La empresa BK ha recibido un nuevo encargo para el desarrollo de una aplicación informática que se describe seguidamente:

En un gimnasio desean tener una aplicación informática para llevar un control de la gestión del mismo. El gimnasio está abierto las 24 horas del día (de lunes a viernes) como servicio novedoso para los clientes.

Dispone de una serie de aparatos para realizar actividades que son reservadas para sesiones de los clientes. Aunque haya varios aparatos iguales se considerarán como si fueran distintos. Cada sesión tiene una duración de una hora. Actualmente no se sabe en qué horas están libres y ocupados cada uno de los aparatos por lo que la aplicación deberá poder generar un listado, para un determinado día de la semana (de lunes a viernes), de las horas en las que está ocupado y por qué cliente.

Por otra parte, los clientes pagan una mensualidad fija e igual para todos, independientemente de la actividad que realicen. Una vez al mes se generan todos los recibos de ese mes para los clientes. Se desea poder llevar un control de qué clientes han pagado. También se desea poder obtener un listado de clientes que son morosos.

Los clientes acceder al gimnasio mostrando un carnet de socio de la instalación. La aplicación deberá generar estos documentos.

PROPUESTA DE SOLUCIÓN

Determinación y justificación del ciclo de vida que vamos a seguir.

En principio, disponemos de 4 ciclos de vida:

- Cascada.
- Cascada con realimentación.
- Iterativo incremental.
- Espiral.

Cascada: es prácticamente imposible realizar un proyecto de software sin volver atrás en las fases del ciclo de vida, por lo que este modelo lo descartamos.

Por lo expuesto no hay riesgos que hagan peligrar el proceso de desarrollo de software por lo que descartamos el modelo en espiral.

El cliente no ha pedido expresamente disponer de unas partes de la aplicación antes que otras por lo que no es necesario desarrollar versiones sucesivas de la aplicación. Tampoco detectamos motivos para ello. Descartamos el

modelo iterativo incremental. En este caso además, nos evitamos tener que priorizar los requisitos de manera que unos hubiera que abordarlos antes que otros.

Nos quedamos, por tanto, con el modelo en cascada con realimentación.

ANÁLISIS

En base a lo expuesto, la aplicación deberá satisfacer los siguientes:

Requisitos funcionales pedidos:

- R1: Generar un listado de disponibilidad/uso de los materiales.
- R2: Emisión de recibos.
- R3: Relación de clientes morosos.
- R4: Impresión de carnet.

Requisitos no funcionales:

- Disponibilidad 24 horas al día de lunes a viernes.
- Confidencialidad de los datos de los clientes.

Existen algunos requisitos funcionales no pedidos expresamente pero que el cliente podría asumir que ha pedido al trabajar con esas informaciones:

- R5: Listado de clientes.
- R6: Listado de material averiado.

DISEÑO

Realizaremos varios diseños cada uno de los cuales da una visión de la aplicación desde un punto de vista diferente:

Diseño de datos: Se indican los almacenes de información que serán necesarios para satisfacer los requisitos. En nuestro caso, se necesitará almacenar información sobre:

- A1: **Clientes**: número socio, nombre, dirección,…
- A2: **Recibos**: número de cliente, año y mes al que corresponde, pagado o no,….
- A3: **Material**: número de aparato, denominación, averiado o en uso,…
- A4: **Horario**: día de la semana, número de aparato, libre u ocupado, número socio,…

Para comprobar si con estos almacenes de información podemos satisfacer todos los requisitos los representamos en una tabla (técnica matricial).
Cada uno de los requisitos conlleva la realización de una tarea para la cual son necesarias ciertas informaciones. Por ejemplo, para R6 vamos a necesitar información de los clientes (A1) y de los pagos que han realizado y los que no (A3). Pondremos una cruz en la intersección de R6 con A1 y A3.
Una vez construida la tabla deberemos hacer las siguientes comprobaciones:

- Si hay algún requisito en cuya columna no hay ninguna X puede ser debido a:
  - Faltan almacenes de información.
  - Sobra el requisito.
- Si hay algún almacén de datos que no tiene ninguna X puede ser debido a:
  - Este almacén no lo necesitamos.

- Falta algún requisito que use el almacén.

En cualquier caso debemos revisar el diseño.

En nuestro caso tenemos:

VERIFICACIÓN CUMPLIMIENTO REQUISITOS	R1:generar carnet de socio	R2: listado disponibilidad de máquinas	R3: control clientes que han pagado	R4: listado máquinas usadas	R5: generar recibos	R6: listado clientes morosos	R7: informe de caja
A1: Clientes	X		X	X	X	X	
A2: Máquinas		X		X			
A3: Pagos			X		X	X	X
A4: Horario		X		X			

Diseño arquitectónico: establecemos los bloques que tendrá nuestra aplicación. Se trata de agrupar los requisitos funcionales afines en bloques.

Gestión de clientes	**Gestión de máquinas**	**Gestión de pagos**
R1, R6	R2, R4	R3, R5, R7

Todos los requisitos funcionales deben estar en algún bloque.

Diseño de la interfaz: se deben diseñar las pantallas, intercambios de información,… de nuestra aplicación con otros sistemas de información y con los usuarios.

Diseño procedimiental: se describen los programas que se deberían realizar. Para cada programa indicar qué debe hacer, qué datos tiene de entrada y qué salidas (listados, comunicación con otros sistemas,…) debe generar.

Requerimientos para su implantación: se detallan todos los elementos tecnológicos[1] necesarios para la implantación y funcionamiento de la aplicación.

-Lenguaje programación: Java.

-Sistema de almacenamiento: Base de datos MySQL.

-Infraestructuras:

- Ordenadores: sólo uno de sobremesa.
  - Procesador: i5 o equivalente
  - RAM: 4 GB
  - HD: 500 GB
- Red de ordenadores: No
- Webcam: Sí. Para hacer foto al cliente y poderle imprimir el carnet de socio.
- Impresora: Sí, en color.

[1] Algunos de los elementos detallados no se usarán pero se han incluido para tener una perspectiva más amplia de qué tipo de elementos se pueden incluir.

- Sistema Operativo: debe tener Máquina virtual java.
- SAI: (Sistema de Alimentación Ininterrumpida).
- RAID (sistema de discos redundantes): No.

CODIFICACIÓN

Determinaremos los algoritmos más apropiados para los módulos que se han definido en el diseño. En esta fase los implementaremos/programaremos.

Los compilamos y eliminamos los errores de sintaxis obteniendo el código objeto (bytecodes) para su ejecución con la máquina virtual java.

PRUEBAS

Tipos de pruebas unitarias:

- De caja negra: cada elemento de software lo consideramos como una caja negra a la que entran datos y se obtienen resultados. Se hacen pruebas fijándonos sólo en los datos que entran y los resultados que se obtienen.
- De caja blanca: se realizan pruebas en base a la estructura interna de cada elemento de software, considerando los bucles y selecciones que tiene.

Para cada módulo/programa de software haremos pruebas unitarias de caja blanca y de caja negra.

Una vez superadas, haremos pruebas de integración.

DOCUMENTACIÓN

Se generará:

- Guía técnica.
- Guía de uso.
- Guía de instalación.

EXPLOTACIÓN

Se instala la aplicación y configura en los equipos del cliente.

Por último realizaremos la beta test en la instalación del cliente.

MANTENIMIENTO

Se pacta con el cliente un contrato de mantenimiento. No obstante, suele haber un periodo de garantía durante el cual nos comprometemos a asumir los errores que aparezcan en el normal funcionamiento de la aplicación.