Ejercicio 1 (2.5 puntos)

La empresa BK, tras el desarrollo exitoso de la aplicación de venta de productos cosméticos nos ha encargado el desarrollo de una aplicación para móviles que permita a sus clientes consultar y realizar compras de sus productos. Los objetivos que persiguen con esta aplicación son los siguientes:

- Consultar los productos por categorías.
- Posibilidad de recibir notificaciones en el móvil con las principales novedades.
- Visualización de ofertas en función de las preferencias y compras previas de los clientes.
- Realización de pedidos y consulta del estado de un pedido ya realizado.

Tu tarea consiste en diseñar la planificación del proyecto software encargado, teniendo en cuenta todos los aspectos estudiados. Para ello lo primero que tendrás que determinar es el modelo de Ciclo de vida y a continuación planificar las distintas etapas de desarrollo del software.

Ciclo de Vida

El modelo de ciclo de vida idóneo es un modelo en cascada con retroalimentación ya que nos encontramos ante un modelo de software clásico, pero donde es necesario introducir realimentación entre etapas, de forma que podamos volver atrás en cualquier momento y corregir, modificar o depurar aspectos necesarios. Dado que no se prevén muchos cambios durante el desarrollo es el modelo más idóneo. Se trata de un proyecto con pocos cambios, poco evolutivo y los requisitos están claros. El inconveniente de este modelo es que puede ser muy rígido para este proyecto.

No obstante, también podría ser válido un modelo iterativo incremental, que está basado en el modelo en cascada con retroalimentación, ya que se trata de una tecnología novedosa donde se van ha producir muchos cambios y es posible que haya que introducir mejoras en diferentes versiones para adaptarnos a los avances tecnológicos. En este caso sería interesante desarrollar varias fases que se repitan y refinan donde se vayan propagando las mejoras de una fase a las siguientes. Con este modelo también evitamos la rigidez que puede introducir el modelo en cascada con retroalimientación.

Basándonos en lo anterior, también sería adecuado un modelo en espiral que nos permita una construcción repetida en forma de versiones mejoradas del proyecto donde se vaya incrementando la funcionalidad de cada una de las versiones desarrolladas. No obstante, este modelo presenta el inconveniente de que es un modelo muy complejo para este proyecto y de los tres presentados es el menos adecuado.

Aálisis de Requisistos del Sistema:

Requisitos funcionales.

- Consultar los productos por categoría.
- Posibilitar la recepción de notificaciones en el móvil con las principales novedades.
- Visualizar las ofertas en función de las preferencias y compras previas de los clientes.
- Realizar pedidos.
- Consultar el estado de un pedido realizado.

Requisitos no funcionales.

- Tratamiento simultáneo de peticiones.
- Tiempo de respuesta rápido del programa.
- Seguridad.

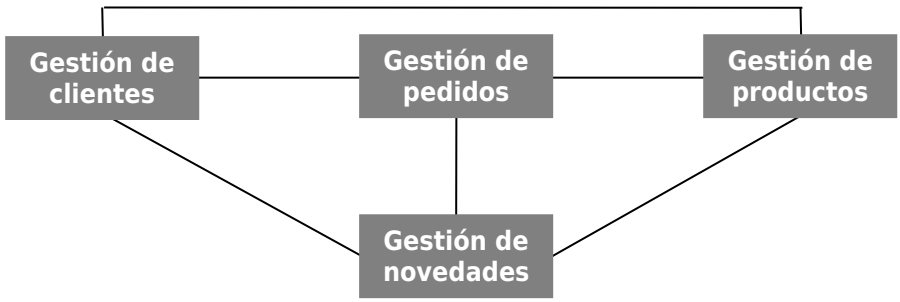
Diseño:

En esta etapa vamos a dividir el sistema en partes y a determinar la función de cada una de ellas. Debemos de determina que hará exactamente cada parte, para ello debemos crear un modelo funcional-estructural de los requerimiento del sistema global y poder así dividirlo y afrontar las distintas pares por separado.

De este modo, podríamos distinguir las siguientes partes:
Gestión de productos. Debe de realizar todas las acciones relacionadas con la gestión de los productos (alta, baja, modificación, consulta). La información de los productos se guardará en una tabla de la base de datos con los siguientes campos: improducto, nombreProducto, descripcionProducto, marcaProducto, modeoProducto, precioProducto.

- Gestión de clientes. Debe de realizar todas las acciones relacionadas con la gestión de los clientes (alta, baja, modificación, consulta). La información de los clientes se guardará en una tabla de la base de datos con los siguientes campos: idCliente, dniCliente, nombreCliente, apellidosCliente, direccionCliente, telefonoCliente, sexoCliente, fechaNacimientoCliente.
- Gestión de pedidos. Debe de realizar todas las acciones relacionadas con la gestión de los pedidos (alta, baja, modificación, consulta). La información de los pedidos se guardará en una tabla de la base de datos con los siguientes campos: idPedido, fechaPedido, idCliente, productosPedido, precioPedido.
- Gestión de novedades. Debe de realizar todas las acciones relacionadas con la gestión de las ofertas (alta, baja, modificación, consulta, configuración). La información de las ofertas se guardará en una tabla de la base de datos con los siguientes campos: idOferta, productosOferta, precioOferta, fechaInicioOferta, fechaFinOferta.

Estos módulos estarían relacionados de la siguiente forma en el sistema.



Tratándose de una aplicación para telefonía móvil, lo ideal es usar un lenguaje de programación multiplataforma que proporcione portabilidad a los distintos sistemas operativos para dispositivos móviles. Un leguaje tan difundido como Java sería una buena elección.

Lo mismo ocurre a la hora de elegir un sistema gestor de bases de datos. Lo ideal es buscar una solución tipo Oracle o MySQL.

Codificación:

En la etapa de codificación de debe de elegir un lenguajes de programación y codificar el programa. La elección del lenguaje de programación para codificar un programa dependerá de las características del problema a resolver, pero teniendo en cuenta que cualquier código fuente, debe de presentar características como modularidad, corrección, fácil de leer, eficiencia y portabilidad. Por ello una buena elección es usar como lenguaje de programación Java, que además de facilitar la consecución de las características ante-

riormente expuestas se trata de un software libre y permite programación orientada a objetos. Al tratare se un lenguaje ampliamente difundido, también dispone de una amplia gama de librerías que puede facilitar mucho nuestro trabajo.

Como entorno de desarrollo se puede usar NetBeans ya que es una herramienta consolidada en el mercado.

Pruebas:

En esta etapa se prueban los programas para detectar errores y se depuran. Generaremos un conjunto de datos de prueba para poder realizarlas. Usaremos un conjunto seleccionado y predefinido de valores límite a los que someteros la aplicación.
En la realización de pruebas es imprescindible para asegurar la validación y la verificación del software desarrollado.

Se efectuarán se siguientes tipos de pruebas sobre el software:

- Pruebas unitarias. Consisten en probar, una a una, las diferentes partes de software y comprobar su funcionamiento (por separado, de manera independiente). Una buena elección para este tipo de pruebas es la herramienta JUnit de Java.
- Pruebas de integración. Se lleva a cabo una vez que se han realizado con éxito las pruebas unitarias y consisten en comprobar el funcionamiento del sistema completo.
- Levaremos también a cabo una prueba Beta Test que se realizará sobre el entorno de producción donde el software va a ser utilizado por el cliente. En este caso, un teléfono móvil.

El período de prueba será pactado con el cliente.

Documentación:

Todas las etapas en el desarrollo deben quedar perfectamente documentadas. Es necesario dar toda la información a los usuarios del software que hemos desarrollado y poder acometer futuras revisiones del proyecto.

Se debe de ir documentando el proyecto en todas las fases del mismo, para pasar de una a otra de una forma clara y definida. De esta forma crearemos los siguientes documentos:

- Guía técnica. El objetivo de este documento es facilitar un correcto desarrollo, realizar correcciones en los programas y permitir un mantenimiento futuro. En él especificaremos el diseño de la aplicación, la codificación de los programas y las pruebas realizadas. Irá dirigido al personal técnico en informática (analistas y programadores).
- Guía de uso. Con este documento buscamos dar a los usuarios finales toda la información necesaria para utilizar la aplicación. Contendrá la descripción de la funcionalidad de la aplicación, la forma de comenzar a ejecutar la aplicación, ejemplos de uso del programa, los requerimientos software de la aplicación y la solución de los posibles problemas que se pueden presentar. Este documento irá dirigido a los usuarios que van a usar la aplicación (clientes).
- Guía de instalación. A través de este documento se pretende dar toda la información necesaria para garantizar que la implantación de la aplicación se realice de forma segura, confiable y precisa. Contendrá toda la información necesaria para poner en marcha la aplicación, para su explotación y para mantener la seguridad del sistema. Este documento estará dirigido al personal informático responsable de la instalación, en colaboración con los usuarios que van a usar la aplicación (clientes).

Explotación:

La fase de explotación es la fase en la que los usuarios finales conocen la aplicación y comienzan a utilizarla. Durante esta fase, se instalará la aplicación, se realizará la puesta a punto y se comprobará el funcionamiento de la aplicación en el equipo final del cliente (teléfono móvil). Los programas serán transferidos al computador del usuario cliente y se configurarán y verificarán. Al tratarse de una aplicación para telefonía móvil, la instalación la llevará a cabo el cliente descargándola desde la página web correspondiente, pero antes de esto, habremos realizado una fase de explotación previa que garantice que no se produce ningún error mientras lo hace el cliente.

En este momento, llevaremos a cabo las Beta Test, que son las últimas pruebas que se realizan en los propios equipos del cliente y bajo cargas normales de trabajo.

Una vez instalada la aplicación, pasaremos a la fase de configuración. En ella, asignaremos los parámetros de funcionamiento normal y probaremos que la aplicación es operativa.

Mantenimiento:

La fase de mantenimiento es la fase en la que el software deberá actualizarse y evolucionar. Deberá ir adaptándose de forma paralela a las mejoras del hardware en el mercado y afrontar situaciones nuevas que no existían cuando el software se construyó.

Por otro lado, siempre surgen errores que habrá que ir corrigiendo y nuevas versiones mejoradas del producto.

Por ello, se pactará con el cliente un servicio de mantenimiento de la aplicación. Este servicio, tendrá un coste temporal y económico.

Durante este proceso se realizará el control, mejora y optimización del software.

También tendremos en cuenta aspectos como nuevas necesidades del cliente, adaptaciones a nuevas tendencias del mercado o nuevos componentes software y la corrección de posible errores futuros.

Ejercicio 2 (2.5 puntos)

Se está desarrollando un software que permita realizar operaciones con números complejos. El siguiente código muestra la codificación de la clase Complejo que ha realizado un programador y que deberás testear adecuadamente.

```
/**
* Clase básica que modela las operaciones de un número complejo
* @author Jose Luis Berenguel
*/
public class Complejo {
private double real;
private double imaginaria;

/**
* Constructor por defecto de la clase. Construye el objeto inicializa-
do
* a los valores (0,0).
*/
public Complejo(){
      this(0,0);
}
```

```
/**
* Constructor con parámetros para inicializar el objeto
* @param real Valor de la parte real del número complejo
* @param imaginaria Valor de la parte imaginaria del número complejo
*/
public Complejo(double real, double imaginaria){
      this.real = real;
      this.imaginaria = imaginaria;
}

/**
* Obtiene el valor de la parte real del número complejo
* @return El valor de la parte real del número complejo
*/
public double getImaginaria() {
      return imaginaria;
}

/**
* Modifica el valor de la parte imaginaria del número complejo
* @param imaginaria Nuevo valor de la parte imaginaria
*/
public void setImaginaria(double imaginaria) {
      this.imaginaria = imaginaria;
}

/**
* Obtiene el valor de la parte real del número complejo
* @return El valor de la parte real del número complejo
*/
public double getReal() {
      return real;
}

/**
* Modifica el valor de la parte real del número complejo
* @param real Nuevo valor de la parte real
*/
public void setReal(double real) {
      this.real = real;
}

/**
* Suma el número complejo <code>c</code> al objeto.
* @param c Objeto a sumar.
*/
public void suma(Complejo c){
      this.real = this.real + c.real;
      this.imaginaria = this.imaginaria + c.imaginaria;
}

/**
* Calcula la multiplicación del complejo <code>c</code> con el objeto.
* @param c Objeto a multiplicar
*/
public void multiplicacion(Complejo c){
    this.real = this.real*c.real - this.imaginaria*c.imaginaria;
    this.imaginaria = this.real*c.imaginaria + this.imaginaria*c.real;
}
```

```
/**
* Compara si dos números complejos son iguales. Dos números complejos
son
* iguales si la parte real y la parte imaginaria de ambos es igual.
* @param c El número complejo a comparar.
* @return <code>true</code> si los números son iguales y
<code>false</code>
* en caso contrario.
*/
public boolean equals(Complejo c){
boolean sonIguales=false;
    if(this.real == c.real && this.imaginaria == c.imaginaria)
        sonIguales = true;
        return sonIguales;
    }
}
```

Las operaciones de números complejos se realizan del siguiente modo. Sean los dos números complejos c1(3,5) y c2(4,8):

- c1 + c2 = (3+4, 5+8) = (7,13);
- c1 * c2 = (3*4 – 5*8, 3*8 + 5*4) = (-28, 44)

Tu tarea será realizar la implementación de los test unitarios para los siguientes métodos:

- Método suma.
- Método multiplicación.
- Método de comparación de igualdad.

Test unitario para el Método Suma

```
/**
 * . Prueba Exitosa.
 */
public void testSumaOk() {
    System.out.println("Test para el método Suma");

    double parteReal1 = 3;
    double parteReal2 = 4;
    double  parteImagniaria1 = 5;
    double  parteImagniaria2 = 8;

    Complejo numComlejo1 = new Complejo(parteReal1, parteImaginaria1);
    Complejo numComlejo2 = new Complejo(parteReal2, parteImaginaria2);

    try{
        numComplejo1.suma(numComplejo2);

        assertTrue( (numComplejo1.getReal()==7) &&
(numComplejo1.getImaginaria()==13));
    }catch (Exception e){
        fail("El método suma ha fallado.");
    }
}
```

Test unitario para el Método Multiplicación

```
/**
 * Prueba exitosa.
 */
public void testMultiplicacionOk() {
    System.out.println("Test para el método Multiplicacion");

    double parteReal1 = 3;
    double parteReal2 = 4;
    double  parteImagniaria1 = 5;
    double  parteImagniaria2 = 8;

    Complejo numComlejo1 = new Complejo(parteReal1, parteImaginaria1);
    Complejo numComlejo2 = new Complejo(parteReal2, parteImaginaria2);

    try{
        numComplejo1.multiplicacion(numComplejo2);

        assertTrue((numComplejo1.getReal()==-
28)&&(numComplejo1.getImaginaria()==44));
    }catch (Exception e){ {

        fail("El método multiplicacion ha fallado.");
    }
}
```

Test unitario para el Método de Comparación de Igualdad

```
/**
 * Prueba exitosa.
 */
public void testEqualsOk() {
    System.out.println("Test para el método Equals");

    double parteReal1 = 3;
    double parteReal2 = 3;
    double  parteImagniaria1 = 7;
    double  parteImagniaria2 = 7;

    Complejo numComlejo1 = new Complejo(parteReal1, parteImaginaria1);
    Complejo numComlejo2 = new Complejo(parteReal2, parteImaginaria2);

    boolean expResult = true;

    try{
        boolean result = numComplejo1.equals(numComplejo2);
        assertEquals(expResult, result);

    }catch (Exception e){
        fail("El método equals ha fallado.");
    }
}
```