8 Ingeniería del software: Diseño

En el capítulo anterior se ha realizado el análisis del problema mediante la especificación. Sin embargo, el resultado de la especificación no es suficiente para llevar a cabo la implementación del sistema, ya que ésta se orienta a la descripción del problema, independientemente de las tecnologías a utilizar.

Es este capítulo, es donde se estudian las diferentes posibilidades de abordar la solución al problema. Esta solución no es única por lo que se puede encontrar distintas soluciones correctas al mismo problema.

Para guiar la construcción de la solución idónea en cada caso se tendrán que adoptar ciertos criterios de diseño y establecer las tecnologías y estructura generales del sistema siguiendo los mismos. Las primeras decisiones importantes que se tendrán que tomar son respecto a:

1. La arquitectura del software y la estructura general del sistema.
2. Las tecnologías a utilizar durante su desarrollo.

La arquitectura del software describe los subsistemas y componentes de un sistema software y las relaciones entre éstos. Un componente es una parte encapsulada de un sistema software, son los diferentes bloques que constituyen el sistema. A nivel de lenguaje de programación se presentan como: módulos, clases, objetos o conjuntos de funciones relacionadas.

Escoger la arquitectura del software utilizada, implicar tomar un conjunto de decisiones importantes que afectan a los siguientes puntos:

1. La organización del sistema de software.
2. La selección de los elementos estructurales y sus interfaces.
3. El comportamiento de los elementos estructurales.
4. La composición de elementos en subsistemas más grandes.

Es por esto que el capítulo comienza estudiando y evaluando la arquitectura del sistema para establecer los diferentes módulos que lo compondrán. Posteriormente se identificarán las distintas tecnologías disponibles que nos permitan realizar la implementación para dicha arquitectura, y se compararán sus diferentes características para escoger la más apropiada en este caso.

8.1. Factores de calidad

Antes de adentrarnos en el diseño del sistema conviene tener claros los factores de calidad que nuestro sistema debe satisfacer, ya que de ellos dependerá en gran medida la elección de los patrones arquitectónicos.

Extensibilidad: posibilidad de ampliar el sistema con nuevas funcionalidades o mejoras en los componentes, sin que ello suponga rehacer el sistema.

Interoperabilidad: capacidad de dos o más entidades de software de intercambiar funcionalidades y datos.

Reusabilidad: Capacidad de los elementos del software de servir para la construcción de aplicaciones diferentes.

Fiabilidad: Capacidad del sistema para hacer las tareas tal como han sido especificadas. Robustez frente al uso incorrecto y tratamiento de errores inesperados.

Eficiencia: Capacidad para usar el menor número de recursos posibles, como tiempo, memoria, etc. En un sistema en el que se van a tratar muchos datos en memoria es muy importante. Las operaciones han de ser lo más rápidas posibles.

Facilidad de uso: Facilidad en que personas con distintos niveles culturales aprenden a usar el programa y a aplicarlo para resolver problemas.

Puntualidad: Capacidad para entregar el sistema cuando los usuarios quieren. Este proyecto no ha sido una excepción. Además de tener el límite temporal de la entrega del PFC se contaba con la fecha de entrega al cliente.

8.2. Arquitectura del sistema

Para definir la arquitectura del sistema se ha utilizado el patrón arquitectónico de tres capas. Esto significa que a cada capa se le otorgan unas responsabilidades, que sólo ella puede hacer, y que tendrá comunicación con el elemento superior, que puede ser otra capa o el usuario, y el elemento inferior, otra capa o el dispositivo con los datos, en este caso la base de datos. Este método destaca por su cambiabilidad y portabilidad, además de tener una separación de responsabilidades clara y organizada. Por el contrario la eficiencia disminuye ya que se realizan llamadas entre capas que a menudo se podrían hacer en una misma capa que dispondría de todos los datos necesarios.

Uno de los motivos de elección de este patrón ha sido que es particularmente interesante cuando queremos proteger los datos del usuario, cuando tenemos una base de datos y queremos que el usuario sólo tenga acceso a la interfície. Además el código es mucho más inteligible, y es probable que el programa tenga continuidad pues es susceptible de ser complementado con otras funcionalidades en el futuro.

8.2.1. Capa de presentación

La capa de presentación conoce cómo presentar los datos a los usuarios, pero ignora qué transformaciones son necesarias para dar respuesta a sus peticiones.

Esta capa es la encargada de interactuar con el usuario. Recoge sus peticiones, ordena la ejecución de acciones como respuesta y presenta los resultados. Aquí se tratan ventanas, diálogos, menús, botones, etc.

En esta capa se ha usado el patrón vista-controlador lo que permite separar en clases las responsabilidades de la gestión de la interacción con el usuario y de la comunicación con la capa del dominio.

Dado la gran cantidad de vistas necesarias para cubrir los requisitos impuestos por el cliente, he decido no incluir en este apartado la explicación de cada una de las vistas y las funcionalidades que cumple, evitando así repetir información que aparece más adelante. Esta información se puede ver con detalle en la sección del manual de usuario que se incluye en los anexos de la memoria (capítulo 13).

8.2.2. Capa del dominio

La capa del dominio conoce cómo satisfacer las peticiones del usuario, pero ignora dónde se guardan los datos y como presentarlos al usuario.

Esta capa se entera de los acontecimientos, controla su validez, cambia el estado del dominio, ejecuta las acciones requeridas, obtiene resultados y comunica respuestas.

Esta capa está compuesta básicamente por los objetos del dominio (los mismos que se definieron en la especificación), diccionarios de objetos para poder acceder por identificador a algunos de ellos y por controladores, que son los que interceptan los acontecimientos de la capa de presentación y ejecutan las acciones correspondientes.

En general los controladores usados en la capa del dominio han controladores fachada. La idea principal es la de ocultar todo lo posible la complejidad del sistema, el conjunto de clases o componentes que lo forman, de forma que solo se ofrezca unos pocos punto de entrada al sistema tapado por la fachada.

Una ventaja más de usar una clase fachada para comunicar las dos partes o componentes, es la de aislar los posibles cambios que se puedan producir en alguna de las partes. Al cambiar, por poner un ejemplo, el medio de comunicación o de almacenamiento de una de las partes, la otra, que por ejemplo hace la presentación, no tiene porque enterarse, y viceversa.

8.2.3. Capa de gestión de datos

La capa de gestión de datos conoce dónde y cómo están guardados los datos, pero ignora cómo tratarlos.

Esta capa permite que el dominio pueda ignorar donde se encuentran los datos. Las funciones dependen del sistema gestor de datos que se use.

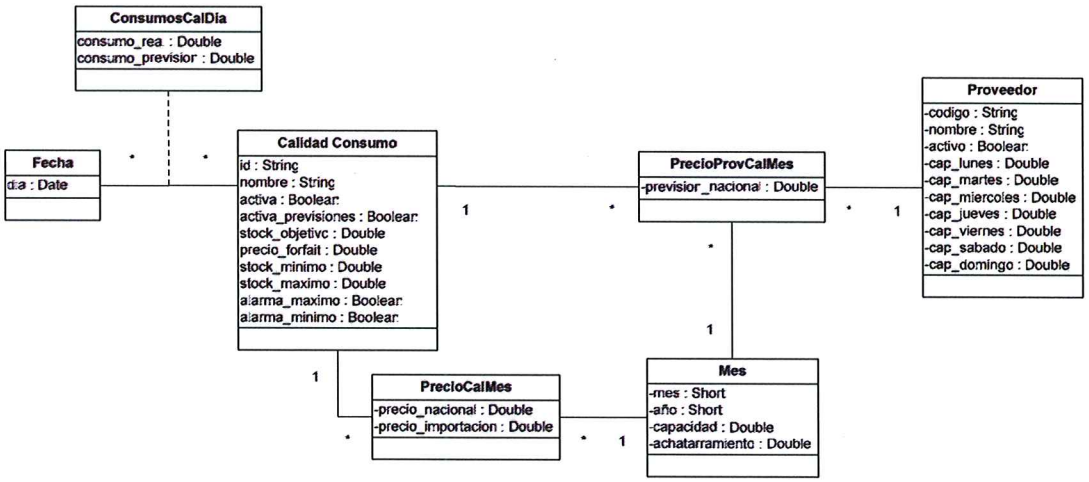
Básicamente posee varias clases encargadas de hacer las sentencias de consulta o modificación sobre la base de datos y una encargada de abrir y cerrar la conexión con la BD y de lanzar estas consultas.

8.3. Modelo estático – Diagrama de clases

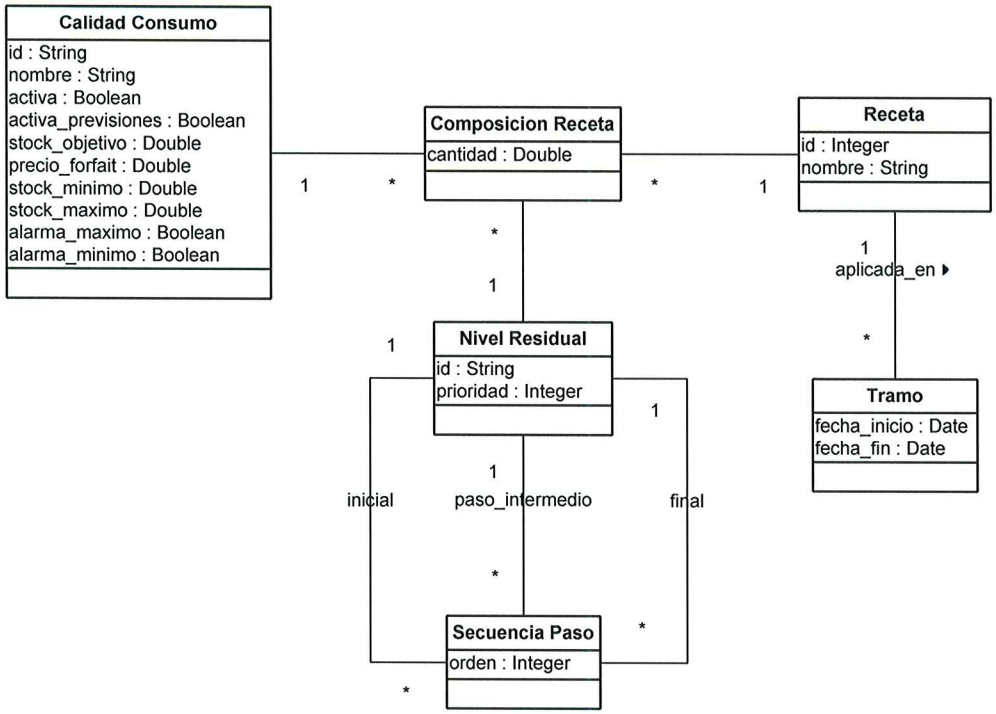
Durante la fase de diseño se ha procedido a normalizar el modelo conceptual de clases de dominio creado durante la etapa de especificación (capítulo 6). Los principales cambios realizados han sido el tratamiento de las asociaciones n-arias y las clases asociativas. Estos cambios se deben a la limitación actual de la tecnología orientada a objetos ya que no permite implementar directamente todos los conceptos usados durante la especificación.

A continuación se encuentran dos ejemplos de los cambios realizados.

En el caso de las calidades de consumo se puede apreciar que en el proceso de diseño se ha procedido a normalizar el modelo creado durante la fase de especificación.



Al igual que nos ha ocurrido en el caso de las calidades de consumo, en este apartado también se puede apreciar que en el proceso de diseño se ha procedido a normalizar el modelo.



8.4. Modelo relacional de la Base de datos

El proceso de diseño de la base de datos consta básicamente de dos pasos, el diseño del modelo lógico y la construcción del modelo físico a partir del modelo lógico.

El modelo lógico es una traducción del modelo conceptual al modelo relacional que a través de los diagramas Entity - Relationship muestra los conceptos que van a ser almacenados en la base de datos. El resultado es un conjunto de tablas relacionales con atributos en las columnas y registros y valores en las filas.

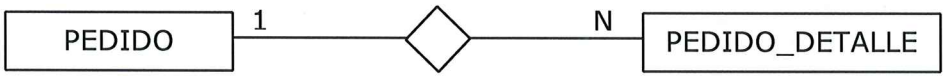
El modelo físico es la adaptación del modelo lógico a una implementación concreta de un sistema gestor de bases de datos.

Zona (*idZona*, descripcion, diasCarga, diasNavegacion)

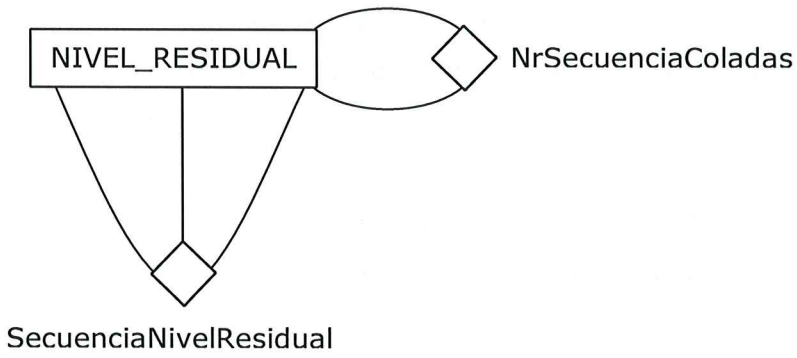
Origen (IdOrigen, descripcion, idZona)
- *idZona es una clave foránea a Zona (idZona)*

Nacional (pedido)
- *Pedido es una clave foránea a Pedido (IdPedido)*

Importacion (*idPedido*, nombreBarco, diasCarga, diasNavegacion, diasDescarga, fechaEstimadaIni, fechaEstimadaFin, fechaLaycanIni, fechaLaycanFin, fechaBL, pesoBL, fechaETA, fechaAtraque, fechaCerrado, motrarLevante)
- *idPedido es una clave foránea a Pedido (IdPedido)*



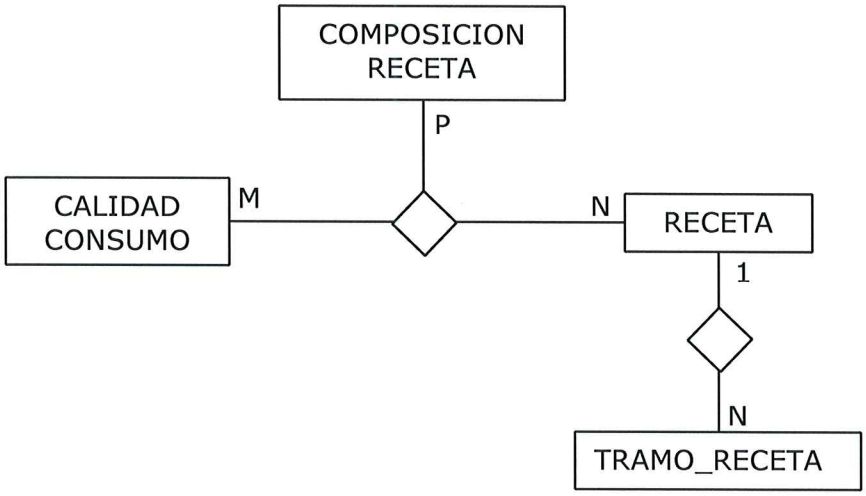
PedidoDetalle (IdDetalle, *idPedido*, calidadConsumo, calidadCompra, nombreCalCompra, toneladas, precio)
- IdDetalle es una clave artificial (surrogate de usuario)
Clave alternativa: (pedido, calidad, calidadCompra)
idPedido es una clave foránea a Pedido (IdPedido)



NivelResidual (IdNivel, orden, ordenPantallaResults)

NrSecuenciasColadas (*idNrTeorico, idNrReal,* posicion, fechaIni)
- *idNrTeorico* es una clave foránea a NivelResidual (IdNivel)
- *idNrReal* es una clave foránea a NivelResidual (IdNivel)

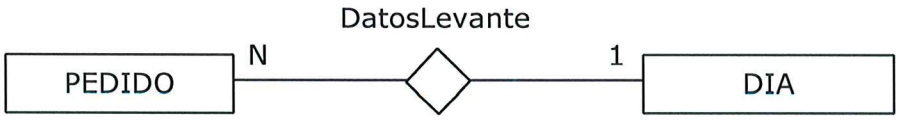
SecuenciaNivelResidual (*idNrOrigen, idNrDestino,* posicion, secuencia)
- *idNrOrigen* es una clave foránea a NivelResidual (IdNivel)
- *idNrDestino* es una clave foránea a NivelResidual (IdNivel)



Receta (Id)

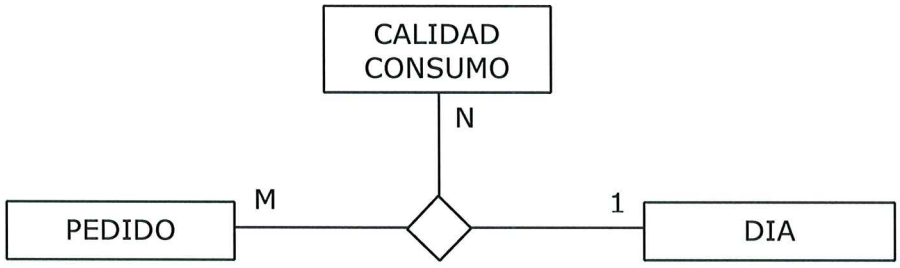
ComposicionReceta (*idReceta, idNR, idCalidad*, toneladas)
- *idReceta* es una clave foránea a Receta (idReceta)
- *idNR* es una clave foranea a NivelResidual (idNivel)
- *idCalidad* es una clave foranea a CalidadConsumo (idCalidad)

TramoReceta (diaInicio, diaFin, *idReceta*)
- *idReceta* es una clave foránea a Receta (idReceta)



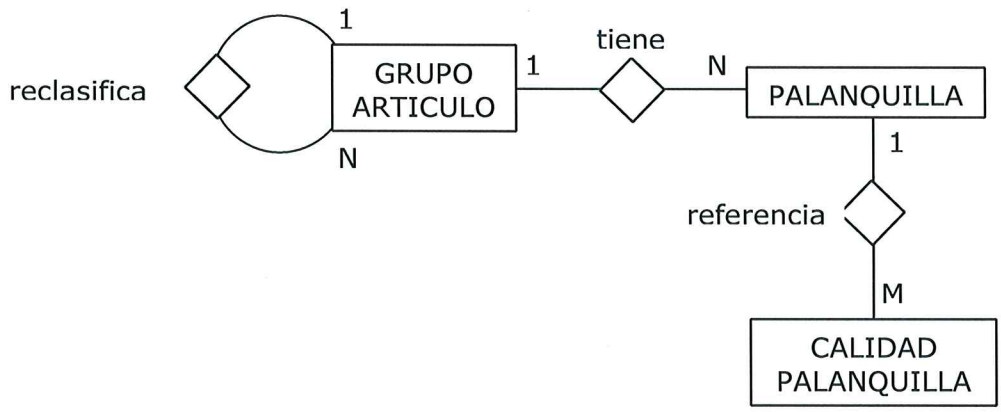
DatosLevante (*idPedido*, dia, cantidad, validado)

- *idPedido es una clave foránea a Pedido (idPedido)*



EntradasDiaProv (*idPedido, idCal*, dia, mercado, total)

- *idPedido* es una clave foránea a Pedido (idPedido)
- *idCalidad* es una clave foránea a CalidadConsumo (idCalidad)



GrupoArticulo (IdGrupo, descripcion, stockObjetivo, *idGrReclasificado*, porcetajeAchatarramiento, porcentajeReclasificacion)

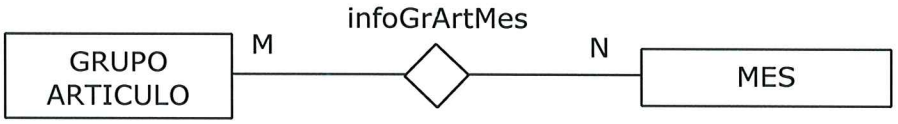
- *idGrReclasificado es una clave foránea a* GrupoArticulo *(idGrupo)*
- Clave alternativa: descripcion

CalidadPalanquilla (IdCalPal, descripcion, *idGrupo*)

- *idGrupo es una clave foránea a* GrupoArticulo *(idGrupo)*
- Clave alternativa: descripcion

Palanquilla (IdPalanq, descripcion)

- *idGrupo es una clave foránea a* GrupoArticulo *(idGrupo)*
- Clave alternativa: descripcion

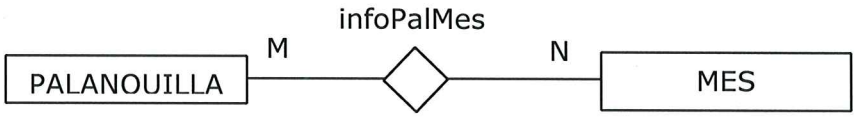


InfoGrArtMes (*idGrArt, idMes*, consumos, produccion, reclasManual)

- *idGrArt es una clave foránea a* GrupoArticulo *(idGrupo)*
- *idMes es una clave foránea a* Mes *(idMes)*

Mes (idMes, mes, año, capacidad, achatarramiento)

- Clave alternativa: mes, año



EiInfoPalanqMes (*idPalanq, idMes*, produccionCorregida)

- *idPalanq es una clave foránea a* Palanquilla *(idPalanq)*
- *idMes es una clave foránea a* Mes *(idMes)*

CalidadConsumo (idCal, descripción, stockObjetivo, precioForfait, cantidadMin, cantidadMax, alarmaMin, alarmaMax, ordenReceta, ordenLlegadaDmn, tipo, orden, activa, activaPrev)

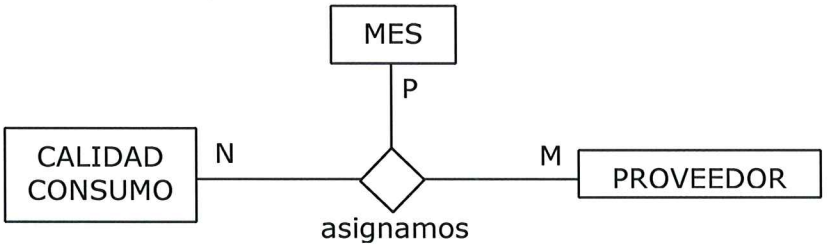
- Clave alternativa: descripcion

InfoCalMes (*IdCal, idMes*, precioEntCalNac, precioEntImp, tnPendientes

- *idCal es una clave foránea a* CalidadConsumo *(idCal)*
- *idMes es una clave foránea a* Mes *(idMes)*

InfoCalDia (*idCal, dia*, stockMuelle, stockDia, compraEspecial, consumoPrevisto, consumoReal, entradaPrevista)

- *idCal es una clave foránea a* CalidadConsumo *(idCal)*



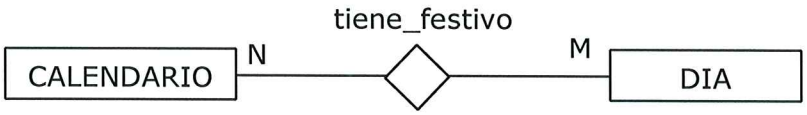
PreciosEntNac (*IdCal, idProv, idMes*, precio)

- *idCal* es una clave foránea a CalidadConsumo (idCal)
- *idProv* es una clave foránea a Proveedor (idProv)
- *idMes* es una clave foránea a Mes (idMes)

BarcoEstibar (IdBarco, destino, barco, cantidad, diaIni, diaFin, tipo, FechaCerrado)

Parámetros (IdParametro, descripcion, valor, valor2, valorString, fecha)

Escenarios (IdEscenario, descripcion, version, ultimoGuardado)

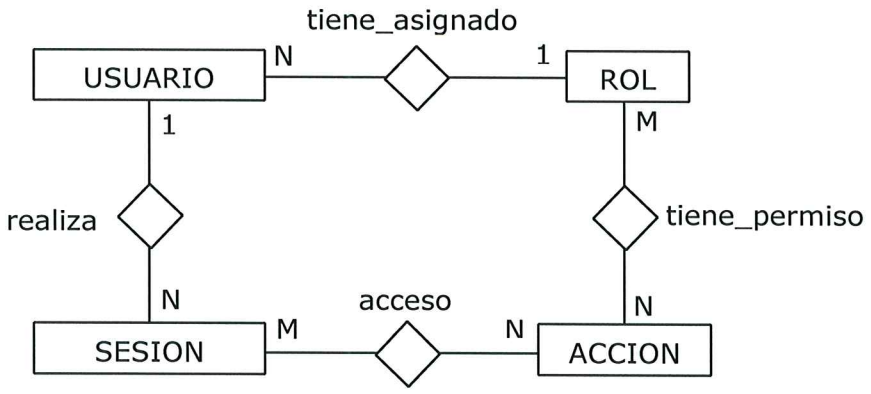


Calendarios (idCal, descripción)

- *idProv es una clave foránea a* Proveedor *(idProv)*
- Clave alternativa: descripcion

Festivo (idCal, dia, tipoParada)

- *idCal es una clave foránea a* CalidadConsumo *(idCal)*



Usuario (idUser, nombre, estado, *idRol*, login, contraseña, ultimo_acceso

- Clave alternativa: nombre
- Clave alternativa: login
- *idRol es una clave foránea a* Rol *(idRol)*

Rol (idRol, descripcion)

- Clave alternativa: descripcion

Accion (idAccion, descripcion)

- Clave alternativa: descripcion

Permiso (*idRol*, *idAccion*, tipo)
- *idRol es una clave foránea a* Rol *(idRol)*
- *idAccion es una clave foránea a* Accion *(idAccion)*

Sesion (*idUser*, horaEntrada, horaSalida)
- *idUser es una clave foránea a* Usuario *(idUser)*

Acceso (*idSesion*, *idAccion*, hora, idPermiso)
- *idSesion* es una clave foránea a Usuario (idSesion)
- *idAccion* es una clave foránea a Accion (idAccion)

8.4.2. Diseño físico de la base de datos

De acuerdo con lo explicado con anterioridad, el modelo físico es una implementación concreta del modelo lógico en un SGBD concreto. En este caso se ha utilizado Oracle 8i como SGBD, tal como se indicaba en los requisitos no funcionales.

Oracle tiene varias restricciones y particularidades sobre el lenguaje estándar SQL '99, sin embargo, es capaz de interpretarlo casi en su totalidad. Es gracias a esta posibilidad que se ha podido utilizar una descripción estándar en la creación de la base de datos. Para describir el modelo físico se ha usado directamente un script de creación de la base de datos con las sentencias SQL necesarias.

9 Implementación

Ya hemos comentado con anterioridad que el lenguaje de programación escogido para implementar la aplicación ha sido Microsoft Visual Basic .Net y para la base de datos emplearemos Oracle 8i.

Visual Basic .Net lleva el nombre de un lenguaje de programación con bastantes años de antigüedad, pero se ha visto gravemente modificado desde sus inicios, y en esta versión es más parecido a otros lenguajes de la misma plataforma .Net. Microsoft ha creado una serie de lenguajes dentro de esta plataforma, definiendo un estándar y dotando a todos estos lenguajes de características comunes. Una de estas características es la orientación a objetos, que en esta versión de VB es mucho más sencilla de utilizar que en anteriores versiones.

Por el contrario se ha comprobado que VB .Net no es un lenguaje demasiado apropiado para aplicar el patrón de arquitectura en tres capas, ya que los formularios deberían constituir únicamente la capa de presentación y sin embargo invitan a operar también como capa de gestión. No obstante se ha respetado ese patrón por considerar que tiene ventajas interesantes. Con todo, teniendo en cuenta que VB no es un lenguaje muy eficiente, y que la arquitectura en 3 capas tampoco fomenta la eficiencia, el programa puede pecar en ese apartado, aunque gana en coherencia y organización. De todos modos consideramos que el uso que se le dará a esta aplicación no requerirá un rendimiento espectacular, ya que pocos usuarios lo utilizarán en un mismo equipo y las llamadas no serán continuas.

9.1. Normas de codificación

Dentro de ALG disponemos de varios documentos donde se intenta establecer unas directrices para que la codificación este unificada y sea más clara para todos.

Una de estas normas hace referencia a la nomenclatura de variables, para la que usamos una variación de la notación húngara. Por ejemplo, para definir un booleano, utilizamos como prefijo "bln". Así tenemos variables como blnActivado que nos indica si un elemento se encuentra activo o no.

Otras de las normas es el uso de comentarios y "summaries" en las cabeceras y cuerpos de los métodos donde se explican que contiene los parámetros y cual es el cometido del método; de esta forma se facilita el trabajo a cualquier persona que posteriormente necesite modificar el código.

Disponemos también de un portal interno donde publicamos noticias y monografías sobre determinadas utilidades de manera que podamos aprender nuevas técnicas. También disponemos de una pequeña Wiki interna donde se introducen trozos de código útiles (como por ejemplo parsear un XML o serializar una clase) y donde también se encuentran explicaciones de los diversos módulos y sistemas que ha creado el departamento a lo largo de su historia.

9.2. Integración del modulo de previsiones

Para realizar la funcionalidad de previsión de llegadas de chatarra nacional, se ha realizado una integración de un módulo genérico de previsiones desarrollado por Fran Miguel Mangas un compañero de trabajo que al igual que yo es estudiante de la FIB y ha que ha presentado el modulo de previsiones que explicaré a continuación como proyecto de carrera.

El modulo nos permite definir los diferentes niveles de agregación para los cuales queremos solicitar la información. En nuestro caso tendremos dos niveles distintos: proveedor, calidad de consumo de chatarra.

Además de estos niveles de agregación el módulo de previsiones nos permite definir los niveles temporales para los cuales queremos generar las previsiones. En el caso de este sistema necesitamos generar tanto previsiones diarias usadas para el cálculo a corto plazo, como previsiones mensuales que serán usadas en el cálculo a medio plazo.

9.2.1. Dificultades encontradas

La dificultad está en que en cada fábrica o sistema tiene sus particularidades y en consecuencia el código no puede ser completamente idéntico. La solución ha sido un diseño respetando el principio abierto-cerrado. Este principio dice que un diseño correcto debe ser abierto para la extensión, es decir, el comportamiento de los módulos que forman sistema se puede extender para satisfacer nuevos requisitos. Pero a la vez, cerrado para la modificación, es decir que la extensión no implica cambiar el código de los módulos ya acabados.

Satisfacer el principio abierto-cerrado se consigue añadiendo código y no modificándolo. La forma concreta en este proyecto ha sido aplicar polimorfismo en los objetos del diseño. Las operaciones se aplican a varias clases de una jerarquía tal que su semántica depende de la subclase concreta donde se aplique. Es decir, las clases, concretamente los controladores que son los susceptibles de tener comportamientos variados, son heredadas. También se aplica el patrón de diseño método plantilla, útil cuando una operación en una jerarquía tiene un comportamiento común en todas las subclases y otro específico en cada una de ellas.

9.2.2. Pasos seguidos para realizar la integración

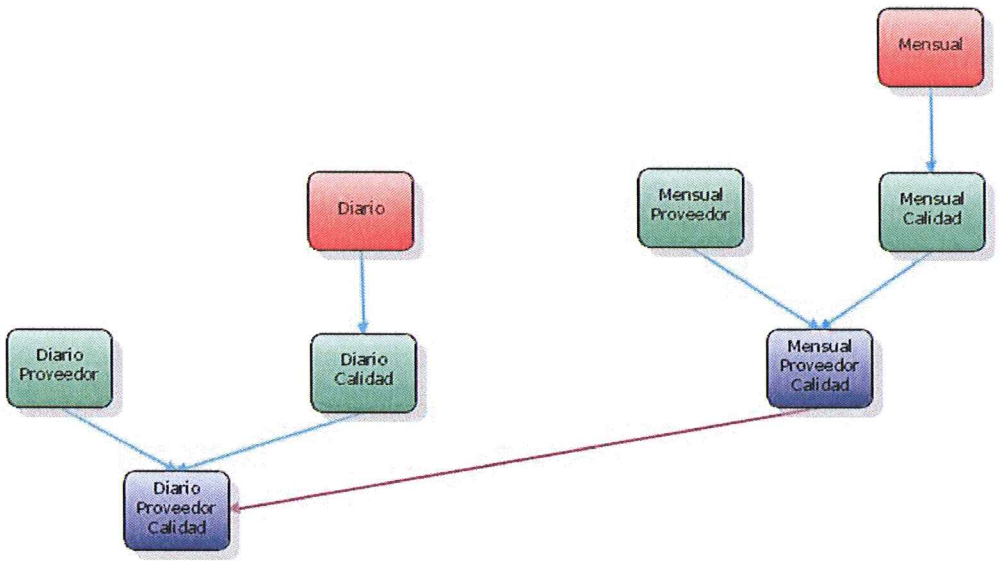
A continuación se muestran los pasos generales que se han seguido a la hora de integrar el sistema en el proyecto:

1. Definir estructura en fichero XML.
2. Cargar previsiones al cargar el sistema de gestión de la cadena de suministro.
   - 2.1. Inicializar las previsiones definiendo la conexión a la base de datos, las tablas de datos reales y de históricos, el número de meses desagregados hasta nivel diario y el número total de meses, el calendario a usar con los días festivos y los controladores concretos del dominio que se usarán. Estos serán superclases en caso de que tengan el mismo comportamiento al definido por defecto o subclases en caso de que tengan algún comportamiento específico.
   - 2.2. Crear nuevos meses y/o días en la base de datos cuando sea necesario.
   - 2.3. Cargar el esqueleto de previsiones. Primero el fichero XML y después los elementos de cada nivel.
   - 2.4. Lanzar un thread para permitir usar otras funcionalidades del SGCS excepto previsiones, mientras se continúan cargando.
     - 2.4.1. Propagar la información de los niveles inferiores (guardados en la base de datos) por toda la estructura.
     - 2.4.2. Si es necesario generar un nuevo mes de previsiones.
     - 2.4.3. Si hay nivel diario realizar el recálculo automático.
3. Insertar las vistas de previsiones y previsiones estándares donde sea necesario del menú del sistema de gestión.
4. Insertar en los maestros de elementos del SGCS (lugar donde los elementos se dan de alta, de baja o se modifican) las llamadas necesarias para crear y eliminar previsiones.

9.2.3. Estructura

La estructura de previsiones para este sistema es bastante simple. Lo único que se debe tener en cuenta es la entrega de chatarra por parte de los proveedores. Un aspecto a destacar es que esta llegada de chatarra es diaria y la evolución de inventario realizada por el SGCS para obtener la cantidad de palanquilla genera resultados diarios.

De estas líneas se desprende las dimensiones que forman la estructura. La primera es la temporal formada por los niveles diario y mensual. La dimensión de aprovisionamiento desagrega la información según los proveedores. Por último, la dimensión de calidad hace referencia al tipo de chatarra entrante. La chatarra se clasifica en distintos tipos según su calidad.



9.2.4. Particularidades

Uno de los requisitos exclusivos del sistema de reaprovisionamiento es sobre un proveedor y una calidad especiales. Éstos deben permitir verse en ciertos niveles de previsiones pero sus cantidades no deben ser agregadas junto con el resto de previsiones de la estructura. Tampoco deben poder modificarse. Para ello, las previsiones de este proveedor y esta calidad no han sido incluidas dentro de previsiones, es el sistema de reaprovisionamiento el que guarda en memoria los valores (muy pocos). El comportamiento específico tiene lugar en este caso en la capa de presentación. Se crea una clase ControladorFormularioPrevisiones_GSW que hereda del controlador de la capa de presentación ControladorFormularioPrevisiones. Esta clase tiene sobrescrita la función rellenarDatos que es la encargada de pintar las previsiones en la pantalla de previsiones. La función ahora, además de mostrar las previsiones, muestra los datos del proveedor y la calidad especiales. Como estas previsiones no son modificables no tiene sentido que aparezcan en la pantalla de previsiones estándares.

Otro requisito específico para esta fábrica tiene que ver con la funcionalidad de avanzar mes. Se ha de generar un nuevo mes de la misma forma que está definido, pero repartir a diario el mes que toca tiene variaciones. Aquí se ha de repartir entre los días del mes, respetando los festivos, pero teniendo en cuenta también un porcentaje de cada proveedor definido en una pantalla del sistema de reaprovisionamiento. Para ello se hereda el ControladorTemporal de la capa del dominio. En la superclase se encuentra definida la operación avanzarMes que llama a la función nuevoMes, con un comportamiento común en todas las subclases, y a la función repartirADiario. Esta función es abstracta y tiene definida su semántica en las subclases. Es ahí donde se satisface dicho requisito.

9.3. Conexión a la base de datos

La conexión a la base de datos se ha realizado mediante ODBC, un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, cuyo objetivo es hacer posible el acceder a cualquier dato de cualquier aplicación, sin importar qué SGBD (Sistema Gestor de Bases de Datos) almacene los datos.

Esta independencia se consigue al insertar una capa intermedia de gestión de Bases de Datos, entre la aplicación y el DBMS, tal como hemos explicado en el capítulo de diseño del sistema (capítulo 8). El propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el SGBD entienda. Para que esto funcione tanto la aplicación como el SGBD deben ser compatibles con ODBC, esto es que la aplicación debe ser capaz de producir comandos ODBC y el SGBD debe ser capaz de responder a ellos.

Para conectarse a la Base de Datos se crea una cadena de conexión que define los parámetros, ruta y características de la conexión según los datos que solicite el fabricante. Mediante esta cadena y las llamadas a la API del ODBC que contiene el driver específico de ODBC para Oracle podemos realizar la conexión con la base de datos.

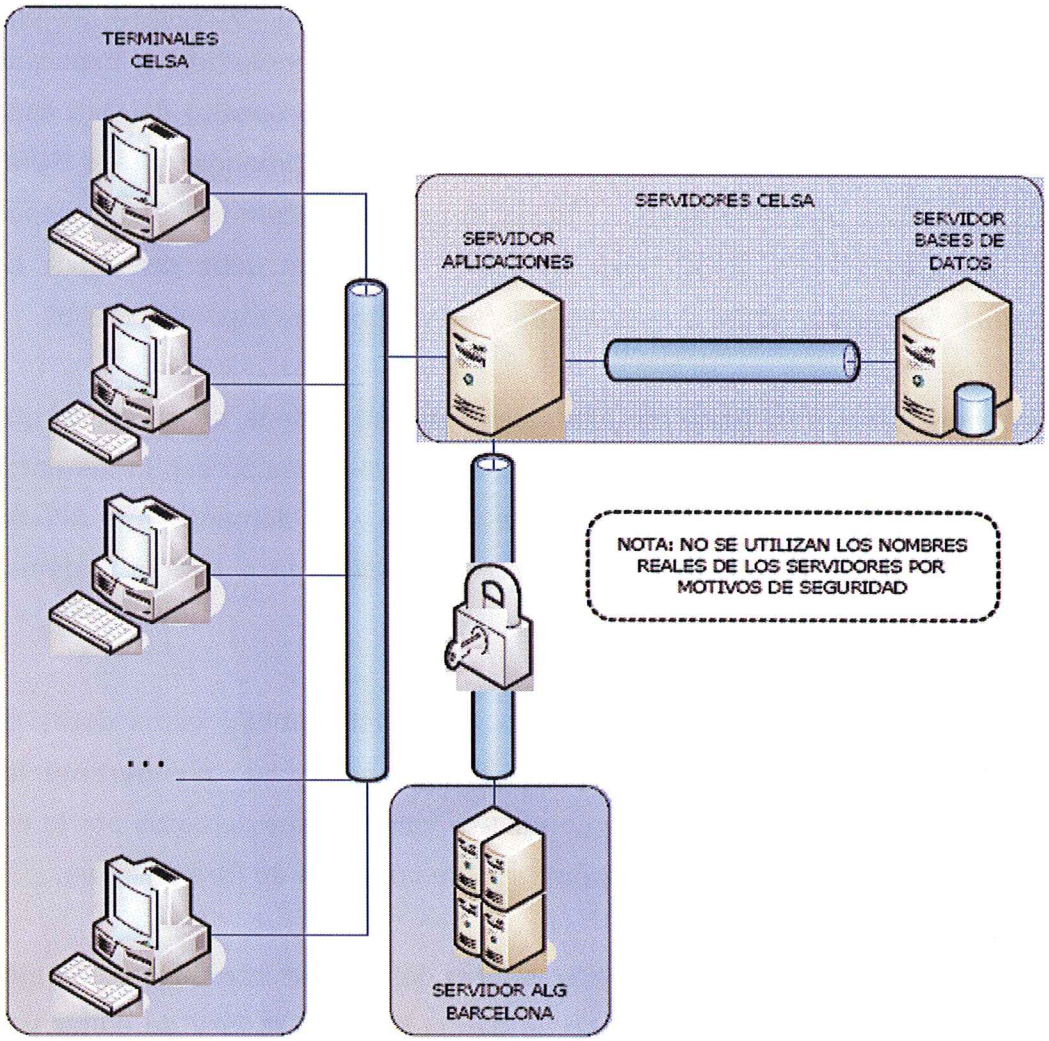
El problema encontrado es que interesaba poder cambiar estos datos de conexión a la base de datos de una forma rápida, ya que durante las pruebas del sistema en la oficina se usaban varias bases de datos por lo era interesante poder cambiar los parámetros de conexión de forma rápida.

La solución fue crear un fichero XML dentro del cual se introducen los datos necesarios para conectar a la base de datos. Un XML es fácil de editar y al iniciar el programa tan solo tenemos que recuperar los datos del XML y dar a escoger al usuario entre las diferentes conexiones contenidas en éste.

9.4. Integración en la red interna de Celsa

El sistema funciona de la misma manera que el resto de sistemas de información en Celsa (menos SAP). Los usuarios establecen sesiones de Terminal Server con el servidor de aplicaciones.

A continuación mostramos un esquema de la red de Celsa.



Las aplicaciones se ejecutan mediante sesiones de Terminal por exigencias del departamento de IT de Celsa que prefieren tener las sesiones centralizadas para facilitar su control.

Así pues en este apartado no hemos tenido posibilidad de elegir otra alternativa y, el hecho de que todas las instancias de la aplicación se ejecuten en una misma máquina, nos obligó a controlar más estrictamente el consumo de memoria (ver requerimientos no funcionales).

9.5. Testing

Para la parte de testing del producto se ha tenido que recurrir a un Beta Tester externo ya que no podíamos garantizar un testing en profundidad con la carga de trabajo que teníamos los responsables del proyecto.

Para ello se introdujo una herramienta de seguimiento de bugs (bug-tracking) vía web llamada Mantis. El uso de esta herramienta provoco el diseño de un protocolo para cambiar de estado un bug una vez se había reportado.

El primer paso era reportar el bug y adjuntar una toma de pantalla (screenshot) o una descripción junto con un dump de la Base de Datos. Esto lo podía hacer cualquier persona asociada al proyecto: Desarrollador, Consultor o Beta Tester. Una vez reportado el bug se asignaba al responsable en cada caso y en cuanto el desarrollador comenzará a trabajar en su solución se marcaba como aceptado. Finalmente, el desarrollador lo marca como resuelto y adjunta una pequeña descripción de la solución. Entonces empieza el proceso de verificación.

El proceso de verificación se realizaba en dos pasos, en un primer paso el beta tester interno del departamento, verificaba el buen funcionamiento y lo marcaba como validado. En un segundo paso, Elisabeth, consultora responsable del proyecto realizaba la última validación y si todo funcionaba correctamente el bug pasaba a estar cerrado.

Este proceso fue clave para que pudiéramos dedicarnos enteramente a programar y ya, llegando al final del proyecto, pulir los errores que podíamos haber cometido.

10 Conclusiones

Llegado este punto, y antes de acabar, merece la pena echar un vistazo atrás para sacar las conclusiones de lo que ha sido el proyecto, ver lo que se ha hecho bien y lo que se podría haber hecho mejor, y extraer lo que me podría aportar en un futuro.

10.1. Resultados obtenidos

Los objetivos que se debían cumplir con este proyecto estaban muy fijados desde el principio, tenia que ser un sistema capaz de gestionar las necesidades de materia prima que este caso se trata de chatarra con distintos niveles de calidades. Para ello el sistema tenia que ser capaz de integrar y tratar los datos procedentes de cuatro departamentos diferentes de la empresa y en función de éstos realizar una estimación de las necesidades futuras que tendrá la empresa a medio plazo.

Todos estos objetivos se han cumplido y se ha podido implementar toda la lógica que se definió en su momento. Ahora los diferentes departamentos de Celsa en Santander disponen de una herramienta que les permite disponer de los datos actualizados del resto de departamentos (no como ocurría hasta ahora) con el consiguiente ahorro de tiempo que les supone.

Además de estos objetivos funcionales también había un objetivo adicional que era cumplir el plazo de tiempo que se había acordado. Por tanto, se tenía que realizar un proyecto complicado, extenso, que interactúa con otros módulos y con sistemas de información externos en un tiempo muy ajustado.

A pesar de tan ambiciosos objetivos se ha conseguido con mucho esfuerzo y muchas horas completarlos todos ellos, y sobretodo estoy contento por haber cumplido con la calidad del software. Aunque sea nuestro trabajo y cobremos tanto si el cliente lo utiliza como si no, es muy gratificante ver como el usuario esta contento, valora y usa una herramienta a la que le has dedicado tanto tiempo y esfuerzo.

10.2. Problemas hallados

Dentro del proceso de desarrollo del proyecto no todo ha ido saliendo según lo esperado, ya que en ciertos puntos han surgido ciertos imprevistos que han retrasado o paralizado temporalmente su avance.

Uno de estos cambios no previstos y quizá el más importante, fue la incorporación del módulo genérico de previsiones al sistema. Inicialmente se pensó en diseñar un módulo más sencillo de previsiones propio para este proyecto, pero una vez se comenzó a ver la potencia y posibilidades que ofrecía el nuevo módulo de previsiones se decidió incorporarlo a este proyecto. Es importante tener en cuenta que tanto el modulo de previsiones como este proyecto se fueron desarrollando a la vez, y fue este el motivo de que no se decidiera incorporarlo al sistema desde un principio.

Las consecuencias negativas que tuvo esta decisión fue que llegó un momento en que se hubo de paralizar temporalmente el desarrollo este proyecto hasta que estuvo finalizado el módulo de previsiones. Este hecho a parte del retraso en el desarrollo que supuso me hizo ver a nivel personal los problemas que supone dejar temporalmente un proyecto de este tipo, ya que a la hora de retomarlo se hace difícil volver a situarse en el punto donde se había dejado todo.

Otro contratiempo que tuvimos fue el retraso al recibir los diferentes ficheros con datos de SAP que necesitaba el sistema. Para recibir estos datos dependíamos del departamento de Celsa que gestiona todo el funcionamiento de SAP, y que debía realizar los cambios necesarios en su sistema para que diariamente se generasen los ficheros necesarios para generar las entradas de datos de nuestro sistema. A pesar de los esfuerzos que realizamos por pedirles que realizaran los cambios lo antes posible, hasta que no llegamos a la fase de testeo no pudimos disponer de estos ficheros. Aunque para realizar la implementación no era estrictamente disponer de estos ficheros ya que se crearon ficheros propios para realizar las pruebas, sí que nos habríamos ahorrado problemas de última hora debido a pequeños cambios que había en el formato interno de estos ficheros respecto el formato original acordado y que nos obligó a modificar nuestro sistema.

Quizá, cabe remarcar que los principales problemas hallados durante el desarrollo son comunes en muchos otros proyectos, especialmente en el mundo de la consultoría. Durante el desarrollo de un sistema software a la hora de definir los requisitos con los clientes, éstos requisitos nunca se cierran definitivamente porqué ni tan solo los clientes están seguros de lo que quieren o esperan recibir. Este hecho comporta cambios en diferentes etapas del proyecto que no siempre se abordan de la mejor forma pero que son necesarios para que una vez entregado el producto este acabe satisfaciendo las expectativas del cliente.

10.3. Estado de la implantación

En el momento de la redacción de esta memoria el sistema ya se encuentra implantado y funcionando. Los distintos departamentos involucrados en el desarrollo del proyecto ya lo utilizan como herramienta de trabajo diaria y con resultados muy satisfactorios.

La implantación empezó a finales de mayo de 2007 y duró 2 semanas y media, los principales problemas fueron la comunicación con los sistemas de información ajenos a nosotros (SAP y SGCS) y los ajustes que se tuvieron que hacer para poder guardar los datos de escenarios históricos.

Hay que destacar los hechos que han hecho esta implantación más "fácil". El primero es el hecho de que es un software que no ha sido impuesto por directivos sino que fue pedido directamente por varios de los usuarios finales (sobretodo los departamentos de compras y logística) esto hizo que la actitud por su parte fuera muy buena y se involucraran de lleno en los requerimientos funcionales y en el diseño de los algoritmos de cálculo, así que han tenido exactamente lo que pedían.

Otro elemento fundamental ha sido la gran labor realizada por Elisabeth (consultora especializada en logística) durante la especificación, diseño y seguimiento del desarrollo. Estuvo muy involucrada durante todo el proyecto transmitiéndonos las necesidades del cliente y ejerciendo de Beta-Tester.

10.4. Líneas de futuro

Este proyecto es susceptible de contar con numerosas ampliaciones, que se supone que irán surgiendo a medida que el grupo Celsa desee completar el proceso de gestión de su cadena de suministro e inicie nuevos proyectos, en los que se deba incluir la gestión de previsiones.

Un ejemplo de estas ampliaciones del sistema que ya se ha iniciado es la integración del reaprovisionamiento de ferroaleaciones y aditivos en el sistema. Como se ha comentado anteriormente, a la hora de realizar la fundición de la chatarra en la acería es necesario incorporar ciertos aditivos que mejoren la calidad del producto resultante.

Esta nueva fase del proyecto aprovechará los datos de la evolución de inventario de palanquilla usando las previsiones del programa de producción para realizar un plan de consumos previsto de ferroaleaciones y aditivos.

Otra modificación que ya se esta llevando a cabo por petición expresa del cliente, es incorporar al sistema la posibilidad de realizar ventas de chatarra. Debido a la gran cantidad de proveedores que tiene el grupo Celsa en muchas ocasiones le sale más a cuenta comprar grandes cantidades de chatarra, superiores incluso a las necesidades marcadas por el sistema. Más adelante si llegan a tener almacenada una cantidad de chatarra por encima de la capacidad de almacenaje que tiene la planta, vuelven a vender ésta dentro del mercado nacional.

10.5. Valoración personal

Este ha sido el primer proyecto que realizo íntegramente como profesional informático y además me permite finalizar mis estudios. Con él he podido comprobar las dificultades de un gran proyecto y aplicar una metodología de trabajo estructurada en su desarrollo dentro de un entorno real de trabajo.

Como es lógico pensar, en este proyecto ha sido imposible aplicar todos los conocimientos adquiridos a lo largo de estos años de estudio en la facultad debido a la gran diversidad de temáticas que he tratado en las asignaturas que he cursado. Sin embargo, sí que he podido sobretodo usar y ayudarme de lo aprendido en aquellas asignaturas relacionadas con los perfiles, de "ingeniería del software y sistemas de información" y "gestión y recuperación de la información". Estos perfiles son los más adecuados para dirigir proyectos de desarrollo de software destinado a la gestión de organizaciones ya que proporcionan los conocimientos necesarios para analizar los requisitos y de especificar, diseñar e implementar los sistemas software con alta calidad y bajo coste. En resumen, lo necesario para la realización de este proyecto.

Sinceramente, he quedado bastante satisfecho con el trabajo realizado. Por una parte, me ha permitido introducirme en la plataforma .NET, muy extendida en la actualidad y que complementan los conocimientos de otras plataformas con las que he trabajado a lo largo de estos años en la facultad.

Por otra parte, con la experiencia adquirida durante este proyecto me he podido introducir dentro del mundo de la logística en el cual no había trabajado hasta el momento, y que seguro me serán de gran utilidad de ahora en adelante en la gestión y desarrollo de proyectos de gran envergadura.

10.6. Agradecimientos

Con este proyecto cierro una etapa de mi vida, aunque no por ello mi vida de estudiante. He conocido muchísimas personas durante este tiempo y todas me han influido de uno modo u otro, así que una pequeña parte de lo que yo soy se lo debo a esas personas. Me es imposible aunar mis agradecimientos a todas ellas en esta memoria, así que me limitaré a aquellos que han estado más cerca de mí a lo largo del periodo en el que he desarrollado este proyecto.

Primero agradecer a mi familia sin la cual todo esto no seria posible y mis amigos que han estado a mi lado y que han dado su apoyo en la medida de lo posible.

Gracias a Miquel Barceló por haberme dirigido el proyecto y a Carlos por haberme ayudado a comenzar y guiarme a lo largo de éste. Gracias a Elisabeth por escuchar mis dudas y ayudarme a decidir ofreciendo siempre su punto de vista en los problemas que le planteaba. A Vivi que a pesar de haber estados distanciados todo este año no ha cesado de darme ánimos día a día desde Grecia y cuyo apoyo he tenido siempre que lo he necesitado. A Ramiro, Pere y Ferran con los cuales he podido compartir estos últimos años y cuya amistad es el mejor regalo que me llevo de la facultad, espero continuar disfrutando de ella muchos años más. A Joan y Laura, la mejor pareja de toda la facultad y que siempre han tenido tiempo para compartir conmigo, cuyo valor no tiene precio. Y como no dar gracias a toda la sección de sistemas de ALG cuya compañía estos meses ha hecho que la carga que este proyecto pareciera mucho menor de lo que habría sido sin ellos.

Gracias también a todos los que no nombro pero que habéis puesto de vuestra parte para que esto haya llegado a buen puerto. Esto va por todos vosotros.