Documento de PLANIFICACIÓN

Sistema de Administración y Gestión de RESTaurantes

v0.9

ÍNDICE DE CONTENIDO

Apartado de control de versiones....4
1. Introducción....5
2. Declaración de alcance....6
Objetivos....6
Especificación a alto nivel....6
Viabilidad....6
3. Recursos disponibles....7
Personal y Organización....7
Hardware y Software....8
Infraestructura de manutención de código y documentos....10
4. Estimación de costos....11
Datos históricos utilizados....11
Datos para la estimación de costos....11
5. Planificación temporal....13
Descripción de tareas....13
Modelado de requisitos....13
Análisis....13
Diseño....14
Implementación....14
Prueba....14
Diagrama de Gantt....14
6. Análisis de los riesgos....16
7. Formato de la documentación....17
1. Estructura de carpetas....17
2. Documentación....17
3. Estructura en los documentos de código....19
Reglas semánticas....19
Denominación de variables....19
Denominación de funciones....20
Denominación de clases....20
Documentación y comentarios....21
Llaves e identación....21

Apéndice 0.1.....22
Apéndice 0.2.....23
Apéndice 0.3.....24
Apéndice 0.9.....25

Apartado de control de versiones

Todas las versiones están especificadas a fondo en el apartado de "Apéndices", al final de este documento, cada apéndice se corresponde en nombre con su número de versión. Por ejemplo, el "Apéndice 0.1" se corresponde con la versión v0.1. Para ver los cambios realizados sobre cada versión, hay que ir deshaciendo los cambios desde el final.

Versión	Fecha	Descripción
V0.1	08/03/10	Presentación inicial del documento.
V0.2	09/03/10	Cambio de plataforma de comunicación al Awacate.
V0.3	10/03/10	Cambios en el diragrama de Gantt y añadido un nuevo miembro al grupo.
V0.9	13/03/10	Reestructuración completa del documento, añadidos detalles en todos los campos y nuevas secciones.

1. Introducción

Este documento describe la organización y planificación del grupo TouchTeam para el proyecto de Ingeniería del Software 3, al cual hemos denominado SAGRES (Sistema de Administración y Gestión de REStaurantes).

Este proyecto se realizará durante el curso 2009/2010 para aprender el desarrollo del ciclo de vida del software. El proyecto se desarrolla durante tres iteraciones.

Se desea desarrollar una aplicación para el restaurante del hotel “Duerme Mucho”, para ello el sistema gestionara los pedidos de los clientes, a partir de ella los cocineros gestionaran las cartas del menú, los pedidos de los clientes y los pedidos a los proveedores con el sistema, también el metre gestionara los pedidos de las bebidas y la facturación.

2. Declaración de alcance

Objetivos

En este primer paso, vamos a tratar de completar toda la tarea de gestión de los ingredientes y de los productos ofertados en la carta. A continuación haremos una especificación a alto nivel de los elementos que esperamos llevar a cabo.

Especificación a alto nivel

El cocinero elaborará la carta una vez cada seis meses, si lo desea, aunque podrá realizar modificaciones puntuales siempre y cuando el restaurante esté cerrado. Así se evitará el riesgo de que un cliente pida un plato que esté siendo modificado.

La carta está dividida en distintas secciones:

- Entradas
- Carnes
- Pescados
- Bebidas
- Postres

Estas secciones serán fijas y no se podrán eliminar de la carta.

A la hora de añadir un nuevo plato a la carta, el cocinero deberá especificar cada uno de los ingredientes que componen el plato, el precio, el gasto de elaboración (asociado a los ingredientes usados) y si es posible adjuntar una foto del mismo. Todos estos detalles del plato podrán ser modificados posteriormente por el cocinero, e incluso se podrán eliminar de la carta si fuese necesario.

Respecto a los ingredientes de cada plato, el cocinero se encargará de ir marcando en el sistema los ingredientes que se van utilizando al cocinar un plato. El sistema también le deberá permitir notificar si un ingrediente se encuentra en mal estado (estos casos se marcarán como pérdidas).

El cocinero y el camarero conjuntamente fijarán el mínimo y el máximo de cada producto. A partir de los productos consumidos y del mínimo que debe haber de existencias el sistema deberá elaborar una lista de necesidades con las cantidades requeridas de cada uno de los productos que se deban comprar. El sistema deberá permitir imprimir dicha lista.

Cuando se recibe el pedido del proveedor el cocinero deberá notificarlo en el sistema, actualizándose la base de datos de ingredientes de este.

Viabilidad

Para la conclusión de las tareas anteriormente descritas disponemos de, aproximadamente, 30 días. A priori conocemos que tendremos las etapas de Modelado de requisitos, Análisis, Diseño e Implementación. En principio, suponiendo que llevara una media de una semana por cada etapa, dada la embergadura de las funcionalidades que esperamos completar, vemos que el proyecto por cuestiones de tiempo es viable, teniendo en cuenta los recursos de los que dispondremos.

3. Recursos disponibles

Personal y Organización

El grupo consta de diez miembros, los cuales listamos a continuación:

1. Dionisio Ruiz, Jose David (j2dr@correo.ugr.es)
2. García Sánchez, Ángel (pilli@correo.ugr.es)
3. Guerrero Martinez, Daniel (lein@correo.ugr.es)
4. Guirado Navarro, Samuel (sgn88@correo.ugr.es)
5. Moreno Muñoz, Carlos (camomu@correo.ugr.es)
6. Muñoz Soria, Gaspar (gmunozs@correo.ugr.es)
7. Pérez Lopera, Adrián Victor (elpelu@correo.ugr.es)
8. Rodríguez Lumley, Sergio (lumley@correo.ugr.es)
9. Sabeg, Nabil (nsabeg@correo.ugr.es)
10. Salas Morales, Carlos (csalasm@correo.ugr.es)

Hemos dividido el grupo en tres pequeños subgrupos, (A, B, C). Estos son:

Equipo C:

- Pérez Lopera, Adrián Victor
- Rodríguez Lumley, Sergio
- Sabeg, Nabil

Equipo A:

- Guirado Navarro, Samuel
- Moreno Muñoz, Carlos
- Muñoz Soria, Gaspar

Equipo B:

- Dionisio Ruiz, Jose David
- García Sánchez, Ángel
- Guerrero Martinez, Daniel
- Salas Morales, Carlos

La división de tareas para las tres iteraciones es la siguiente:

- **Iteración 1**: Planificacion (C), Análisis y Diseño (A), Implementacion (B)
- **Iteración 2**: Planificacion (B), Análisis y Diseño (C), Implementacion (A)
- **Iteración 3**: Planificacion (A), Análisis y Diseño (B), Implementacion (C)

Esta división ha sido para asegurar que en la última etapa (Que preveemos, será la más difícil, por las fechas en las que se encuentra) tengamos cuatro miembros en el grupo encargado del Análisis y Diseño, haciendo que sea más asequible terminar el proyecto en los tiempo planeados.

Hardware y Software

Todos los miembros del equipo disponen de un PC portátil, con lo que las reuniones entre los subgrupos se podrán realizar en cualquier parte y cualquier fecha.

El grupo dispone de un servidor remoto privado para realizar las pruebas necesarias de conexión a servidores y para construir el programa deseado.

Como grupo de desarrollo de bajo costo, nos interesa utilizar software libre, con licencias abiertas y con el respectivo ahorro de recursos. Esto, sin embargo, no es posible en todos los casos.

El software del que dispondremos es: GIMP 2.6, OpenOffice3.1.0, Enterprise Architect 7.5, OpenProject, NetBeans6.8, Dia, Gmail, Awacate, Google Code, TortoiseSVN, Axure.

Nombre	Versión	Descripción	Campo asociado	Licencia
GIMP	2.6	Programa de edición de imágenes.	Gráficos.	GNU
OpenOffice	3.1.0/3.2.0	Edición de texto.	Documentación.	GNU
Enterprise Architect	7.5	Editor de diagramas UML.	Modelado de requisitos, análisis y diseño.	Propietario
OpenProject	1.4-2	Editor de diagramas de Gantt.	Planificación.	GNU
NetBeans	6.8	IDE de Java y PHP.	Implementación.	CDDL
Dia	0.97	Editor de diagramas ligero.	Planificación.	GNU
Gmail	2.0	Gestor de correo.	Comunicación.	Propietario
Awacate	Beta	Gestor y comunicador de grupo.	Comunicación.	LGPL
Google Code	Beta	Servidor SVN.	Servidor.	LGPL
TortoiseSVN	1.6.7	Cliente SVN	Control de ficheros	LGPL
Axure	5.6	Herramienta de prototipado.	Diseño.	Propietario

Tabla 1. Tabla de recursos de software.

A continuación vamos a mostrar los campos en los que generalmente asociaremos el software utilizado y la razón de su selección. Los campos que a continuación se mostrarán, no son los únicos que los utilizarán, sin embargo si será más común que aparezcan ahi:

1. ***Planificación***
    a) ***Diagramas de Gantt***: Utilizaremos OpenProject, al ser de código abierto, nos ahorra la tarea del pago de licencias, disminuyendo el costo asociado al proyecto. Es también multiplataforma, con lo que no hay restricción a la hora de elegir un sistema operativo. Además, es plenamente compatible con Microsoft Project 2000, herramienta de la que dispusimos inicialmente, con lo que la portación de proyectos es instantánea.
    b) ***Comunicación con el grupo***: toda la comunicación se gestiona a través del sistema Awacate de la ugr (http://awacate.ugr.es). Con ella, todos los miembros del grupo nos enviamos mensajes, disponemos de un foro de discusión, un apartado para subir documentos en común y un sistema de asignación de tareas, a través del cual se puede llevar un seguimiento continuado de tareas realizadas (El porcentaje de acabado y las horas dedicadas a ellas). En el apartado de documentos se seguirá un estricto formato de árbol de carpetas, lo cual detallaremos en el apartado 7 de este mismo documento ("Formato de la Documentación").
    c) ***Documentación escrita***: esto será realizado con el programa OpenOffice3. Elegimos este editor de textos puesto que es libre, de código abierto y multiplataforma. Además es de la calidad suficiente para la generación y gestión de toda la documentación necesaria para el proyecto. Otra razón de la elección de este formato es que nuestro sistema de control de versiones queda muy simplificado gracias a que SVN (nuestro sistema de repositorios) puede leer este formato (.odt) y, por tanto, notificar las diferencias entre las versiones que maneja. Con este programa también es posible portar el documento a formato PDF. Para la lectura de este formato, el equipo dispone de diversas herramientas como son FoxitReader, Adobe Reader o Evince.
2. ***Análisis y Diseño***
    a) ***Prototipado de diagramas***: utilizaremos la herramienta Dia, por su facilidad a la hora de representar diagramas UML. Además es libre, de código abierto (Licencia GNU) y multiplataforma, lo que significa que todos podremos compartir los diagramas y los bocetos de forma rápida y sencilla.
    b) ***Diagramas UML***: para la realización de los diagramas UML finales, utilizaremos la herramienta EnterpriseArchitect 7.5, ya que cumple con todos los requisitos y la calidad que esperamos de un software de este tipo. Además tiene la posibilidad de generar código a partir de sus diagramas, lo que puede reducir el trabajo a los miembros del grupo de implementación.
    c) ***Diseño de la interfaz de usuario***: para la realización de bocetos y acabado de diseño, utilizaremos GIMP 2.6, editor de dibujo pixelado libre, de código abierto (GNU) y multiplataforma. Presenta plena compatibilidad para todos los formatos estándar de imágenes y con los proyectos de otros editores (como Photoshop).
3. ***Implementación***
    a) ***Edición de prototipos***: utilizaremos en principio el programa Axure 5.6, propietario, con licencia de estudiante para la UGR. Esta herramienta nos permite realizar en a penas unas horas un prototipo visual con una apariencia visual muy similar al resultado final y con una aparente funcionalidad (que no será real). Esto nos permitirá avanzar rápidamente por el diseño de la interfaz de usuario.

b) ***Generación de código***: nuestra elección, tanto para Java como PHP como bien cualquier otro lenguaje soportado, es NetBeans6.8. Este IDE libre, de código abierto y multiplataforma es líder en su sector por la calidad de su entorno y su estricta regla de seguir con los estándares abiertos tanto en la generación de código como en la generación de documentación y bibliotecas utilizadas. Otra posible elección de alto nivel podría ser Eclipse Galileo 3.5.1, sin embargo este IDE no es tan conocido por los integrantes del grupo, con lo que habría que tener en cuenta un tiempo de aprendizaje y acomodación al mismo.

c) ***Compartición de código***: utilizamos una herramienta cliente SVN, TortoiseSVN. Ésta es libre, de código abierto y multiplataforma, con lo que solo será necesario aprender a utilizar una herramienta. Su sencilla interfaz y sus múltiples funcionalidades convierten esta herramienta como clara elección a la hora de compartir código desde Windows. Además es plenamente integrable en NetBeans6.8. Para linux tenemos también una versión de TortoiseSVN, o bien de otros como KDEsvn.

Infraestructura de manutención de código y documentos

Para compartir y mantener el código seguro, utilizamos un sistema de repositorio basado en SVN. El servidor SVN se encuentra en Google Code, una solución libre y de código abierto, puesta para los desarrolladores de proyectos de código abierto. Esto significa que no tenemos ningún costo asociado a este servicio, sin embargo disponemos de un servicio de calidad profesional, puesto que Google pone a nuestra disposición suficientes servidores como para asegurar una efectiva redundancia, con la velocidad de actualización y conexión que sólo una empresa del tamaño de Google pueden ofrecer. El acceso a dicho servidor es libre y abierto, basta con conectar con el servidor:

http://touchteam.googlecode.com/svn/trunk

No son necesarios ni usuario ni contraseña. Esto permite que cualquier persona pueda llevar cuenta de cómo avanza el proyecto, sin embargo para tener privilegios para realizar cambios y subirlos al servidor, si que es necesario pertenecer al grupo.

A pesar de este control, es posible que algún día (seguramente en el que más necesario sea) el servidor se encuentre con dificultades técnicas. Como regla general, el equipo mantiene una copia en disco, con lo que se puede compartir rápidamente a través del sistema de Awacate o bien por e-mail convencional. Los documentos tienen una doble redundancia, ya que cada versión final se sube al sistema de documentos de Awacate, conservando todos los antiguos.

4. Estimación de costos

Datos históricos utilizados

Como referencia en la realización de diagramas y especificaciones en las etapas de Modelado de Requisitos, Análisis y Diseño, tenemos todos los miembros del equipo la experiencia de la asignatura Ingeniería del Software 2 (2009-2010). Con lo que podemos estimar un tiempo necesario para la realización de cada una de las tareas individuales de cada etapa, así como el tiempo de la etapa en sí.

Como experiencia en implementación, algunos miembros del equipo han desarrollado asignaturas de cometido final similar al sistema pedido por el cliente. A pesar de ser experiencias basadas en proyectos de menor tamaño, si nos sirven como estimación del tiempo que emplearemos en el desarrollo y utilización de herramientas.

Datos para la estimación de costos

Para valorar el esfuerzo empleado sobre los subsistemas seleccionados, hemos realizado un boceto de un diagrama de casos de uso, con lo que se puede ver la funcionalidad esperada del sistema al completo y la proporción de lo que debemos realizar en esta iteración.

Como podemos comprobar en la figura 1, las funcionalidades que completaremos en esta iteración (Marcadas en Azul celeste, arriba a la derecha, y en Violeta (Arriba)) son el subsistema de gestión de carta y de ingredientes. Todo esto se ha elaborado para hacer una repartición óptima de tareas y una estimación de costos más acertada, sin embargo todo esto es susceptible a cambios.

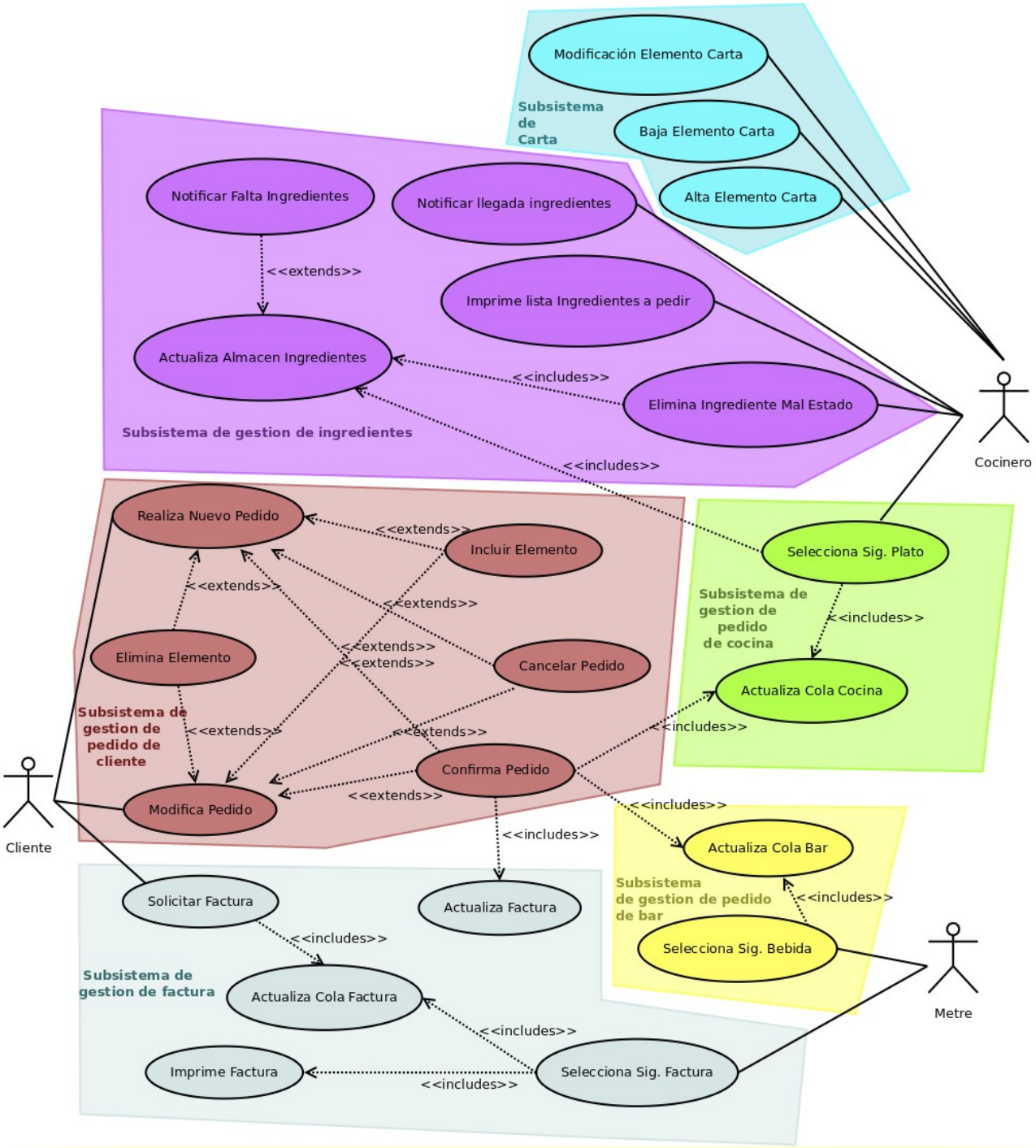


Figura 1. Boceto de diagrama de Casos de Uso de sistema.

5. Planificación temporal

Descripción de tareas

Para cada una de las tareas se ha asociado el tiempo estimado de dedicación en horas por los miembros del grupo.

Modelado de requisitos

Tarea 1: Realizar el modelado de requisitos funcionales, mediante el diagrama de casos de uso del sistema. (2 horas)

Tarea 2: Realizar la especificación detallada de los casos de uso, intentando identificar en cada uno de ellos requisitos no funcionales del problema. (3 horas)

Tarea 3: Realizar descomposición del sistema en subsistemas funcionales, mediante el diagrama de paquetes funcionales. (1 hora)

Tarea 4: Llevar a cabo la identificación de requisitos no funcionales del problema. (1 hora)

Tarea 5: Realizar los diagramas de secuencia del sistema. Se realiza un diagrama por cada caso de uso. (5 horas)

Tarea 6: Realizar una lista con las operaciones del sistema. (1 hora)

Tarea 7: Revisión del modelado de requisitos. (2 horas)

Tarea 8: Realizar el documento de modelado de requisitos. (1 hora)

Análisis

Tarea 9: Identificar en el problema las posibles clases, las relaciones que hay entre ellas y sus atributos. (2 horas)

Tarea 10: Realizar el diagrama de clases del análisis. (1 hora)

Tarea 11: Realizar los contratos de todas las operaciones, obtenidas en los diagramas de secuencia de sistema. (3 horas)

Tarea 12: Definir un diagrama de colaboración por cada contrato, teniendo en cuenta el diagrama de clases de análisis definido. (5 horas)

Tarea 13: Revisión del análisis. (2 horas)

Tarea 14: Realizar el documento de análisis. (1 hora)

Diseño

Tarea 15: Decidir el estilo arquitectónico que se adecúa mejor a la solución. (1 hora)

Tarea 16: Definir el diagrama de clases del diseño inicial, obteniéndolo a partir del diagrama de clases del análisis. (1 hora)

Tarea 17: Realizar diagrama de paquetes estructurales. Se descompone el diagrama de clases en subsistemas estructurales, utilizando el diagrama de paquetes obtenido en el modelado de requisitos. (1 hora)

Tarea 18: Eliminar dependencias cíclicas del diagrama de paquetes estructurales. (2 horas)

Tarea 19: Para cada paquete, revisar su diagrama de clases interno, identificando los servicios requeridos y los ofrecidos (interfaces). (3 horas)

Tarea 20: Una vez definidas las interfaces de los paquetes, transformar el diagrama de paquetes en un diagrama de componentes. (1 hora)

Tarea 21: Elaborar los diagramas de colaboración de diseño, teniendo en cuenta los diagrama de clases de diseño definidos. (4 horas)

Tarea 22: Diagrama entidad-relación de la base de datos. (1 hora)

Tarea 23: Diagrama relacional de la base de datos. (1 hora)

Tarea 24: Realizar el diagrama de despliegue, identificando la ubicación física de cada subsistema estructural. (1 hora)

Tarea 25: Diseño de la interfaz de usuario. (3 horas)

Tarea 26: Revisión del diseño. (2 horas)

Tarea 27: Realizar documento de diseño. (1 hora)

Implementación

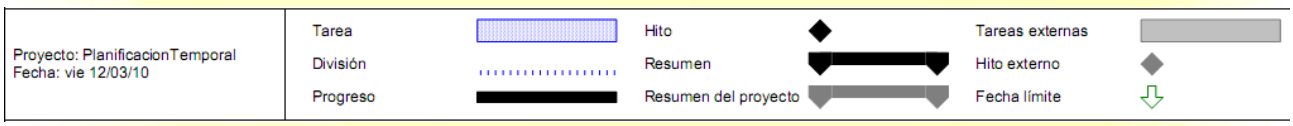
Tarea 28: Implantación de la base de datos. (2 horas)

Tarea 29: Implementación de todas las clases de los subsistemas funcionales. (15 horas)

Prueba

Tarea 30: Realizar pruebas de corrección de las funcionalidades del sistema completo desarrollado. (5 horas)

Diagrama de Gantt



Id	Nombre de tarea	Duración	Comienzo
1	**Modelado de Requisitos**	**7 días?**	**lun 08/03/10**
2	Diagrama de Casos de Uso	1 día?	lun 08/03/10
3	Especificación detallada de Casos de Uso	2 días?	mar 09/03/10
4	Diagrama de Paquetes Funcionales	1 día?	jue 11/03/10
5	Requisitos No Funcionales	1 día?	jue 11/03/10
6	Diagramas de Secuencia del Sistema	3 días?	jue 11/03/10
7	Lista operaciones del sistema	1 día?	dom 14/03/10
8	Revisión	1 día?	dom 14/03/10
9	Documento de Modelado Requisitos	1 día?	dom 14/03/10
10	Entrega de Documento de Modelado Requisitos	0 días	lun 15/03/10
11	**Análisis**	**5 días?**	**lun 15/03/10**
12	Identificar Clases, Atributos y Relaciones	1 día?	lun 15/03/10
13	Diagrama de Clases	1 día?	lun 15/03/10
14	Contratos	3 días?	mar 16/03/10
15	Diagramas de Colaboración de Análisis	3 días?	mar 16/03/10
16	Revisión	1 día?	vie 19/03/10
17	Documento de Análisis	1 día?	vie 19/03/10
18	Entrega de Documento de Análisis	0 días	sáb 20/03/10
19	**Diseño**	**7 días?**	**dom 21/03/10**
20	Decidir Estilo Arquitectónico	1 día?	dom 21/03/10
21	Diagrama de Clases de Diseño	1 día?	dom 21/03/10
22	Diagrama de Paquetes Estructurales	1 día?	lun 22/03/10
23	Eliminar Ciclos	1 día?	lun 22/03/10
24	Indentificar Interfaces	1 día?	mar 23/03/10
25	Diagrama de Componentes	1 día?	mar 23/03/10
26	Diagramas de Colaboración de Diseño	3 días	mar 23/03/10
27	Diagrama Entidad-Relación	1 día?	vie 26/03/10
28	Diagrama Relacional	1 día?	vie 26/03/10
29	Diagrama de Despliegue	1 día?	vie 26/03/10
30	Diseño de la Interfaz de Usuario	1 día?	sáb 27/03/10
31	Revisión	1 día?	sáb 27/03/10
32	Documento de Diseño	1 día?	sáb 27/03/10
33	Entrega de Documento de Diseño	0 días	dom 28/03/10
34	**Implementación**	**6 días?**	**dom 28/03/10**
35	Inplantación de la Base de Datos	1 día?	dom 28/03/10
36	Implementación de Clases	6 días?	dom 28/03/10
37	**Prueba**	**2 días?**	**sáb 03/04/10**
38	Diseño y Ejecución de Pruebas	2 días?	sáb 03/04/10
39	Entrega de Subsistemas Finalizados	0 días	lun 05/04/10

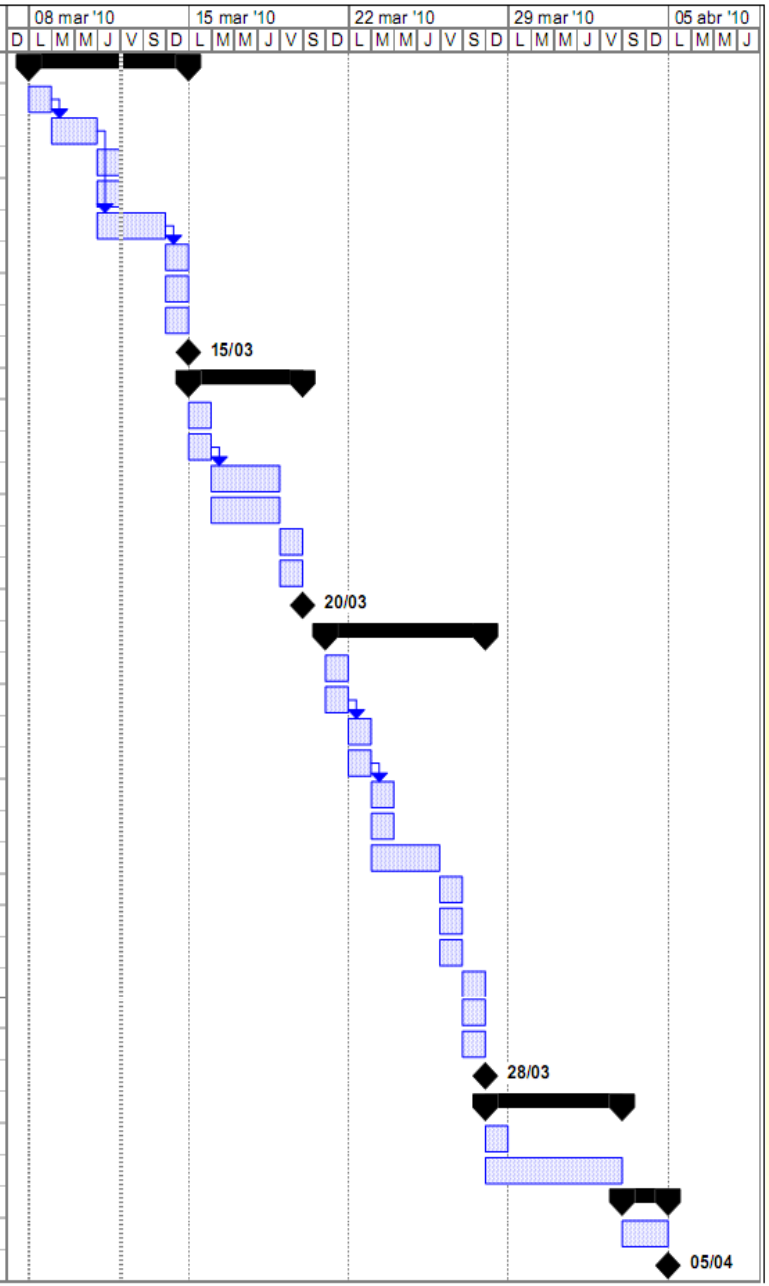


Figura 2. Diagrama de Gantt.

6. Análisis de los riesgos

Un posible riesgo es que un miembro del grupo falle. Entiéndase por fallo que abandone el grupo, que no termine las tareas a tiempo o bien que quede incomunicado con el resto del equipo. En este caso, procederemos a repartir las tareas sobre los miembros restantes del subgrupo en el que se ha producido la falta. Si las tareas faltantes fuesen demasiado voluminosas como para ser cumplidas dentro del plazo estimado, se pedirá ayuda a algún miembro de otro subgrupo, que podría en cualquier caso, colaborar de la forma necesitada.

No consideraremos los riesgos por fallo del servidor SVN, el grupo de contacto o de conexión a Internet, puesto que todos los miembros disponemos de una red suficientemente fiable en la Universidad de Granada y los servidores de datos son gestionados por una empresa que oferta soluciones software profesionales sin costo, con suficiente redundancia de datos como para mantener todos los datos seguros.

7. Formato de la documentación

1. Estructura de carpetas

Puesto que utilizamos para compartir carpetas, tanto el repositorio SVN como el sistema de documentos del Awacate, llevamos a cabo una estricta estructura de carpetas. Esta queda como sigue:

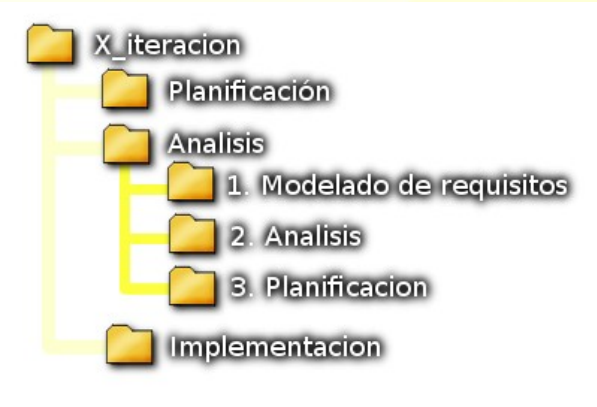


Figura 3. Estructura de carpetas.

Con X el número de la iteración. Esta estructuración permite clonar los documentos en su interior y presentar una plena homogeneidad en el proyecto, ya que un documento dentro de "1_iteración → Planificacion" se da por hecho que es un documento única y exclusivamente perteneciente a la planificación de la primera iteración. Además da cierto grado de libertad a cada subgrupo puesto que cada grupo tiene únicamente control sobre su carpeta, haciendo que los cambios en una no estorben al resto del equipo.

2. Documentación

Toda la documentación es realizada en OpenOffice3, luego definiremos cómo se estructura con dicho programa de edición de textos. Para asegurar una documentación que siga acorde con los puntos que detallaremos a continuación, se han creado plantillas de documentación (formato .ott), con lo que se facilita el cumplir con ellos. Todos los miembros del grupo cuentan con una copia online o local de ellas. Las plantillas creadas son:

- **Plantilla de Documento**: para crear un documento sobre una etapa, es la documentación que se entregará en cada hito. Tiene anexado un fichero de cabeceras para el título disponible en cada documento, acorde con el diseño seguido en dicho formato. En detalle, punto por punto y en orden, se define:
  - *Título al completo*: prediseñado, con logo incluido, será necesario cambiar la versión marcada por la actual, al igual que la iteración.
  - *Índice de contenido*: es autogenerado, sólo es necesario actualizarlo tras cada cambio. El formato de cada nivel es el mismo que el correspondiente con las cabeceras. Detallaremos éstas a continuación.
  - *Apartado de control de versiones*: presenta una introducción de explicación sobre dicho apartado, además de una tabla con un diseño acorde al del documento, la tabla tiene por

formato una cabecera (la primera fila) Times New Roman, tamaño 12 en negrita, centrado. El resto de filas serán en Times New Roman, tamaño 12 sin negrita. Las primeras dos columnas están alineadas a la derecha, mientras que la tercera a la izquierda. Esto resulta en una mayor comodidad a la hora de leer, ya que la versión y la fecha se encuentran más cercanas a su descripción, ayudando a los ojos del lector a tener que desplazarse menos sobre el texto para encontrar su concordancia. La letra elegida es por su tamaño estándar, puesto que en el mundo empresarial, la gente está acostumbrada a leer en Times New Roman, es un formato fácilmente reconocible que no requiere a penas esfuerzo de interpretación.

    - *Cabeceras*: tienen todas en común que utilizan el estilo de letra Segoe Print, esta tipografía es libre y se encuentra disponible en nuestro repositorio y en el apartado de documentos del Awacate, para asegurar que todos los miembros puedan disponer de ella. La elección de esta tipografía es para romper con la homogeneidad, lo que destaca más a los ojos del lector, diferenciando fácilmente que este texto es de mayor importancia. Cada nivel de cabecera reduce su grado de oscurez (por ejemplo, el nivel 1 es más oscuro que el 2), para ayudar a distinguir qué apartado es superior a cual.
        - Cabecera de nivel 1: La letra es grande 16.1, con un color amarillo oscuro en consonancia con el tema del documento, la letra es de capitalización automática, en negrita y con subrayado completo. Esto simboliza un corte sobre el tema del documento, significa que a partir de ahora cambiamos completamente del tema de lectura. Ayuda a diferenciar los apartados estructurales del documento.
        - Cabecera de nivel 2: La letra es de tamaño 14, en negrita. El color es amarillo no muy oscuro, con sombreado (para ayudar a reconocer los contornos y no confundir el texto con el fondo). Una cabecera de segundo nivel indica que se va a tratar un punto referente al tema tratado en la cabecera de primer nivel.
        - Cabecera de nivel 3: Letra de tamaño 12, en negrita. El color es ligeramente más claro que uno de segundo nivel, también con sombreado por la misma razón que la anterior. Una cabecera de tercer nivel sencillamente indica que se entra en detalle sobre un apartado tratado en la cabecera de segundo nivel.
    - *Apartado de apéndices*: Al final de cada documento, debe haber un apartado de apéndices. Estos tienen el nombre de la versión a la que hacen referencia (Por ejemplo, el “Apéndice 0.1” hace referencia a los cambios realizados en la versión v0.1).
- **Plantilla de Especificación de casos de uso**: es una tabla que especifica todos los detalles a enumerar, el formato de las enumeraciones en los cursos de los eventos, tanto normales como alternativos. Se ha utilizado un bordeado amarillo suave dispuesto de tal forma que de la sensación de profundidad. Los colores son acorde al tema utilizado en el documento. La cabecera de la tabla es la primera columna, la letra elegida es Segoe Prints, tamaño 11, en negrita. El cuerpo de la tabla tiene letra Times New Roman tamaño 12, por las razones anteriormente detalladas. Esta diferenciación permite al lector reconocer rápidamente cual es la cabecera y cual el contenido.
- **Plantilla de Contratos**: es una tabla idéntica a la anterior, con la cabecera de tabla necesaria para detallar un contrato correcto.

No se permite insertar ninguna identación (o sangría) que no sea implícita del texto, tal como las enumeraciones. Esto hará que se aproveche en todo momento el ancho del documento al completo, evitando tener que cambiar de línea de lectura demasiado a menudo.

Los documentos siguen todos el nombre de lo que son, terminando con la versión a la que corresponden. Cuando un documento se encuentra finalizado, se realiza una copia en PDF y se comparte con el resto de compañeros, estos documentos **NUNCA** se borran, son acumulativos. Se le añade al nombre " v[version]". El formato especificado queda como sigue:

Documento de Planificacion v0.2.pdf.

3. Estructura en los documentos de código

Reglas semánticas

Las siguientes reglas se aplican para el código de Java, de PHP y SQL.

Todos los nombres de variables, funciones, comentarios, clases, etc. están en *español. Los nombres seguirán la estructura "*Camel Case*" donde todas las letras son minúsculas excepto la primera letra de cada palabra, la primera letra de la primera palabra es también una minúscula. Tienen que tener un nombre de valor informativo, acorde con su función o valor. Por ejemplo:

obtenerEdad(), edad, edadPadre, etc.

Por razones de compatibilidad los nombres de variables y métodos no podrán tener caracteres especiales españoles, como 'ñ' o acentos.

**Hay dos tipos de funciones que, por facilidad de escritura no estarán en español, estas son las funciones de asignación y de devolución de valor.*

Denominación de variables

Vamos a utilizar generalmente sustantivos para nombrar a las variables, ya que suelen significar instancias. El nombre de las variables globales de las clases empezarán con "g" (por "variable global"), por compatibilidad con la internacionalización de código (global es igual en inglés, idioma extensamente utilizado en el código). Por ejemplo:

gEdad, gEdadPadre

Las variables utilizadas para las iteraciones en los bucles no tienen porqué necesariamente tener un nombre con significado español, por su corta vida y su continuada referencia en bucles, lo más común será utilizar una única letra (tal como 'i', 'j', 'k', etc).

Las constantes no seguirán las regla de "*Camel Case*" ya que estarán totalmente mayúsculas. Para solventar el problema de tener nombres compuestos por distintas palabras, la separación la haremos mediante un guión bajo:

EDAD, EDAD_PADRE

Denominación de funciones

Para la denominación de funciones tendremos en cuenta cuatro partes, la denominación de funciones genéricas, funciones booleanas, funciones de obtención de valor y funciones de asignación de valor:

1. Funciones genéricas: Los nombres de funciones están , si es aplicable, en forma verbal imperativa, aunque podrán contener un sustantivo. Por ejemplo:

```
pinta(...), conecta(...), pintaVentana(...)
```

2. Funciones de retorno booleano (o condicionales): estas funciones se utilizan para devolver estados (verdadero o falso). El nombre de este tipo de funciones será "está" seguido de un adjetivo o un participio.

```
estaLleno(...), estaConectado(...)
```

3. Funciones de asignación de valor: asignan un valor a algún dato (generalmente perteneciente a una clase), como excepción anteriormente mencionada, estas comenzarán por "set" seguido del nombre (de valor informativo) de variable. Por ejemplo, si vamos a asignar la altura a una ventana:

```
ventana.setAltura(x);
```

4. Funciones de obtención de valor: estas funciones devuelven algún valor, generalmente de alguna instancia de una clase. Estas funciones **nunca tendrán parámetros**. Al igual que la anterior, esta comienza en inglés con la palabra "get" (obtener) seguido del atributo que se espera obtener. Por ejemplo, obtener la altura de una ventana:

```
ventana.getAltura(x);
```

Denominación de clases

La definición de las clases serán sustantivos en singular, con la primera letra en mayúscula, para distinguirlos de la denominación de variables. Algunos ejemplos:

```
Ventana, Fecha, Persona
```

Documentación y comentarios

Cualquier comentario dentro de línea será aceptado siempre y cuando explique la funcionalidad con claridad, es decir, no se aceptarán comentarios del estilo “Aquí”, sino comentarios que aclaren la función, como por ejemplo:

```
Integer numero=0;
while(numero<5){
        imprimir objeto.getNombre(numero++);      // Aumentamos el valor de numero tras obtener
                                                  // el nombre del numero deseado
}
```

Para documentar el código, se utiliza la estructura de JavaDoc, esta estructura queda como sigue:

Antes de cada función:

```
/**
* Una descripción muy corta de lo que hace la función.
* @param <parámetro>  - descripción del parámetro <parámetro> (uno por cada parámetro)
* @return             - si la función no es de tipo “void“ se pondrá una descripción del valor de retorno
* @throw              - si esta función puede lanzar una excepción se pondrá una descripción ella
*/
```

Antes de cada clase:

```
/**
* Una descripción muy corta sobre el cometido de la clase
* @author             - el nombre de persona que ha implementado esta clase
*/
```

Llaves e identación

La llave de la izquierda '{' estará en la línea de declaración de clase/función/bucle etc. Después de cada '{' las siguientes líneas están identadas con una tabulación. Puesto que todos los miembros están instados a utilizar el IDE NetBeans6.8, todos tendremos un código igual identado, con formateado automático por parte del IDE. Después de cada '}' la identación tendrá una tabulación menos.

Despues de “{“ o antes de “}“ no habrá ninguna línea en blanco, tan sólo código o un comentario explicativo. Por ejemplo:

```
while(...) {
        if (...) {
                // código
                // código
                // código
        }
}
```

Apéndice 0.1

Fecha	08/03/10
Descripción del problema	-
Impacto del problema	-
Soluciones adoptadas	• Se ha generado el documento de planificación inicial.
Anexos a la versión	

Apéndice 0.2

Fecha	09/03/10
Descripción del problema	Google Groups no tiene capacidad de gestionar tareas, el sistema de subida de ficheros es muy básico, no permite estructura de directorios, con el caos que ello conlleva.
Impacto del problema	Un trabajo en equipo más desorganizado y más difícil localizar ficheros y distinguir a qué subgrupo e iteración pertenecen.
Soluciones adoptadas	• Se ha realizado una modificación en la página 6, en la tabla "TablaDeRecursosSoftware", se ha cambiado el sistema Google Groups por el sistema Awacate como gestor de comunicaciones del grupo, el cual si gestiona tareas, dispone de foro y permite crear una jerarquía de carpetas.
Anexos a la versión	

Apéndice 0.3

Fecha	10/03/10
Descripción del problema	Ha entrado un nuevo miembro en el equipo, hay que asignarle una función y un subgrupo. El diagrama de Gantt no reflejaba los hitos deseados. El documento presentaba pequeños errores de expresión y ortografía.
Impacto del problema	El nuevo miembro necesita tener asignado un trabajo para continuar con el correcto funcionamiento del grupo. Sin los hitos reflejados no se puede saber en qué fechas se espera entregar cada documento.
Soluciones adoptadas	• Añadido un nuevo miembro al grupo de implementación. • Añadidos hitos al diagrama de Gantt, figura 2. • Se han corregido errores en el diseño de la lista de tareas. • Corregida una expresión en el apartado de "Análisis de los Riesgos". • Corregidos errores en el índice (No aparecía el Apéndice 0.2).
Anexos a la versión	

Apéndice 0.9

Fecha	13/03/10
Descripción del problema	El documento tenía errores de diseño en la portada, en la lista de versiones y en los documentos de control de cambio. El documento no entraba en detalle en la gran mayoría de los puntos tratados. Se necesitaba especificar el formato de generación de código y de los documentos. Faltaban tareas por tratar en la organización temporal. Necesitábamos una descripción del funcionamiento las operaciones a nivel de diseño, de manera que pudiéramos tenerlas en cuenta en la implementación. También tener en cuenta el diseño de la base de datos. Otra tarea faltante era conocer el aspecto que debería tener el programa, realizar un diseño de su interfaz. Por último era necesario conocer cuándo se montaría la base de datos.
Impacto del problema	Los errores de diseño estropean la lectura, lo que hace el documento más difícil de leer y entender. No entrar en detalle en cada tema supone tener cierto desconocimiento acerca del funcionamiento del grupo y de las decisiones tomadas. El no especificar los formatos conlleva a encontrar documentos y códigos de forma distinta, dando una lectura y comprensión del código más complicada e incómoda de leer. Sin las tareas faltantes, no se puede conocer cuándo se deberían de tratar y, con mayor riesgo, descubrir su importancia con demasiado retraso, lo que podría llevar a tener el trabajo inacabado llegado el momento.
Soluciones adoptadas	• Se ha rediseñado la portada, la lista de versiones y el documento de control de versiones. • Se ha añadido una introducción. • Se han especificado con detalle los puntos “Recursos disponibles”, “Planificación temporal” y “Análisis de los riesgos”. • Se ha añadido una nueva sección “Formato de la documentación”, en la cual se especifican todos los formatos a seguir y las razones de su elección. • Se ha añadido la tarea 21, elaborar los diagramas de colaboración de diseño, teniendo en cuenta los diagramad de clases de diseño definidos. • Se ha añadido la tarea 22, diagrama entidad-relación de la base de datos. • Se ha añadido la tarea 23, diagrama relacional de la base de datos. • Se ha añadido la tarea 25, diseño de la interfaz de usuario. • Se ha añadido la tarea 28, implantación de la base de datos. • Se ha modificado el diagrama de Gantt (Figura 2).
Anexos a la versión	