a tratar", se especificaría que clase de albaranes podrá aceptar dicho servicio.
- **RF6. Definir subtipos de gestión a tratar o a no tratar del servicio.**
  El subtipo de gestión determina condiciones comerciales acerca del tipo de gestión que se trate, como por ejemplo transferencias, descuentos, etc. Se definirá si es a tratar o a no tratar y estará ubicado dentro del *colapsible* anterior, pero en otro panel distinto al de documentos de gestión.

En la figura 7 vemos el formulario de mantenimiento “Servicios”, donde se distinguen la cabecera y los distintos *colapsible* nombrados anteriormente.

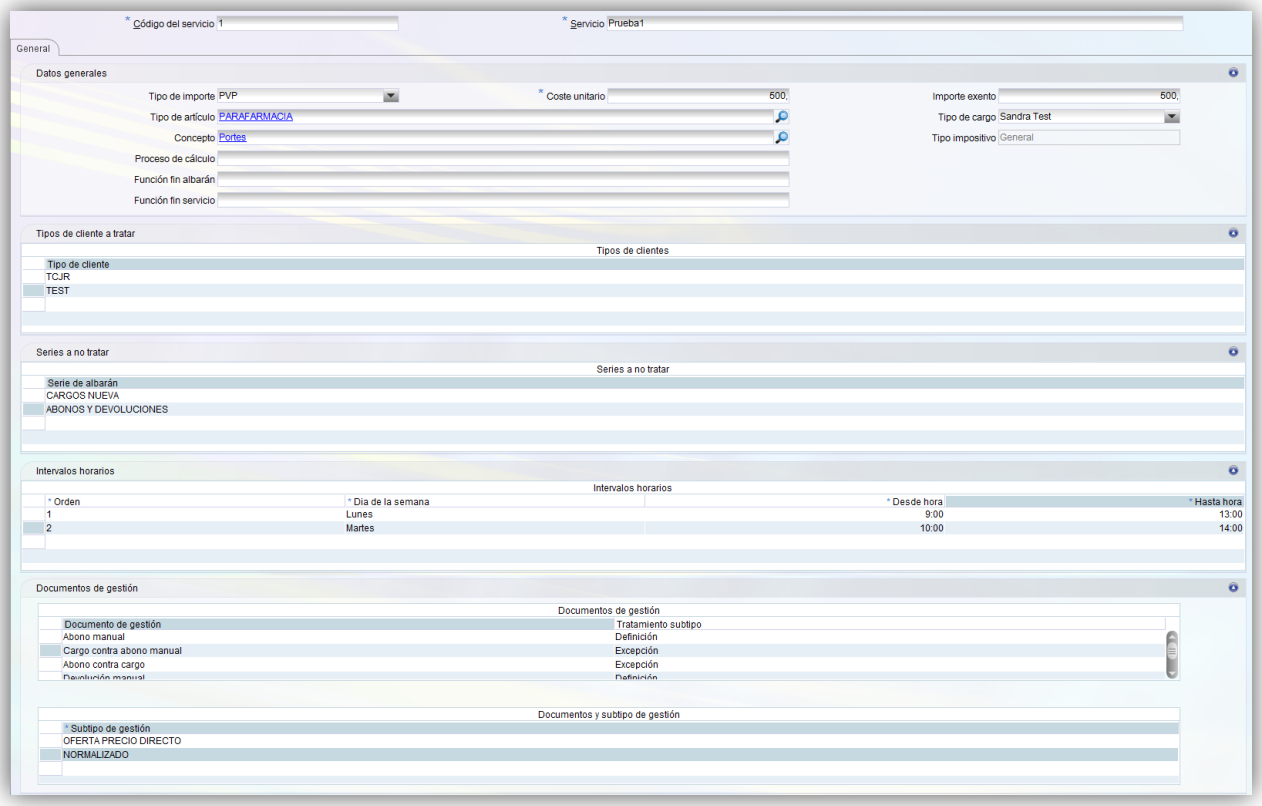

Figura 7: Funcionalidades del sistema. Formulario “Servicios”

Objetivo 2: Relacionar clientes con determinados servicios.

- **RF7. Crear un formulario de mantenimiento llamado “Servicios por cliente”.**
  Este formulario será creado con el fin de relacionar cierta información de los servicios con un cliente determinado. Cada registro en la base de datos contendrá únicamente un cliente y sus datos personales en referencia al módulo.
- **RF8. Definir un importe exento por cliente.**
  Este importe exento significa que, a partir del coste de las compras de un cliente en determinado albarán, no se le cobrará ningún servicio. Por ejemplo, si el importe de compras de cliente es de 4500 € en el albarán a facturar y éste supera o iguala al importe exento, al cliente no se le cobrará ningún servicio.
- **RF9. Definir si el cliente está exento de todos los servicios.**
  Lo que se conseguirá es dejar exento a este cliente de facturar cualquier servicio. Lo que significa que no se le cobrará absolutamente ninguna de estas prestaciones. En el formulario se verá reflejado como un *check* que se puede marcar o desmarcar.

- **RF10. Definir un importe exento por cliente y servicio.**
  Como en el caso anterior del importe exento a nivel de cliente, en este caso el importe será exclusivamente para un servicio en concreto. Dentro del formulario, éste se verá reflejado dentro de un *grid* llamado "Servicios".
- **RF11. Definir si el cliente está exento de un servicio en concreto.**
  Este *check* también estará ubicado en el *grid*, y dejará al cliente exento de facturar el servicio seleccionado. Cualquiera de estos dos *check* marcados elimina totalmente la función del importe exento, si ya no se le va a cobrar ninguna prestación al cliente, no tiene sentido tener este coste.

  Como se ha podido observar, hay distintos campos de importes exentos a lo largo del módulo de "Facturación de servicios". La elección de este importe funciona por preferencias, aquí vemos cada uno de los campos ordenados prioritariamente:
  - Formulario "Servicios por cliente": Importe exento dentro del *grid* "Servicios", a nivel de servicios por cliente.
  - Formulario "Servicios por cliente": Importe exento a nivel de cliente.
  - Formulario de "Servicios": Importe exento a nivel de servicio.
  - Formulario de "Empresa y sección": Importe exento en el *colapsible* de "Secciones", a nivel de sección.
  - Formulario de "Empresa y sección": Importe exento a nivel de empresa.

En la siguiente figura vemos el sencillo formulario de "Servicios por cliente".

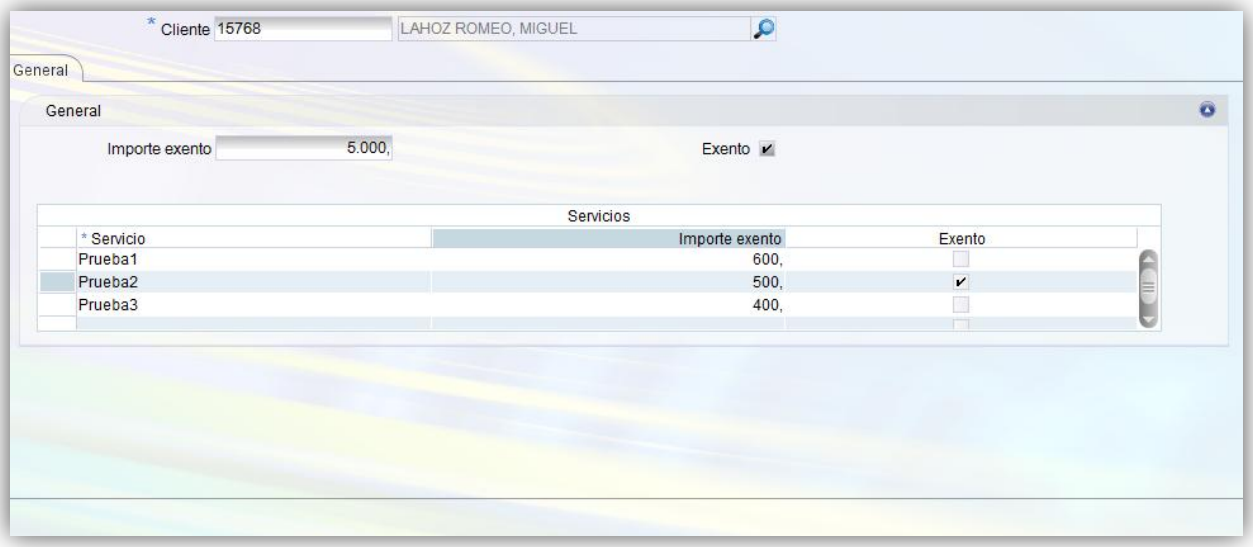

Figura 8: Funcionalidades. Formulario "Servicios por cliente"

Objetivo 3: Establecer datos generales por empresa y sección.

- **RF12. Crear un formulario de mantenimiento llamado "Empresa y sección".**
  La intención de este formulario será implantar las características genéricas de los servicios por empresa, evitando así definir esta información por servicio. En los registros de la base de datos, constará toda esta información por clave de empresa.
- **RF13. Definir series de albarán a no tratar de la empresa activa.**
  Como en el formulario "Servicios", se tratará de un mismo *colapsible* con las series de albarán que no tratarán las empresas.
- **RF14. Definir secciones a tratar de la empresa y el posible importe exento de cada una.**
  En primer lugar definiremos concepto de sección de una empresa. Una sección existe cuando una empresa tiene varias sucursales en distintos lugares. Cada sucursal es una sección. Entonces, se tratará de un *colapsible* donde se introducirá cada filial de la empresa que tratarán los servicios. Para cada sección se podrá definir un importe exento, como se ha explicado en el objetivo 2 y este importe exento ocupará el cuarto lugar prioritariamente hablando.

A continuación, vemos una imagen donde se muestra el formulario "Empresa y sección".

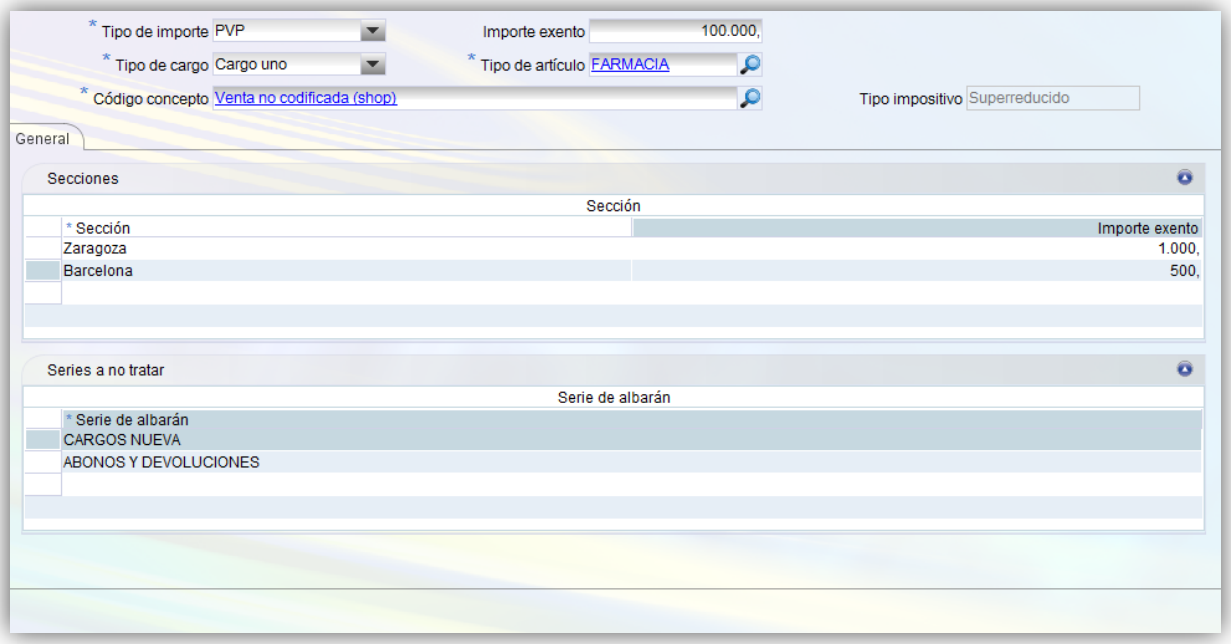

Figura 9: Funcionalidades. Formulario "Empresa y sección"

Objetivo 4: Generación de consumos de los servicios seleccionados.

- **RF15. Crear un formulario de diálogo llamado "Generación de consumos".**
  Antes de todo, decir que un formulario de diálogo o cálculo no guarda información en la base de datos, simplemente captura datos en los campos para su posterior cálculo.
  Es necesario saber qué significa la generación de consumos y qué cálculos realiza. Este proceso consiste en facturar a los clientes los servicios seleccionados. Los servicios están caracterizados por propiedades logísticas y otras reglas de gestión, características que comparten con los albaranes. La generación de consumos buscará en la base de datos los albaranes que contengan las mismas propiedades que los servicios seleccionados en el proceso, entonces se cogerán los clientes que contengan estos albaranes para calcular sus respectivos consumos.
  RF16. Generar consumos entre un intervalo de fechas y clientes.
  En la búsqueda de albaranes en la base de datos, solamente se buscarán coincidencias para los resultados que estén entre estas dos fechas y entre estos dos clientes.
- **RF17. Generar consumos para todos los clientes.**
  Será posible marcar un *check* para generar consumos para todos los clientes del histórico de albaranes. Al marcar este *check* se desactivarán los dos campos donde se introduce el rango de clientes, campos "Desde cliente" y "Hasta cliente".
- **RF8. Generar consumos para un determinado número de servicios.**
  Otra de las funcionalidades de las que ha de disponer este formulario es la capacidad de poder seleccionar más de un servicio para la generación de consumos. Para solucionar este problema, se creará un *grid* donde se introducirán los servicios que se deseen.

En la siguiente figura vemos una vista del formulario "Generación de consumos".

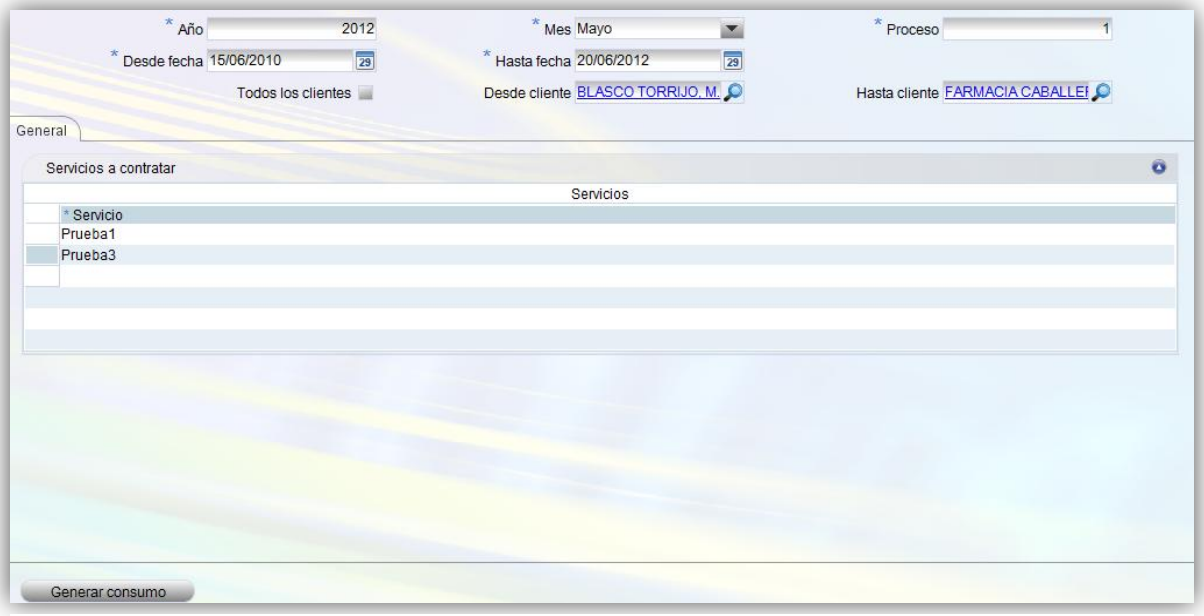

Figura 10: Funcionalidades. Formulario "Generación de consumos"

Objetivo 5: Mantenimiento de los consumos generados.

- **RF19. Crear un formulario de mantenimiento llamado "Consumos".**
  Aquí se mostrarán los resultados obtenidos en el proceso de generación de consumos, pudiendo modificar algunos parámetros a gusto del cliente. Este formulario estará compuesto por 3 *colapsibles*, en el primero donde figurarán los servicios que se van a facturar, en el segundo estarán los clientes con sus consumos correspondientes acerca del servicio seleccionado y en el tercero los albaranes por cliente seleccionado.
- **RF20. Cálculo de algunos campos del formulario en tiempo real.**
  También existirán ciertos campos calculados, esto significa que al modificar algún valor de un campo, cambiará el resultado final del consumo en consecuencia al cálculo.

En la siguiente imagen vemos el formulario de mantenimiento "Consumos".

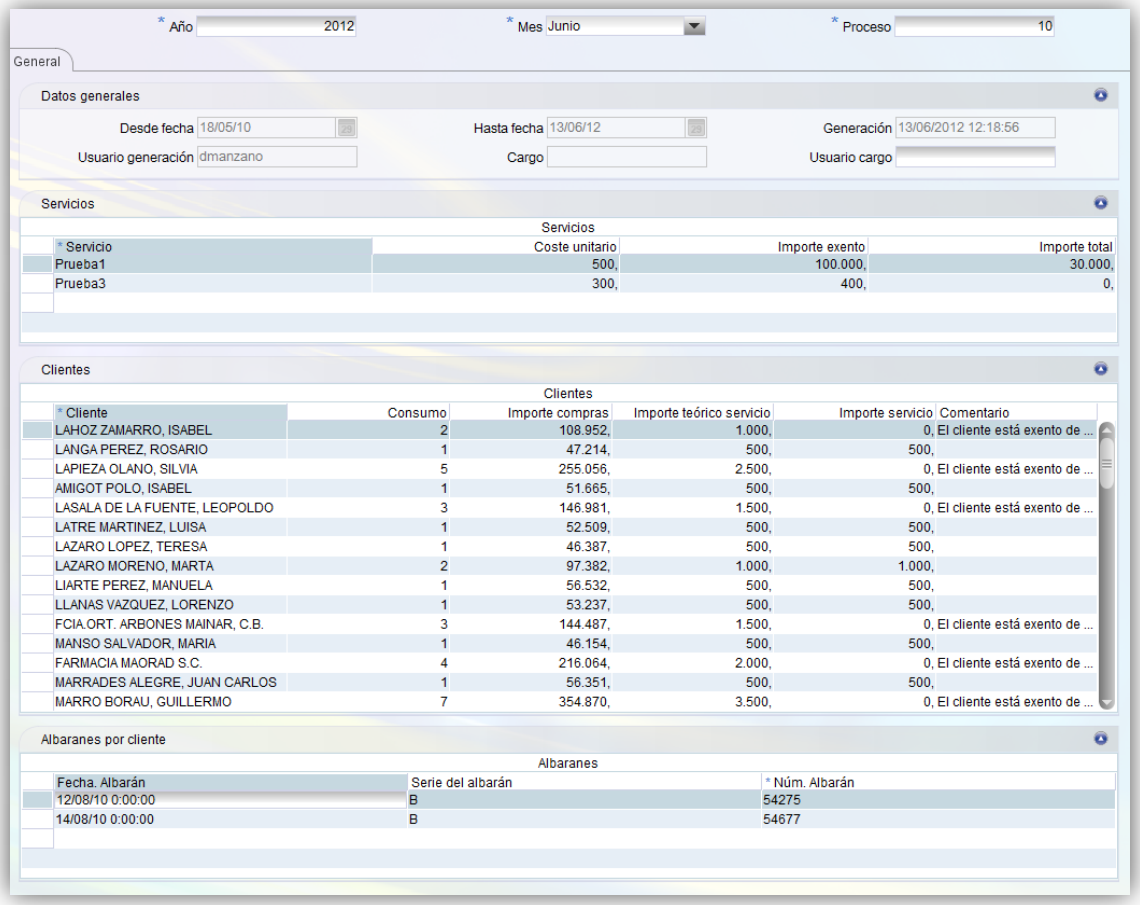

Figura 11: Funcionalidades. Formulario "Consumos"

Objetivo 6: Generación de listados de los consumos anteriores.

- **RF21. Mostrar en vista de impresión los consumos generados.**
  Se creará una vista de impresión del formulario de mantenimiento anterior de consumos, que se llamará "Listado de consumos". Se podrá crear en formato PDF, DOC, RTF, TXT, etc.

Desarrollo

6.1 Introducción

En esta sección se explicarán todos los procesos que se han seguido para la correcta implementación del módulo, tanto el entorno de desarrollo establecido por la empresa, como su codificación en lenguaje Java.

El desarrollo de la aplicación constará de dos partes. La parte más básica y lógica, la base del módulo, se efectuará bajo el software de la empresa Karat Studio. Las funcionalidades más específicas se desarrollarán con Eclipse, ya que con Karat Studio no es posible profundizar tanto. También se ha utilizado Oracle SQL Developer para generar las consultas a la base de datos.

6.2 Entorno de desarrollo

6.2.1 Karat Studio

En este apartado se profundizará en las funcionalidades de este programa de la empresa. Como se ha explicado anteriormente, Karat Studio nos permitirá establecer desde cero la aplicación, con la creación de objetos karat. En la figura 12 vemos la apariencia del software karat Studio.

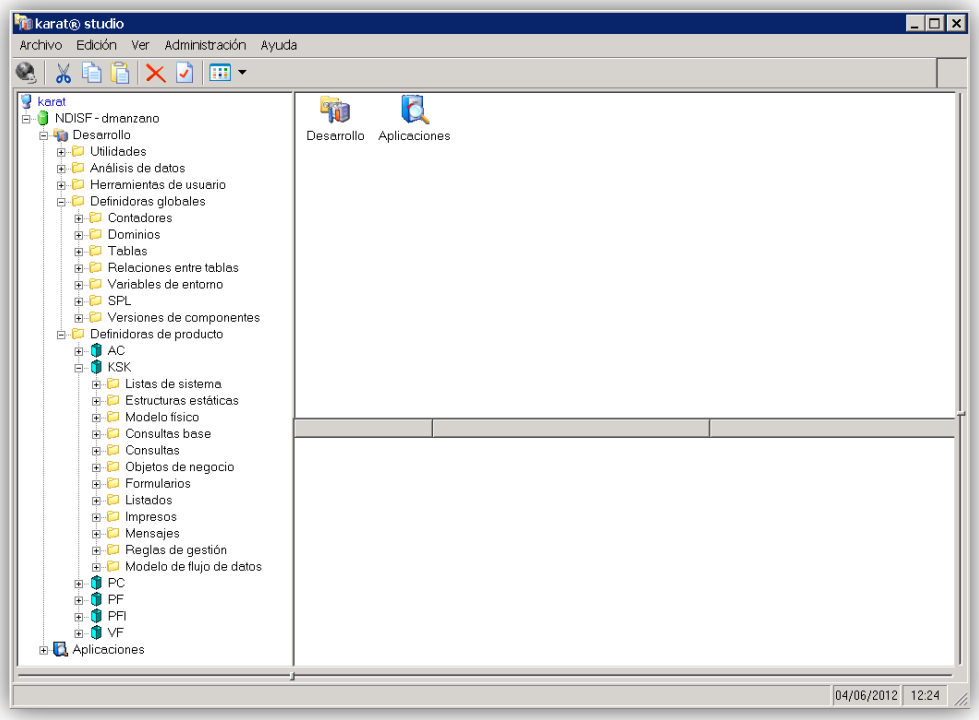

Figura 12: Entorno de desarrollo. Karat Studio

Para cada uno de estos objetos karat existirá una definidora, que es una parte de Karat Studio que nos permitirá crearlos. Las definidoras pueden ser de dos tipos: globales o de producto. Las definidoras globales nos crearán los objetos que no necesiten estar vinculados a ningún producto, mientras que las definidoras de producto si lo estarán. Un producto es una parte de gestión de la empresa, como por ejemplo: finanzas, ventas, logística, etc. La única definidora global es la definidora de tablas, el resto son de producto.

Definidora de tablas: Esta definidora nos creará las tablas en la base de datos, donde se guardará la información que se necesite. Se podrán crear tantos campos como sean necesarios, se nombrarán y se definirá su tipo (ya sea un campo de texto plano, un entero, etc.). También se podrán establecer tanto las claves primarias de la tabla como las claves foráneas a otras tablas de la base de datos. Otra de las ventajas de esta definidora es la opción de introducir etiquetas en los campos y mensajes de ayuda, para la posterior creación del formulario que mostrará la información guardada en la tabla. En la figura 13 vemos una imagen dónde se puede observar el detalle de esta definidora de tablas.

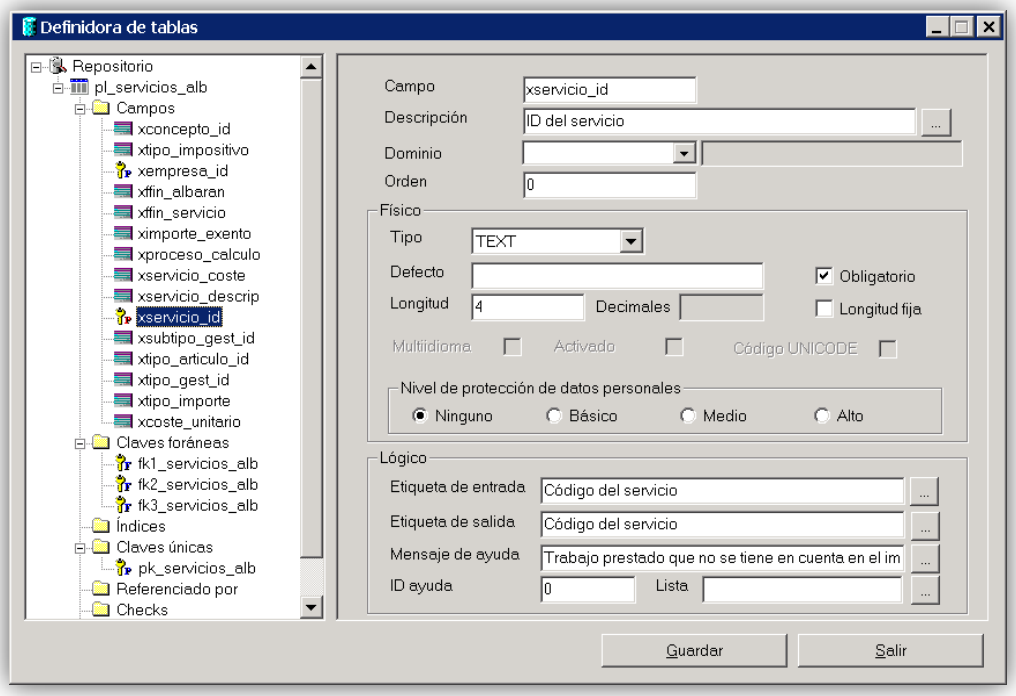

Figura 13: Karat Studio. Definidora de tablas

Definidora de consultas base: Esta definidora nos permitirá seleccionar la tabla que se necesite utilizar. La herramienta simplemente hará una SELECT a la base de datos de la tabla y nos dará la opción de seleccionar los campos que necesitemos. También se podrán establecer otras sentencias SQL, como WHERE, ORDER BY y JOINS. A continuación una figura de esta definidora.

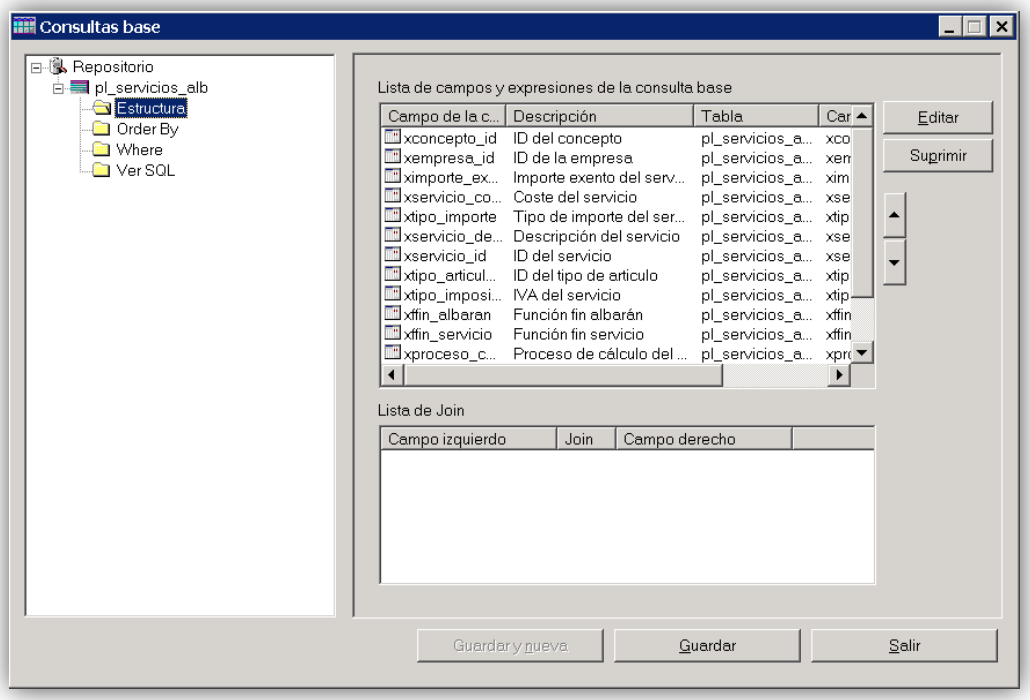

Figura 14: Karat Studio. Definidora de consultas base

Definidora de objetos de negocio: Esta herramienta hará posible la creación de distintos objetos de negocio. Se podrán crear tantos paneles del objeto de negocio como sean necesarios. Éstos se podrán asociar a una consulta y a su vez los controles se podrán asociar a los campos de esta consulta. Esta definidora es quizás la más importante de la plataforma karat, ya que nos permite darle muchas funcionalidades a los controles de los que se dispone. En la siguiente imagen vemos la definidora de objetos de negocio.

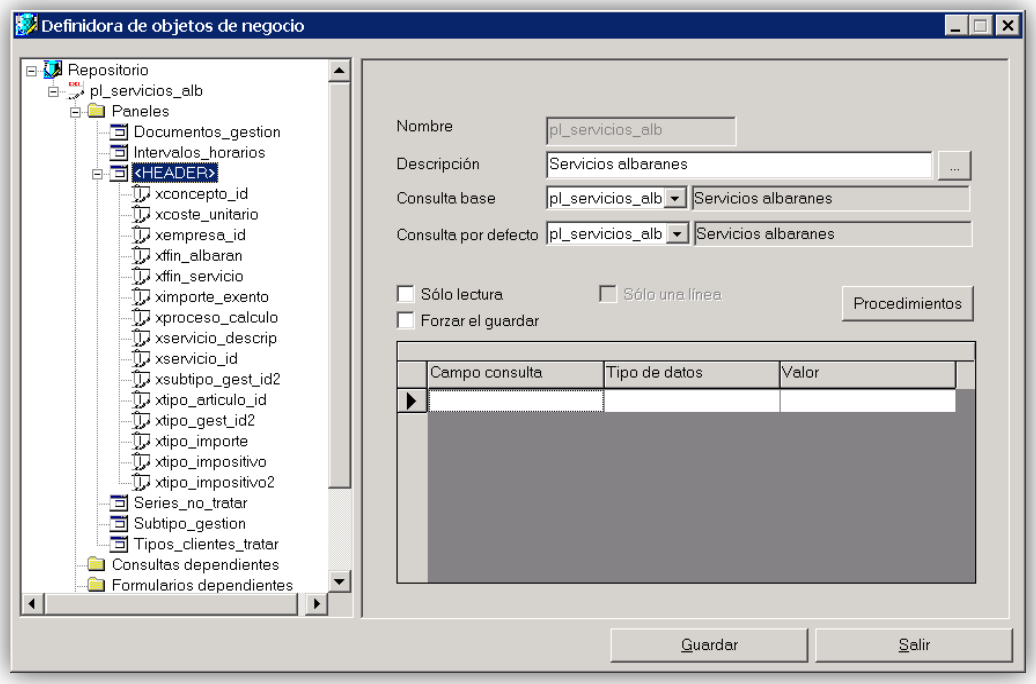

Figura 15: Karat Studio. Definidora de objetos de negocio

Diseñadora de formularios: Es básicamente la definidora de este objeto. Se pueden crear distintos *colapsibles* asociados a paneles, se pueden añadir *grids*, imágenes, calendarios, gráficos, etc. También es posible darle forma a cada campo, éstos pueden ser de tipo *TextBox*, *CheckBox*, *ComboBox* y otros muchos. En la figura vemos la diseñadora de formularios.

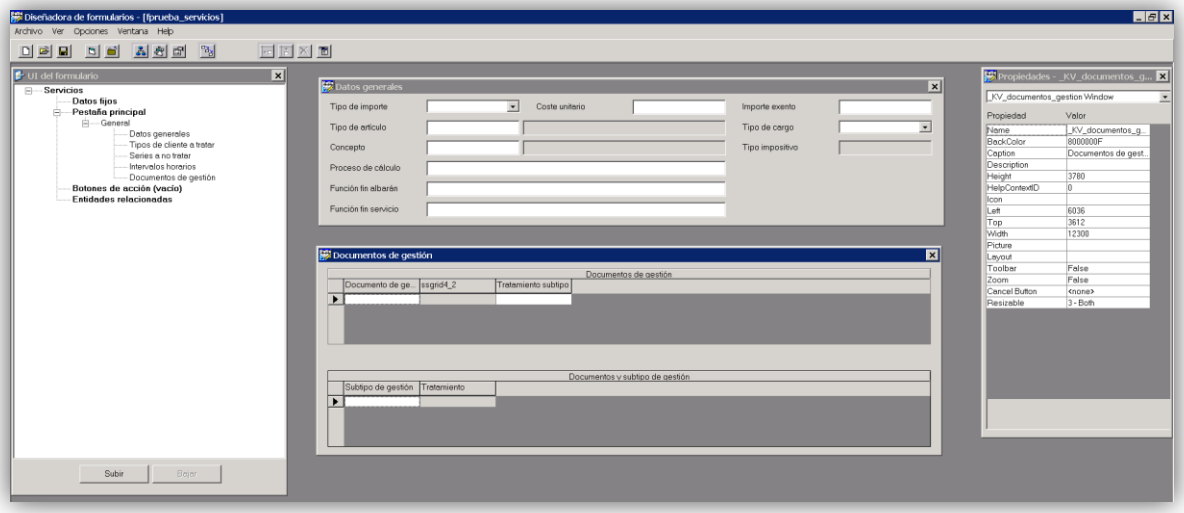

Figura 16: Karat Studio. Diseñadora de formularios

Diseñadora de listados: Y por último tenemos la diseñadora de listados. Esta herramienta hace posible crear la vista de impresión del formulario especificado de la forma que se desee. Pueden colocarse los campos, *grids* y otros objetos, en el orden que se prefiera. También pueden añadirse encabezados y pies de página, líneas separadoras, títulos, etcétera, como si se tratase de un archivo de Microsoft Word. A continuación tenemos una imagen de la diseñadora de listados.

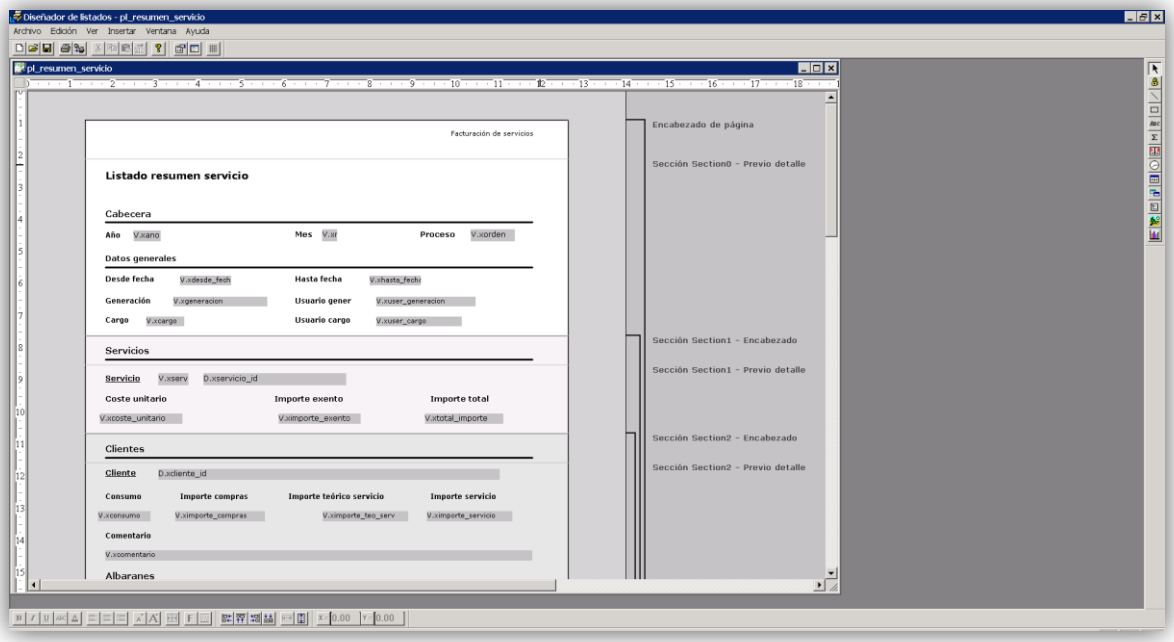

Figura 17: Karat Studio. Diseñadora de listados

6.2.2 Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar todo tipo de aplicaciones en lenguaje Java. Para desarrollar sobre la plataforma karat, la empresa proporciona una API propia, la API de karat. Una API es un conjunto de funciones y procedimientos que ofrecen cierta librería importada. Gracias a esta librería nueva, se podrá programar sobre este entorno.

Con Eclipse se pueden dar nuevas funcionalidades a cada objeto karat. La programación en Java del módulo está basada en los eventos que genera karat. Cada función de esta API identifica un evento, donde es posible programar cualquier acción. Los eventos ocurrirán en cada uno de los siguientes objetos karat: objetos de negocio, formularios y listados. Por ejemplo, la función *viewClick* es ejecutada cuando dentro de algún formulario el usuario hace *click* con el ratón, entonces cabe la posibilidad de programar una acción o ejecutar un algoritmo dentro de la función *viewClick* , ya que el código será ejecutado cuando el usuario haga un simple *click*.

En este módulo han sido necesarios seis *scripts*: dos para formularios y cuatro para objetos de negocio. El código más extenso ha sido el del formulario que ejecuta el complejo proceso de generación de consumos, con unas 1500 líneas. A continuación vemos una imagen que muestra el entorno de eclipse, junto con los formularios y objetos de negocio que han sido programados para obtener nuevas funcionalidades.

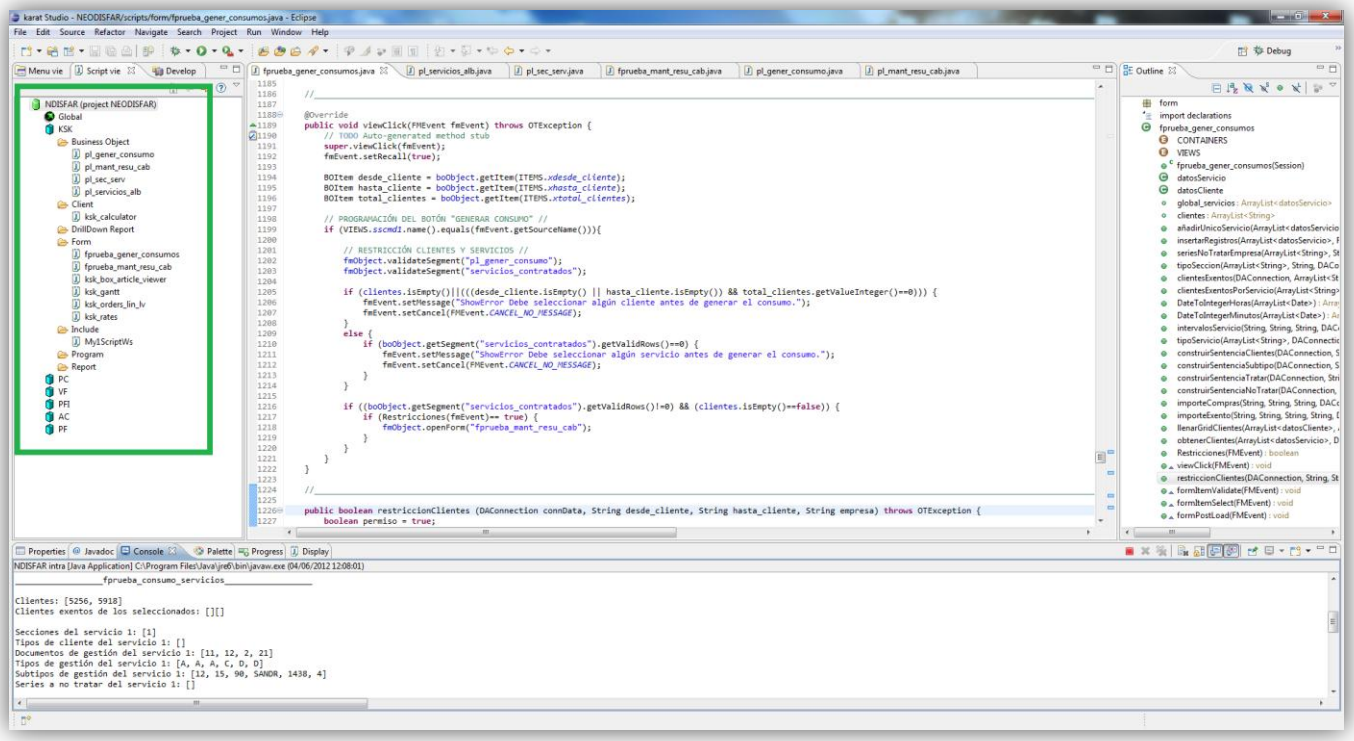

Figura 18: Entono de desarrollo. Eclipse

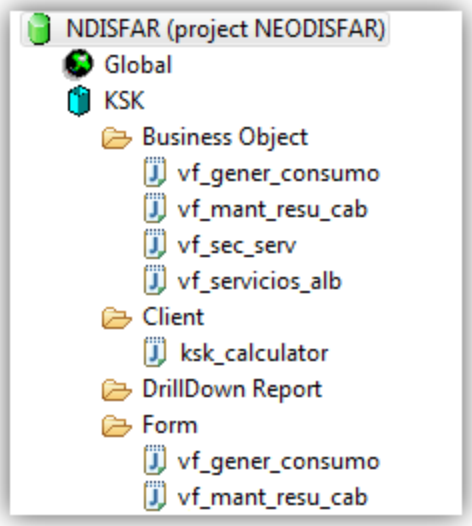

Figura 19: Eclipse. Repositorio de *scripts*

Como se puede observar en la figura 17, el proyecto contiene 6 *scripts* creados para las funcionalidades del módulo.

***Scripts* de los objetos de negocio:**

- vf_gener_consumo (Generación de consumos)
- vf_mant_resu_cab (Consumos)
- vf_sec_serv (Empresa y sección)
- vf_servicios_alb (Servicios)

***Scripts* de los formularios:**

- vf_gener_consumo (Generación de consumos)
- vf_mant_resu_cab (Consumos)

6.2.3 Oracle SQL Developer

Oracle SQL Developer es un entorno de desarrollo integrado (IDE) proporcionado por Oracle, para desarrollar o simplemente para ejecutar consultas o sentencias SQL, sobre las bases de datos Oracle de la empresa.

Este módulo SQL Developer se ha utilizado para hacer consultas SQL, básicamente para confirmar que los formularios grababan los registros correctamente en sus tablas de la base de datos y también para comprobar que los resultados obtenidos en la generación de consumos fueran acertados.

En la siguiente imagen tenemos una consulta total a la tabla del histórico de albaranes (pld_vhalbcli_cab) con el software Oracle SQL Developer. Encuadrado en color rojo vemos la sentencia SQL y en verde el resultado.

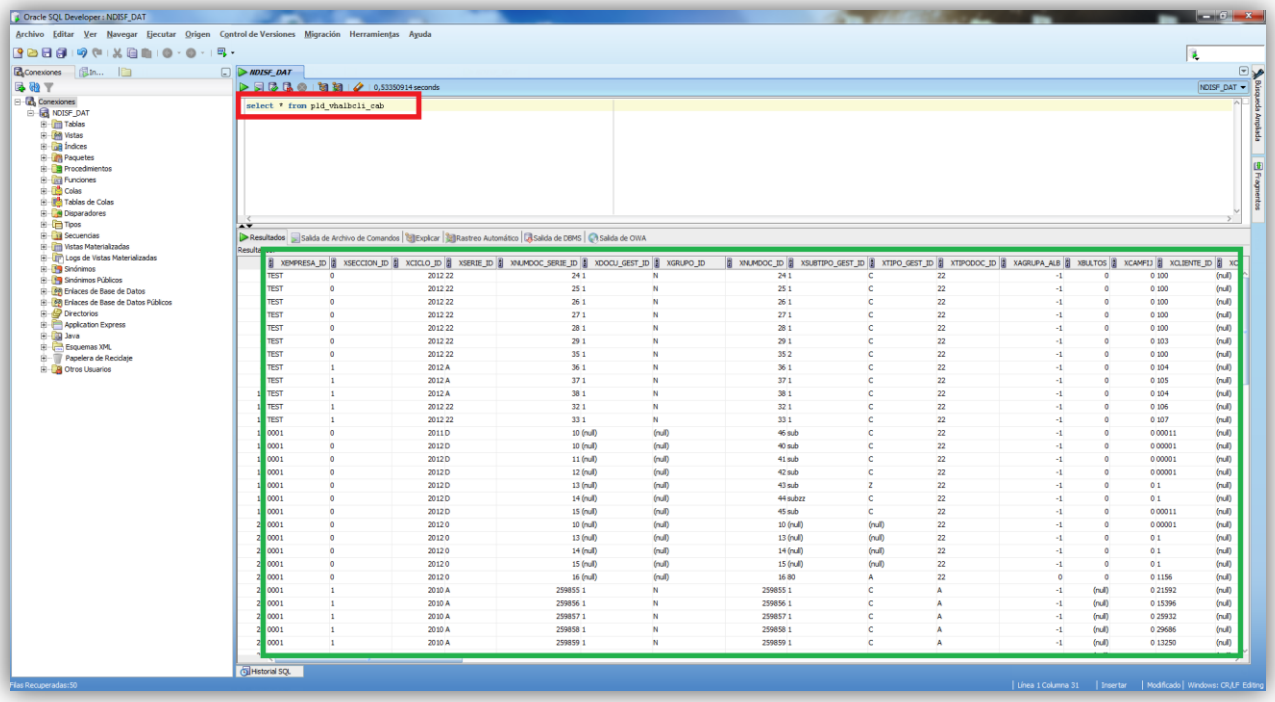

Figura 20: Entorno de desarrollo. Oracle SQL Developer

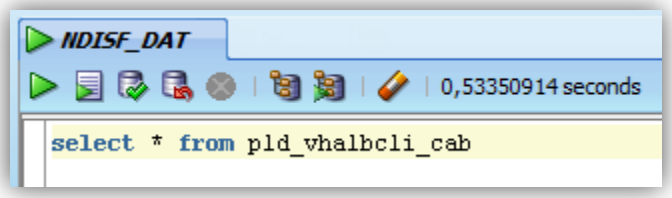

Figura 21: Oracle SQL Developer. Consulta SQL

6.3 Entorno de ejecución

Karat, en su conjunto, es una aplicación distribuida que requiere como mínimo una parte servidora y “n” partes clientes. El grupo de componentes que forman karat no tienen ningún sentido sin la existencia de una entidad llamada Repositorio (también conocido como Superdiccionario). El Repositorio es un conjunto de tablas con una determinada estructura que formarán el núcleo funcional de karat.

En karat todo se almacena, en mayor o menor medida, en una tabla específica del Repositorio. Los componentes que forman karat aportan una forma gráfica y segura de alimentar dichas tablas del Repositorio.

Por ejemplo, al trabajar mediante autenticación karat, cada vez que se intenta acceder al entorno de trabajo la herramienta solicita la introducción de un usuario y una contraseña para permitir el acceso. Por lo tanto, está claro que las herramientas necesitan contrastar la información introducida por el usuario con los datos almacenados en alguna tabla relacionada con la seguridad. Esta tabla está almacenada en el Repositorio y existen componentes karat que permiten administrar de una forma gráfica su contenido. A continuación vemos una imagen del sistema de autenticación karat.

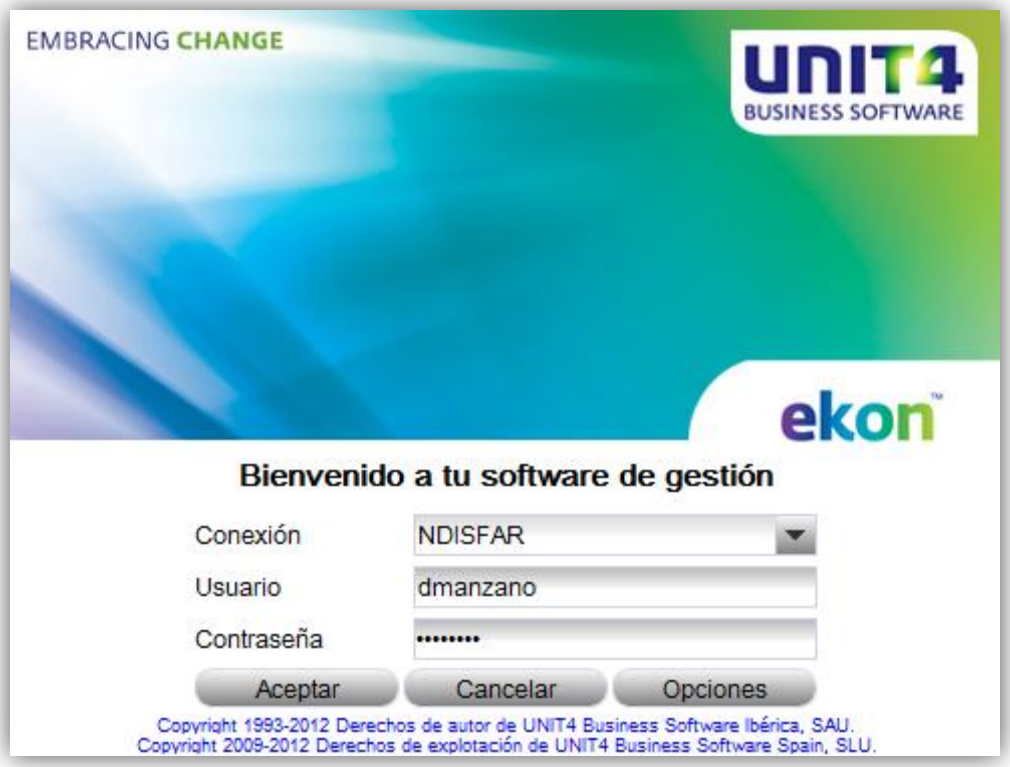

Figura 22: Entorno de ejecución. Autenticación karat

El conjunto de tablas que aportan funcionalidad a karat se denomina Repositorio. Si no se dispone de como mínimo un Repositorio no es posible definir un entorno karat. Estas tablas se ubican en una Base de Datos dentro del SGBD (Sistema Gestor de la Base de Datos).
Queda patente la imprescindible existencia de una entidad llamada Repositorio compuesta por un gran número de tablas que conformarán el núcleo funcional de karat. Evidentemente, además de esta entidad se requiere otra conocida como Data que estará formada por todas las tablas propias de los productos de gestión (clientes, empresas, facturas, albaranes,....) que formarán la verdadera información del usuario final.

Estas dos entidades: Repositorio y Data pueden convivir en un mismo fichero de Base de Datos. Es decir, físicamente podemos disponer de un único fichero de BBDD que realice las funciones de ambas entidades. Aunque esto es totalmente factible, es más recomendable separar ambas entidades ubicando cada una de ellas en sus respectivos ficheros de Base de Datos.

Por último enumeramos los requisitos necesarios para ejecutar el entorno karat:

- Cliente karat: Sistema operativo Windows, Linux o MAC, procesador de 32 o 64 bits, CPU 2,4 GHz o superior y 2 GB de memoria RAM o superior.
- Servidor karat: CPU 3.0 GHz o superior, 4 GB de memoria RAM o superior y red de 1 GB entre el servidor de aplicaciones y de base de datos.

Codificación y pruebas

7.1 Introducción

En este apartado, se mostrará el estilo de codificación utilizado durante el desarrollo del módulo y también las pruebas que han sido necesarias para testear la aplicación. Se han diseñado tanto pruebas unitarias como pruebas de integración. Aunque el capítulo de pruebas sea posterior al desarrollo, se han ido haciendo test y también corrigiendo sus errores durante la implementación de la aplicación.

7.2 Estilo de codificación

Para el estilo de codificación del módulo se ha utilizado el método *Kernighan&Ritchie/Kernel*, que simplemente contempla la posición de las llaves que abren condiciones o bucles. Como por ejemplo:

```
while (condición) {
        instrucción;
}
```

En comparación con otros estilos, como el de *BSD/Allman*, véase en el siguiente ejemplo:

```
while (condición)
{
        instrucción;
}
```

La única diferencia, es que en el primero la llave abre en la misma línea donde se escribe la condición, mientras que el segundo la abre una línea más abajo.

Aunque el segundo estilo puede parecer más claro y conciso, se ha utilizado el primero para ahorrar líneas innecesarias ya que los *scripts* desarrollados son de gran extensión, son 6 y 1 de ellos contiene unas 1500 líneas.

La empresa también utiliza un estándar respecto al nombre de las funciones y las variables. Estos nombres empezarán por minúscula y si contienen más de una palabra, éstas empezarán por mayúscula exceptuando la primera. Por ejemplo: variable *numeroAlbaran*, función *viewClick*.

7.3 Pruebas unitarias

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

Las pruebas unitarias se han efectuado en todos los formularios del módulo, exceptuando las dos últimas para el formulario de "Generación de consumos", ya que no es un formulario de mantenimiento y no guarda información en la base de datos.

Las pruebas unitarias que se han realizado son las siguientes:

- **Validar campos obligatorios:** Se comprueba en todos los formularios que los campos obligatorios estén completos. Antes de grabar un registro la aplicación no dejará que se continúe con el proceso, saltará un mensaje de error.
- **Comprobar formato correcto en los campos:** Se comprueba que cada campo tenga su correspondiente formato, algunos sólo admiten texto, otros números, etc. Con las máscaras de entrada y salida se soluciona este problema.
- **Actualizar información de los campos:** Se comprueba que sea posible cambiar la información de algunos campos y grabar el registro sin problemas.
- **Comprobación en la base de datos:** Se comprueba que la información actualizada en la prueba anterior se haya actualizado correctamente en la base de datos.

7.4 Pruebas funcionales

Las pruebas funcionales son las que permiten asegurar que los componentes de la aplicación realizan las funcionalidades especificadas para el módulo.

En este caso, solamente se han efectuado pruebas funcionales en el proceso de generación de consumos, ya que este proceso puede llegar a testear todos los formularios del módulo.

Las pruebas funcionales realizadas son las siguientes:

- **Comprobar prioridad del importe exento:** Se comprueba si el proceso de generación de consumos respeta el orden de prioridades establecido para el importe exento.
  Para comprobarlo se han completado todos los campos de importe exento en todos los formularios donde figure este campo. Por orden de prioridad, estos serían los campos especificados:
  - Formulario "Servicios por cliente": Importe exento a nivel de cliente por servicio.
  - Formulario "Servicios por cliente": Importe exento a nivel de cliente.
  - Formulario "Servicios": Importe exento a nivel de servicio.

- Formulario “Empresa y sección”: Importe exento a nivel de sección.
- Formulario “Empresa y sección”: Importe exento a nivel de empresa.

Una vez fijados los valores en todos los campos, se ha ejecutado el proceso de generación de consumos y se ha comprobado si a un mismo cliente se le ha dejado libre de pagar por superar el importe de compras correctamente, siguiendo el orden de prioridades fijado.

- **Comprobar prioridad del *check* exento:** Se comprueba exactamente el mismo caso que en la prueba anterior, pero con el marcaje del *check* exento. Dependiendo de en qué formulario esté marcado se seguirá el mismo orden de prioridades que se especifica en la prueba funcional anterior.

- **Comprobar prioridad de series a no tratar:** Se comprueba, como en el caso anterior que el resultado obtenido en el proceso de generación de consumos sea el correcto, de acuerdo con las prioridades de las series de albarán a no tratar en el procedimiento. Estas series estarán definidas en los siguientes formularios, por orden de prioridad:
  - Formulario “Servicios”: Series a no tratar por servicio.
  - Formulario “Empresa y sección”: Series a no tratar por empresa.

- **Comprobar filtrado de tipos de cliente:** Se comprueba que los tipos de cliente definidos en el formulario de servicios encajen correctamente con el resultado que devuelve la generación de consumos. Es decir, que en la solución sólo podrán existir clientes que sean del tipo especificado.

- **Comprobar filtrado de series a no tratar:** Se comprueba que en el resultado no figuren albaranes de los clientes que son resultado del proceso y que pertenezcan a las series especificadas en el módulo.

- **Comprobar filtrado de intervalos horarios:** Se comprueba que la fecha de los albaranes de los clientes que son resultado del proceso, esté dentro de los intervalos horarios del servicio que se vaya a facturar. Por ejemplo, si la fecha y hora del albarán es el 20/10/2010 09:00:00 y éste día es miércoles, el servicio ha de estar disponible los miércoles y en un rango de horas que incluya las 9 de la mañana.

- **Comprobar filtrado de documentos y subtipos de gestión:** Se comprueba que los documentos de gestión del albarán de los clientes que son resultado del proceso, coincidan con los documentos de gestión especificados en el módulo. Lo mismo para los subtipos de gestión, aunque en este caso pueden ser a tratar o a no tratar, por lo que

deberían coincidir en el primer caso y no coincidir en el segundo.

- **Comprobar funcionamiento campos calculados:** Se comprueba que en el resultado de la generación de consumos, en el formulario "Consumos", los campos calculados funcionen correctamente. Se han modificado los valores de estos campos y se ha probado si el campo resultado se ha calculado correctamente.

7.5 Pruebas de integración

Las pruebas de integración tienen como finalidad probar el funcionamiento conjunto de la aplicación. Se asegurará que las pruebas testeadas en los puntos anteriores funcionan correctamente de forma conjunta.

Las pruebas de integración que se han realizado son las siguientes:

- **Variable de entorno:** Se comprueba que cuando se selecciona o se cambia la empresa activa, se filtrará el código de la empresa para mostrar la información correcta en los formularios que dependan de la variable.
- **Cambio de registro:** Se comprueba que al cambiar los datos de un campo obligatorio en un formulario y éste sea clave primaria de la tabla donde se guarda la información, automáticamente se cargue en el formulario el registro que pertenezca a este nuevo campo escrito y si no existe que la aplicación dé la opción de crear un nuevo registro.
- **Relación entre paneles:** Se comprueba que al seleccionar una fila en un *grid* de un panel que esté relacionado con otro panel, este último cargará la información adecuada a la fila que se haya seleccionado.

7.6 Pruebas de rendimiento

Las pruebas de rendimiento son las pruebas que se realizan, desde una perspectiva, para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo.

El único proceso lo suficientemente complejo como para necesitar una prueba de rendimiento, es la generación de consumos. Este proceso ejecutará varias sentencias SQL y otros cálculos complejos para conseguir el resultado final.

Se han hecho dos pruebas de rendimiento:

- **Generación de consumos para menos de 100 clientes:** Se ejecuta este proceso para un máximo de 100 clientes. El proceso se ejecuta sin problemas y tarda en mostrar el resultado 5 segundos.

- **Generación de consumos para todos los clientes:** Se ejecuta este proceso para todos los clientes que figuran en la base de datos, aproximadamente unos 1000 clientes. El proceso se ejecuta sin problemas y tarda en mostrar el resultado un poco menos de 1 minuto.

Conclusiones

8.1 Objetivos conseguidos y no conseguidos

Podemos afirmar que todos los objetivos propuestos han sido cumplidos con éxito. Aunque hay un objetivo que se quedó fuera del alcance del proyecto porque necesitaba de otra aplicación previa para su funcionamiento, trabajo que no estaba dentro de mi proyecto. Se trata del cargo de servicios en albaranes. Para este objetivo era esencial que el módulo de generación de albaranes estuviera operativo, pero como este proyecto finalizó antes de que se realizara la aplicación, el objetivo se desestimó del alcance.

Hablando del resto de objetivos, se ha cumplido correctamente la inserción y mantenimiento de servicios en la base de datos, gracias al formulario "Servicios" y otros parámetros y reglas de gestión. La relación entre servicios y clientes ha sido establecida con el formulario "Servicios por cliente", muy importante para la facturación de servicios. Se generalizaron las características principales de los servicios por empresa en el formulario "Empresa y sección." Se consiguió agilizar el proceso de generación de consumos con éxito y mantener estos consumos generados. Y por último un objetivo secundario, la generación de listados, se acabó llevando también a cabo.

El cumplimiento de todas estas funcionalidades hará de la Facturación de servicios un módulo completo y estable, con una interfaz intuitiva y con la posibilidad de poder introducir nuevas mejoras y ampliaciones.

Concluyendo, el proyecto final de carrera se ha desarrollado siguiendo los estándares de UNIT4, el módulo es adaptable a la plataforma karat y a la interfaz de directrices *Walnut*, tanto visualmente como también en la codificación.

8.2 Ampliaciones

La única ampliación propuesta, según lo visto en el punto anterior, sería el cargo de servicios en albaranes y la generación de los mismos.

El cargo de servicios en albaranes consiste en que una vez los servicios se han facturado en ciertos albaranes, estos albaranes quedarían marcados para que no se volvieran a facturar para este servicio en concreto. Para el desarrollo de esta funcionalidad se ha adelantado ya una tabla más en la base de datos que guardará cada albarán con sus respectivos servicios facturados correctamente.

Para su implementación haría falta un nuevo módulo, la generación de albaranes. Al generar un albarán se mostraría en vista de impresión para poder tenerlo en papel y allí figuraría como que ya se le ha facturado dicho servicio, de forma que no se volverá a hacer nunca más para dicho albarán.

8.3 Desviaciones de la planificación

Se ha seguido en gran medida la planificación inicial del proyecto. Pero cabe destacar, que aunque la fecha de finalización sea la misma que la de la cronología inicial, la fase de desarrollo se ha visto incrementada y ha necesitado más tiempo para su realización mientras que la sección de pruebas se ha acortado sobre lo que se esperaba inicialmente. También ha sido suprimidas algunas de las tareas iniciales y se han añadido otras nuevas.

En la siguiente tabla se muestran las modificaciones temporales que han sufrido algunas de las tareas respecto a la planificación inicial.

Nombre de la tarea	Duración planificada	Duración real
Desarrollo	**38 días**	**44 días**
Preparación del entorno de desarrollo	2 días	3 días
Configuración de las librerías a importar	1 día	1 día
Desarrollo de las clases	5 días	5 días
Desarrollo de las funcionalidades	30 días	35 días
Test y pruebas	**22 días**	**17 días**
Pruebas unitarias	3 días	1 día
Pruebas funcionales	3 días	3 días
Pruebas de integración	3 días	1 día
Pruebas de rendimiento	-	1 día
Documentación de las pruebas	2 días	2 días
Aprobación de las pruebas	1 día	1 día
Corrección de errores	10 días	8 días
Implantación	**6 días**	**5 días**
Implantación del módulo al producto	5 días	5 días
Generación del archivo JAR	1 día	-

Tabla 13: Desviaciones de la planificación

En la siguiente imagen, vemos como ha quedado el calendario temporal final del proyecto en forma de diagrama de Gannt.

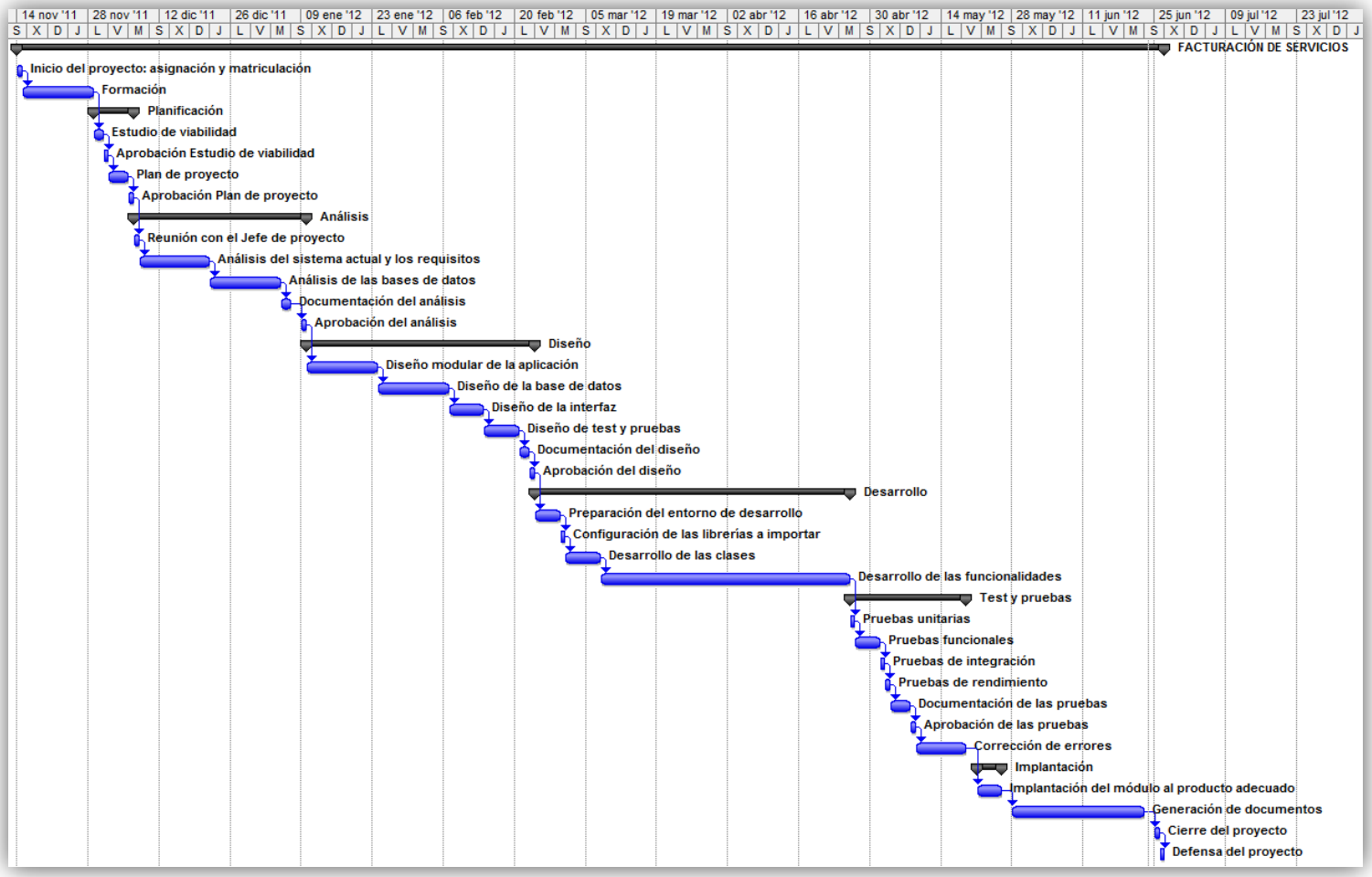

Figura 23: Desviaciones de la planificación. Calendario temporal

8.4 Valoración personal

En referencia al ámbito académico, valoro la experiencia de hacer un proyecto desde cero, pasando por todas sus fases y presentaciones necesarias. Crear una aplicación desde el principio me ha servido también para mejorar notablemente mi nivel de programación en lenguaje Java y SQL para las bases de datos. Cabe destacar que finalizar la ingeniería informática de gestión es un gran paso para la culminación de mis estudios, que tengo previsto continuar con un máster.

Conseguir experiencia en una empresa como UNIT4, una de las empresas de desarrollo de software ERP más importantes a nivel internacional, ha sido un gran punto a favor en mi carrera ya que la experiencia en empresas de este tipo es muy valorable en el mundo laboral. También he despertado cierto interés por lo que se refiere al sector de la consultoría ERP y cabría la posibilidad de que decantase mi vida profesional en esa dirección.

Finalizando, mi valoración personal hacia esta experiencia es muy positiva y me alegro mucho de haber tomado esta opción, ya que la situación laboral actual está muy complicada en estos momentos.

Bibliografía

En este apartado se detallan las referencias bibliográficas que se han consultado para el desarrollo del proyecto. Se ha consultado información en la web y en el documentación interna de la empresa.

Documentación web:

- **API de Java**: utilizada para la codificación del módulo.
  http://docs.oracle.com/javase/1.5.0/docs/api/
  Última entrada: 04/05/2012

- **Todo Java**: página web utilizada para consultas acerca de la codificación.
  http://www.todojava.awardspace.com/
  Última entrada: 04/05/2012

- **Wikipedia**: enciclopedia utilizada para distintas consultas.
  http://www.wikipedia.org
  Última entrada: 22/06/2012

- **WordRefence**: diccionario utilizado para la escritura de la memoria.
  http://www.wordreference.com/
  Última entrada: 22/06/2012

Documentación interna de la empresa:

- **API de karat**: utilizada para la codificación de la aplicación.
- **Documentación de karat Escritorio**: documentación utilizada para desarrollo.
- **Ayuda de karat**: libro utilizado para el diseño.
- **Documento de trabajo**: documento realizado por el jefe de proyecto, donde se especifican los requisitos funcionales del módulo.

Contenido del CD

- Memoria del proyecto en formato PDF.
- Vídeo de la aplicación en funcionamiento.
- Repositorio de *scripts* que componen el código fuente del proyecto.
- Exposiciones realizadas en la empresa.
- Documentos de trabajo del módulo.
- Listados de los consumos generados en formato PDF.
- Manuales de karat.

Agradecimientos

Primeramente me gustaría agradecer tanto a la UAB como a UNIT4, la oportunidad que me han brindado con este convenio de prácticas de realizar el proyecto final de carrera.

Dar las gracias a Javier Marina, que desde Zaragoza ha controlado todos los requisitos y especificaciones del proyecto. También al analista Marc Bardina, que me ha ayudado durante todo el diseño y desarrollo del proyecto.

A mi tutor Jordi Pons, por su rapidez y efectividad en la corrección de la memoria del proyecto.

Y por último, agradecer a mi pareja la ayuda que me ha prestado al preparar todas las exposiciones que he tenido que presentar.

David Manzano Barba

Sabadell, **junio** de **2012**