Documento de PLANIFICACIÓN

Sagres

Sistema de Administración y Gestión de RESTaurantes

Jose David Dionisio Ruiz
Carlos Salas Morales
Ángel García Sánchez
Daniel Guerrero Martínez

V1.0

ÍNDICE DE CONTENIDO

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	09/04/10	Presentación inicial del documento.

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** (**S**istema de **A**dministración y **G**estión de **RES**taurantes).

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.

2. Declaración de alcance

Objetivos

Durante la primera iteración se desarrollaron los subsistemas de Gestión de Cartas, Gestión de bebidas e ingredientes y Gestión de pedidos a los proveedores. Para la segunda iteración, tal y como indicamos en la planificación inicial, nos hemos planteado desarrollar los subsistemas relativos a Gestión de pedidos, Gestión de facturación y Estadísticas. Se contempla en principio la posibilidad del refinamiento del diseño y la implementación de los posibles errores o capacidades incompletas aparecidas en el software resultante de la primera iteración, puesto que a priori, y dado el tiempo invertido en el diseño y la implementación de esta primera iteración, esto podría comprometer el resultado de la segunda iteración. Si con el avance del proyecto se considera que es inviable la corrección de errores, porque se considere imposible el cumplir los plazos previstos y se llegue al acuerdo de que esos errores son subsanables mas adelante, se modificará la planificación reflejando este cambio y reflejando esto en los diagramas correspondientes.

Especificación a alto nivel

Como la especificación a alto nivel de los subsistemas se realizó en la primera iteración, ahora recordaremos solo la especificación perteneciente a los subsistemas relativos a esta iteración.

Gestión de pedidos de los clientes

Cuando se realizan los pedidos se dividen en dos partes: comidas, se pone en la cola de platos del cocinero y Bebidas, se ponen en la cola de bebidas del bar. Cada uno funciona de manera independiente y cuando esté preparado (platos o bebidas) se comunica al camarero para servirlo.

Hay dos tipos de clientes para el restaurante: clientes alojados en el hotel que realizan pedidos desde sus habitaciones y clientes que lleguen al restaurante.

Los clientes del hotel reciben sus pedidos completos mientras que los clientes en el restaurante se les puede servir las bebidas antes de la comida, el pago se puede realizar en efectivo o tarjeta de crédito en caso de los clientes en el restaurante o mediante tarjeta de crédito o tarjeta del hotel en caso de clientes alojados en el hotel.

Los clientes pueden modificar sus pedidos siempre y cuando el cocinero no empezó a prepararlo, en caso contrario solo pueden añadir o eliminar solamente bebidas a su pedido..

Gestión de facturas

La facturación se puede dividir en dos formas: si el cliente realiza la comanda en el restaurante deberá pagar al Metre en el mismo lugar, y si el cliente realiza la comanda desde la habitación, pasaran la factura al hotel que posteriormente le cobrará al cliente.

En la factura debe aparecer el logotipo del restaurante junto con los datos identificativos (nombre, dirección, teléfono, cif, etc).

Los precios son iguales para ambos tipos de clientes.

Estadísticas

El sistema generara algunas estadísticas útiles para el restaurante: estadísticas de facturación, platos mas vendidos o menos vendidos, bebidas más vendidas o menos vendidas.

Viabilidad

Como resultado de la primera iteración, y a tenor de los resultados obtenidos, es posible realizar una primera estimación de la viabilidad del desarrollo. Puesto que para esta iteración se dispone aproximadamente del mismo tiempo que para la anterior, se tratarán de acortar los plazos empleados gracias a la experiencia obtenida por los integrantes del grupo. Se tratará de acortar el proceso de Planificación lo mas posible, para dar posibilidad al equipo de Modelado de requisitos, Análisis y Diseño a realizar su tarea con tiempo, evitando así el retraso ocurrido en la primera iteración. Este retraso trajo como consecuencia una etapa de Implementación mas breve, lo que desembocó en la posibilidad de realizar una fase de pruebas completa. Con este acortamiento del proceso de Planificación se pretende compensar también la mayor dificulta del proceso de Análisis, Diseño e Implementación de esta interación, dada la complejidad de los subsistemas a implementar.

Durante esta interación también podremos ahorrarnos el tiempo de aprendizaje de las herramientas de comunicación entre el grupo, lo cual debería corresponderse con una comunicación mas rapida y eficaz entre los integrantes y entre los grupos de trabajo.

Para especificar el proceso y la duración de las etapas de desarrollo de esta segunda iteración se ha utilizado al igual que en la anterior un diagrama de Gantt, donde se especificarán los hitos de entrega para los distintos equipos, y que también se modificará según vayan los objetivos siendo alcanzados o no por los miembros del equipo. En caso de que no se cumplan algunas se plantean alternativas, se describen, luego se hace un estudio de inversión en el trabajo y de análisis de riesgos, después se planifican las alternativas y se selecciona la solución más adecuada.

Rendimiento

La aplicación no cuenta con complejos procesos de cálculo, por tanto las limitaciones al rendimiento pueden venir en relación a la gestión de la Base de datos. La Base de Datos puede afectar al rendimiento en caso de que el volumen de datos sea superior a lo que la base de datos soporta. Esto podria afectar en nuestro caso a la velocidad de inicio del sistema, y se valorará la conveniencia o no de las decisiones adoptadas durante la primera iteración.

Siguiendo las recomendaciones hechas por el profesor durante la presentación de la primera iteración, se valorara la inclusión de una capa intermedia entre el Gestor de base de datos y los Gestores de los Subsistemas, para facilitar la concordancia entre los datos que maneja el sistema y residen en memoria y aquellos que se almacenan en la Base de Datos. Se valorará también la idoneidad de este proceso, valorando otras alternativas posibles.

Interfaces Externas

Aunque en el caso de una aplicación para gestión de servicios de restauración pudiera ser conviente la comunicación con sistemas externos de pago, para poder permitir el pago con tarjeta de credito, esta eventualidad queda fuera del alcance del desarrollo de esta aplicación por tanto no sera considerado.

Fiabilidad

Se toma la fiabilidad de la aplicación como uno de los aspectos clave en el desarrollo de la aplicación. Se toman distintas medidas y métodos en el desarrollo para conseguirlo:

- Control exhaustívo de los datos que contiene la base de datos, es decir, evitar tajantemente inconsistencias o repetición de datos, para tener calidad en los datos contenidos.
- Interfaz de usuario segura ante errores del usuario, pidiendo siempre confirmaciones antes de realizar operaciones que impliquen cambios en los datos almacenados.

Restricciones

En cuanto a funcionalidad, las acotaciones de funcionamiento se encuentran debidamente delimitadas en los correspondientes documentos de análisis y diseño de la aplicación.

Si que existe una restricción temporal, debiendo estar operativos y entregados los subsistemas arriba mencionados para esta segunda iteración el día 7 de Mayo. Los diferentes hitos referentes a cada proceso se especifican en el diagrama de Gantt correspondiente a este iteración, también incluido en este documento.

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 Martínez, 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 Víctor (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 Víctor
- 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 Martínez, Daniel
- Salas Morales, Carlos

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

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

Esta división ha sido para asegurar que en la última etapa (Que prevemos, 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.

Nombre y apellido	Marca	procesador	Ram	Disco Duro	Sistema operativo
Carlos Salas Morales	Pórtatil ASUS G50V	Core 2 Duo P8600, 2.4 GHz	4 GB DDR 800 MHz	320 GB	Windows Vista Ultimate
Samuel Guirado Navarro	Sony VAIO VGN-FE28H	Intel Core Duo 1,6 GHz T2200	1 GB DDR	160GB	Windows XP sp3
Adrián Víctor Pérez Lopera	Asus Z92J	Intel Core Duo T2250 1.7ghz	2 GB DDR2	Hitachi Travelstar 4K120 100 GB	Windows Vista
Gaspar Muñoz Soria	Airis N1005	Intel Core2Duo 1,66 Ghz	1 GB RAM	100 GB Hd	Windows Vista
Carlos Moreno Muñoz	HP	Intel Core 2 Duo T7250 2 GHz	2 GB RAM	500 GB	Windows Vista
Daniel Guerrero Martínez	HP DV3550es	Intel Core 2 Duo 2.66GHz	4GB RAM	320GB	Windows 7/Ubuntu 9.10
Sergio Rodríguez Lumley	HP DV3550es	Intel Core 2 Duo 2.66GHz	4GB RAM	320GB	Windows 7/Ubuntu 9.10
Sabeg Nabil	Compaq presario 740es	Intel Pentium dual_core 1,7 GHz	2GB RAM	160 GB	Windows 7
Ángel Luis García Sánchez	Dell Studio XPS 13	Intel Core 2 Duo P8700	4 GB RAM	500 GB HDD	Archlinux 64 bits y Windows Professional 64 bits
Jose David Dionisio Ruiz	Sony vaio fw22e	Core 2 Duo P8400 2,26 Ghz	4 GB RAM	220 GB	Windows 7 64 bits

El grupo dispone de un servidor remoto privado para realizar las pruebas necesarias de conexión a servidores y para construir el programa deseado. El Sistema de Gestión de Bases de Datos es MySQL 5.0.51. El driver será cualquiera compatible con la versión 5 de MySQL. El driver hay que descargarlo de la web de MySql. El servidor que corre el SGBD es un Intel Celeron 1.2, con 512 MB de RAM. El sistema operativo es Ubuntu Hardy Heron.

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 ahí:

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 Enterprise Architect 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 apenas 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, sí 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.

Ademas se incluye la experiencia obtenida en la 1º iteración de este software.

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 verde (centro a la derecha), en naranja (abajo a la derecha) y Azul claro (abajo)) son el subsistemas de gestión de pedido cocina, gestión de pedido bar, y gestión facturas. 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.

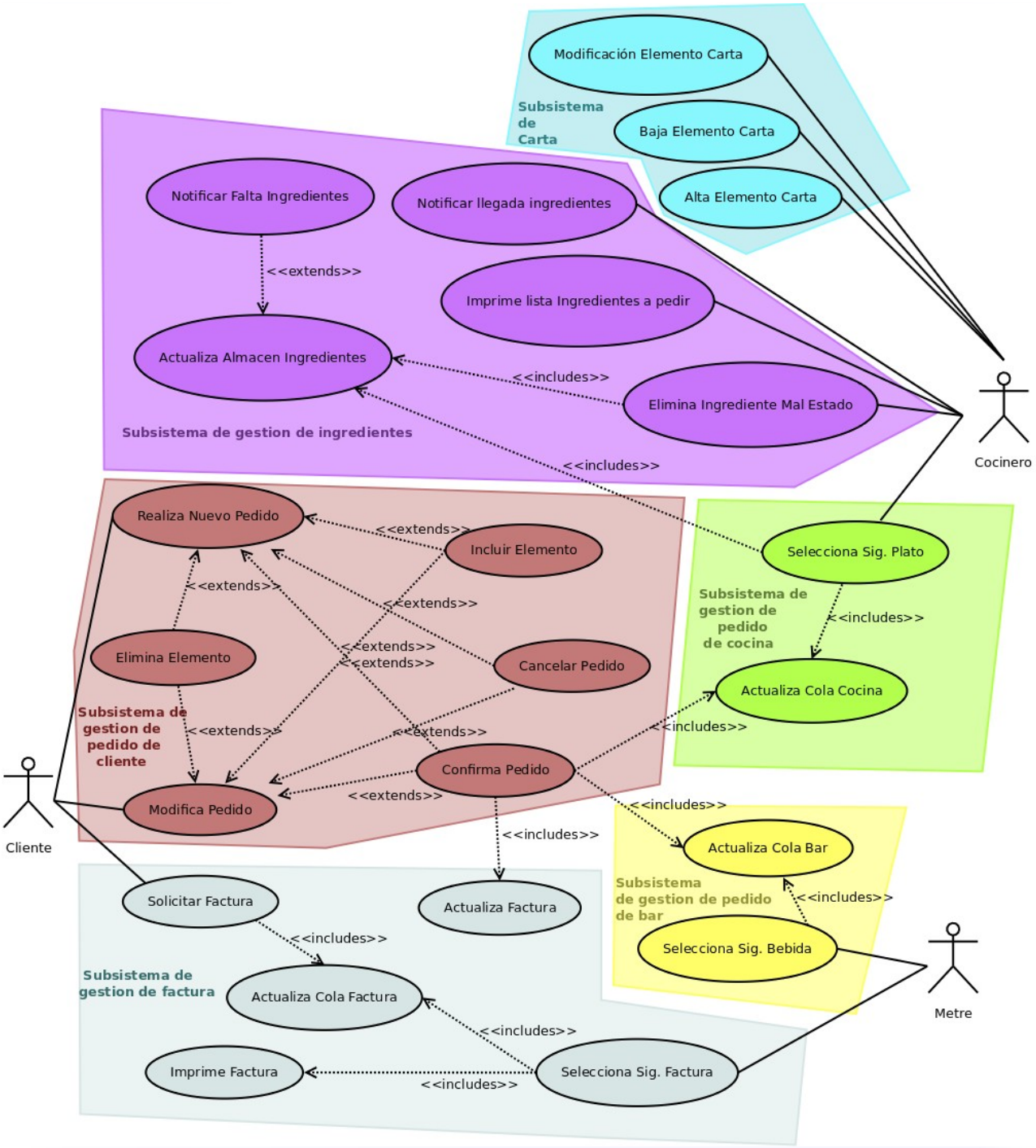

Figura 1. Boceto de diagrama de Casos de Uso de sistema.

Descripción de tareas

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

Planificación

Tarea 1: Revisión del documento de Planificación (1 Iteración). (2 hora)

Tarea 2: Revisión de la planificacion durante toda la iteración. (10 hora)

Tarea 3: Planificación temporal y Diagramas de Gannt. (4 horas)

Tarea 4: Revisión Modelado Requisitos. (3 hora)

Tarea 5: Revisión Análisis. (3 hora)

Tarea 6: Revisión Diseño. (3 hora)

Tarea 7: Diseño y ejecución de Pruebas. (10 horas)

Modelado de requisitos

Tarea 8: Realizar el modelado de requisitos funcionales, mediante el diagrama de casos de uso del sistema. (3 horas)

Tarea 9: Realizar la especificación detallada de los casos de uso, intentando identificar en cada uno de ellos requisitos no funcionales del problema. (3 horas)

Tarea 10: Realizar descomposición del sistema en subsistemas funcionales, mediante el diagrama de paquetes funcionales. (3 hora)

Tarea 11: Llevar a cabo la identificación de requisitos no funcionales del problema. (3 hora)

Tarea 12: Realizar los diagramas de secuencia del sistema. Se realiza un diagrama por cada caso de uso. (10 horas)

Tarea 13: Realizar una lista con las operaciones del sistema. (3 hora)

Tarea 14: Realizar el documento de modelado de requisitos. (3 hora)

Análisis

Tarea 15: Identificar en el problema las posibles clases, las relaciones que hay entre ellas y sus atributos. (4 horas)

Tarea 16: Realizar el diagrama de clases del análisis. (4 hora)

Tarea 17: Realizar los contratos de todas las operaciones, obtenidas en los diagramas de secuencia de sistema. (4 horas)

Tarea 18: Definir un diagrama de colaboración por cada contrato, teniendo en cuenta el diagrama de clases de análisis definido. (6 horas)

Tarea 19: Realizar el documento de análisis. (3 hora)

Diseño

Tarea 20: Decidir el estilo arquitectónico que se adecúa mejor a la solución. (1 hora)

Tarea 21: Definir el diagrama de clases del diseño inicial, obteniéndolo a partir del diagrama de clases del análisis. (4 hora)

Tarea 22: 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 23: Eliminar dependencias cíclicas del diagrama de paquetes estructurales. (2 horas)

Tarea 24: Para cada paquete, revisar su diagrama de clases interno, identificando los servicios requeridos y los ofrecidos (interfaces). (4 horas)

Tarea 25: Una vez definidas las interfaces de los paquetes, transformar el diagrama de paquetes en un diagrama de componentes. (2 hora)

Tarea 26: Elaborar los diagramas de colaboración de diseño, teniendo en cuenta los diagrama de clases de diseño definidos. (8 horas)

Tarea 27: Diseño de la interfaz de usuario. (4 horas)

Tarea 28: Realizar documento de diseño. (6 hora)

Implementación

Tarea 29: Implementación del esqueleto de la interfaz de usuario. (6 horas)

Tarea 30: Paso del modelo de objetos a Tablas e implantación en una base de datos (6 horas)

Tarea 31: Implementación de todas las clases de los subsistemas funcionales. (18 horas)

Revisión de la 1ºIteración

Tarea 32: Revisión y diseño de la 1 Iteración (8 horas)

Tarea 33: Implementación de los nuevos diseños realizados en la revisión (10 horas)

Tarea 34: Reconstrucción y adaptación de la 1 Iteración al nuevo diseño (10 horas)

Diagrama de Gantt

	Nombre	Duración	Inicio	Terminado
1	**Planificación**	**28 days?**	**8/04/10 8:00**	**7/05/10 17:00**
2	Revisión del documento de Planificación (1 Iteración)	1 day?	8/04/10 8:00	8/04/10 17:00
3	Revisión de planificación	26 days?	10/04/10 8:00	7/05/10 17:00
4	Diagrama de Gantt	1 day?	9/04/10 8:00	9/04/10 17:00
5	Revision Modelado Requisitos	2 days?	16/04/10 8:00	17/04/10 17:00
6	Revision Análisis	2 days?	23/04/10 8:00	24/04/10 17:00
7	Revision de Diseño	2 days?	30/04/10 8:00	3/05/10 17:00
8	Diseño y Ejecución de Pruebas	4 days	4/05/10 8:00	7/05/10 17:00
9	**Modelado de Requisitos**	**6 days?**	**10/04/10 8:00**	**15/04/10 17:00**
10	Requisitos No Funcionales	1 day?	10/04/10 8:00	10/04/10 17:00
11	Diagrama de Casos de Uso	1 day?	10/04/10 8:00	10/04/10 17:00
12	Especificación detallada de Casos de Uso	1 day?	11/04/10 8:00	11/04/10 17:00
13	Diagramas de Secuencia del Sistema	3 days?	12/04/10 8:00	14/04/10 17:00
14	Diagrama de Paquetes Funcionales	3 days?	12/04/10 8:00	14/04/10 17:00
15	Lista operaciones del sistema	1 day?	15/04/10 8:00	15/04/10 17:00
16	Documento de Modelado Requisitos	6 days?	10/04/10 8:00	15/04/10 17:00
17	Entrega de Documento de Modelado Requisitos	0 days	15/04/10 8:00	15/04/10 8:00
18	**Análisis**	**7 days?**	**16/04/10 8:00**	**22/04/10 17:00**
19	Identificar Clases, Atributos y Relaciones	2 days?	16/04/10 8:00	17/04/10 17:00
20	Diagrama de Clases	2 days?	16/04/10 8:00	17/04/10 17:00
21	Contratos	2 days?	18/04/10 8:00	19/04/10 17:00
22	Diagramas de Colaboración de Análisis	3 days?	20/04/10 8:00	22/04/10 17:00
23	Documento de Análisis	7 days?	16/04/10 8:00	22/04/10 17:00
24	Entrega de Documento de Análisis	0 days	22/04/10 8:00	22/04/10 8:00
25	**Diseño**	**7 days?**	**24/04/10 8:00**	**30/04/10 17:00**
26	Decidir Estilo Arquitectónico	2 days?	24/04/10 8:00	25/04/10 17:00
27	Diagrama de Clases de Diseño	2 days	24/04/10 8:00	25/04/10 17:00
28	Diagrama de Paquetes Estructurales	2 days	24/04/10 8:00	25/04/10 17:00
29	Eliminar Ciclos	2 days	24/04/10 8:00	25/04/10 17:00
30	Indentificar Interfaces	1 day?	26/04/10 8:00	26/04/10 17:00
31	Diagrama de Componentes	1 day?	26/04/10 8:00	26/04/10 17:00
32	Diagramas de Colaboración de Diseño	5 days?	26/04/10 8:00	30/04/10 17:00
33	Diseño de la Interfaz de Usuario	2 days?	24/04/10 8:00	25/04/10 17:00
34	Documento de Diseño	7 days?	24/04/10 8:00	30/04/10 17:00
35	Entrega de Documento de Diseño	1 day	30/04/10 8:00	30/04/10 17:00
36	**Implementación**	**9 days?**	**26/04/10 8:00**	**7/05/10 8:00**
37	Implementacion Interfaz	4 days?	26/04/10 8:00	29/04/10 17:00
38	Implementacion Base de Datos	4 days?	26/04/10 8:00	29/04/10 17:00
39	Implementación de Clases	5 days?	30/04/10 8:00	6/05/10 17:00
40	Entrega de Subsistemas Finalizados	0 days	7/05/10 8:00	7/05/10 8:00
41	**Revisión 1 Iteración**	**13 days?**	**12/04/10 8:00**	**24/04/10 17:00**
42	Revision y diseño de la 1 Iteración	3 days?	12/04/10 8:00	14/04/10 17:00
43	Implementación del Rediseño	5 days?	15/04/10 8:00	19/04/10 17:00
44	Reconstrucción de 1 Iteración	5 days?	20/04/10 8:00	24/04/10 17:00
45	Finalizacion la revisión	0 days	24/04/10 8:00	24/04/10 8:00

Figura 2. Diagrama de Gantt.

Hitos

Fecha	Descripción
09/04/10	Presentación del documento de Planificación
15/04/10	Modelado de Requisitos
22/04/10	Análisis
24/04/10	Revisión 1º Iteración
30/04/10	Diseño
07/05/10	Implementación

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.

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 → Planificación" 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.

Significado de las plantillas

- **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. El diseño ha sido escogido con tonos azules suaves. La razón de coger azules es porque es un color que da tranquilidad y sensación de bienestar, para asegurar que la lectura resulte lo más placentera posible. 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.
        - Cabecera de nivel 1: La letra es grande 16.1, con un color azul-grisáceo 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 azul-grisáceo, 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, sin negrita. El color es azul-grisáceo oscuro. 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 azul-grisáceo 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.

Reglas de los documentos

- Las cabeceras no pueden terminar en dos puntos, o en punto (Por ejemplo, "Estructuras:"), porque rompe la visualización correcta en el índice y, por tanto, en los accesos del documento.
- **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. Aquí encontramos dos apartados:
    - El documento está en formato .odt (u otro modificable).
        - Estos documentos no necesitan terminar con el nombre de la versión, ya que las herramientas de control de versiones (SVN) llevan control de ellas.
        - Estas no son versiones finales, debido a su carácter modificable, por lo que no se deberán consultar como tales.
    - El documento está en formato .pdf, significa que es una versión acabada (Salvo en el caso de que sea precedido por el nombre de su creador).
        - Estos documentos **deben** terminar con el nombre de su versión. Se le añade al nombre " v[version]".

Documento de Planificacion v0.2.pdf.

        - **NUNCA** se borran, son acumulativos.

- Se permite que los documentos comiencen con el nombre de su creador entre corchetes. Estos aunque estén en formato .pdf (Con lo que se siguen las reglas anteriores) no se considerarán como versión final. Esto es para ayudar a la organización de los documentos.

[Sergio] Documento de Planificacion v0.2.pdf.

Consejos prácticos de uso

- Modificar directamente los archivos editables (por ejemplo, para la documentación, los .odt) desde el repositorio, o bien guardarlos ahí tras cada modificación, aunque no se saque una versión final, esto ayuda a que todos los miembros accedan al documento actualizado y puedan añadir sus modificaciones de forma actualizada.
- Actualizar el repositorio cada vez que se va a explorar, por la misma razón que el punto anterior.
- La portada debe mostrar la última versión en la que se encuentra el documento, es decir, debe coincidir con la última fila del "Historial de versiones" y con el número del último apéndice.
- Comprobar que la fecha tanto del historial de versiones como de los apéndices concuerda.
- Englobar los temas relacionados con encabezados, por ejemplo, si vamos a tener varios encabezados superiores que traten los mismos temas, será más correcto englobarlos en una única cabecera y utilizar cabeceras de menor grado. Por ejemplo:

[Encabezado 1] Revisión del documento de modelado de requisitos *[Encabezado 1] Revisión del documento análisis* *[Encabezado 1] Revisión del documento diseño*	*[Encabezado 1] Revisiones* *[Encabezado 2] Documento de modelado de requisitos* *[Encabezado 2] Documento análisis* *[Encabezado 2] Documento diseño*

- Regenerar el índice antes de imprimir el documento en formato 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 la regla de "*Camel Case*" ya que estarán totalmente en 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();
```

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.

Después 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 1.0

Fecha	09/04/10
Descripción del problema	-
Impacto del problema	-
Soluciones adoptadas	• Se ha generado el documento de planificación inicial basado en el documento de planificación 1.5v de la primera iteración. • Se rehace la declaración de alcance con respecto a la nueva iteración • Se especifica una nueva planificación temporal.
Anexos a la versión	• Agregado anexo “Hitos y Diagrama de Gantt 1.0v.pdf” y Diagrama de Gantt.png