Unidad 1.

La empresa BK ha recibido un nuevo encargo de software.

Se trata de diseñar una aplicación para una tienda especializada en vender productos estéticos.

La tienda desea trabajar con software libre.

Además, desea explícitamente que la aplicación sea capaz de cumplir las siguientes tareas:

- Proporcionar facturas de las ventas.
- Llevar la cuenta de lo que vende cada trabajador.
- Controlar el stock de productos en almacén.
- Operar con lector de código de barras y tarjetas de crédito.
- Controlar los precios de los productos y ofrecer la posibilidad de operar con ellos.
- El tiempo de respuesta de la aplicación ha de ser lo menor posible.
- No se podrán procesar dos peticiones a la vez, aunque haya varios equipos funcionando simultáneamente.
- La empresa también quiere almacenar información de sus trabajadores: DNI, nombre, apellidos, número de la Seguridad Social, fecha de nacimiento, teléfono y localidad. Asimismo, de los productos interesa almacenar: código, marca, nombre comercial, precio, cantidad.

Tendrás que diseñar una planificación del proyecto de desarrollo de ese software que cumpla con las premisas estudiadas en la presente unidad de trabajo. Esencialmente, el proyecto se divide en los siguientes apartados:

1. Sintetiza el análisis de requerimientos del sistema para nuestro cliente. Plantea el diseño y determina el modelo de ciclo de vida más idóneo para esta aplicación.
2. Planifica la codificación, indicando el lenguaje de programación y las herramientas que usarías para la obtención del código fuente, objeto y ejecutable, explicando por qué eliges esas herramientas.
3. Planifica las restantes fases del ciclo de vida, indicando en cada una el objetivo que persigues y cómo lo harías.
4. Indica el ciclo de vida que usarías.

Análisis

Después de reunirnos con el cliente y escuchar y anotar todas las especificaciones que requiere el **software que vamos a desarrollar**, en primer lugar decidimos que **el ciclo de vida** que vamos a implementar en este proyecto es en **cascada con realimentación** debido a que tiene unos requisitos claros ya definidos y prevemos que realizaremos pocos cambios en él y si surgieran, podremos volver atrás gracias a la naturaleza de este tipo de ciclo de vida.

Debido a que el cliente prefiere optar por trabajar con software libre, elegiremos **Java como lenguaje de programación.** Para ello utilizaremos **Netbeans, un IDE open source. El gestor de bases de datos que utilizaremos será PostgreSQL, un potente gestor de bases de datos orientado a objetos y open source.** Más adelante, en el planteamiento de todo el ciclo de vida del software, haremos hincapié en esta elección.

Documento de especificación de requisitos del software.

Procedemos a plantear el **análisis** de todas sus propuestas, estructurando las tareas específicas en el **documento de especificación de requisitos del software**. En él documentaremos **los requisitos funcionales y no funcionales** que hemos tratado con el cliente, el cual quedaría detallado de la siguiente manera:

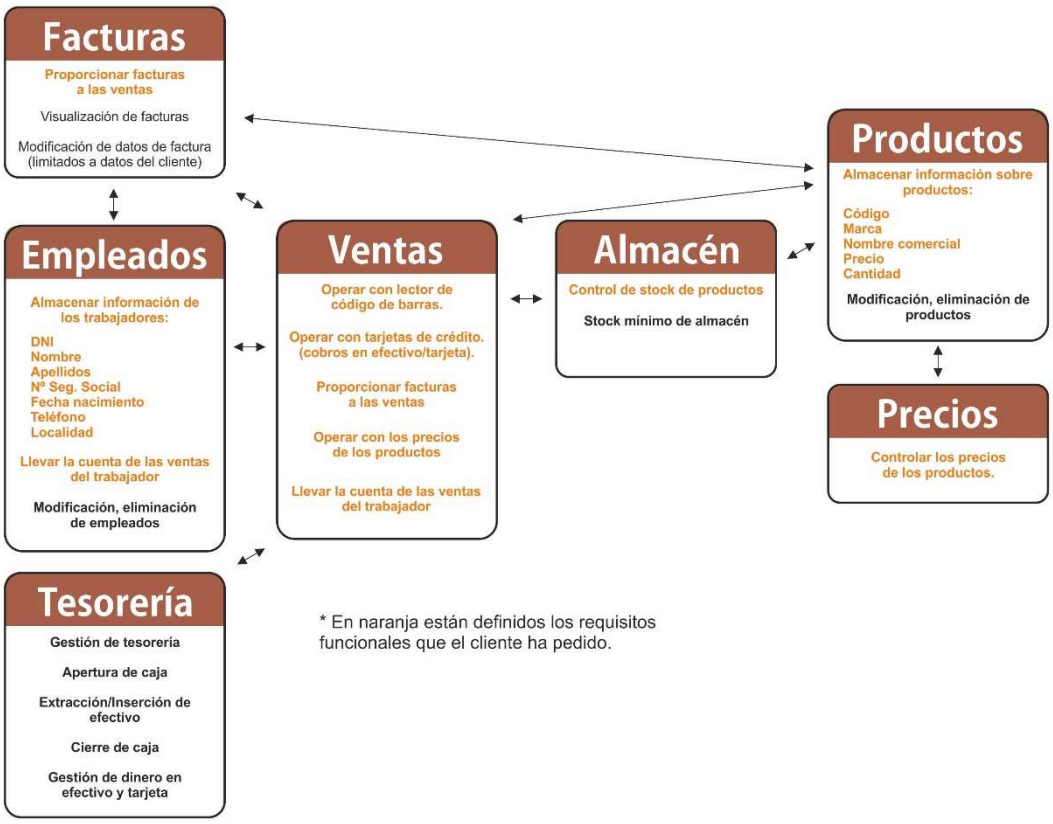
Requisitos funcionales	Requisitos no funcionales
Proporcionar facturas de las ventas	El tiempo de espera de la aplicación ha de ser lo menor posible
Llevar la cuenta de lo que vende cada trabajador	No se podrán procesar dos peticiones a la vez, aunque haya varios equipos funcionando simultáneamente
Controlar el stock de productos en almacén	
Operar con lector de códigos y tarjetas de crédito	
Controlar los precios de los productos y ofrecer la posibilidad de operar con ellos.	
Almacenar información de los trabajadores: DNI, nombre, apellidos, nº de la seguridad social, fecha de nacimiento, teléfono y localidad.	
Almacenar información sobre los productos: marca, nombre comercial, precio, cantidad.	

Diseño.

Documento de diseño de arquitectura.

También desarrollaremos en este punto el **documento de diseño de arquitectura.** Aunque este documento se realiza en la fase de análisis, estamos ya adentrándonos en el comienzo de la **fase de diseño.**

Ciñéndonos a lo documentado en la primera parte de la fase de análisis podemos llegar a la conclusión de que el software tendrá la siguiente estructura global: **gestión de venta, gestión de la caja de tesorería, gestión de facturas, gestión de productos, gestión de almacén, gestión de precios y gestión de empleados:**



Además, tendremos que detallar las funciones individuales de cada una de ellas sin olvidarnos especialmente de los requerimientos del cliente sobre las tareas que debe cumplir:

- **Gestión de ventas de ventas.**
  - **Operar con código de barras)**
  - **Operar con lector de tarjetas** (con lo cual tendremos que implementar los distintos tipos de cobro: efectivo o tarjeta). Ya que los cobros en efectivo hacen modificaciones en la caja de tesorería y en las ventas diarias, mientras que los movimientos en tarjeta solo hacen modificaciones en las ventas diarias.
  - Modificaciones en la caja de tesorería en cada uno de los movimientos de ventas.
  - **Posibilidad de generar facturas a partir de una venta.**
  - Elección de los **productos que se van a vender a través del código de barras**, siempre **y cuando los tengamos en stock, visualizar sus precios y operar con ellos**.
  - **Asignar la venta al empleado.**

- **Gestión de tesorería.**
  - Apertura de caja diario, en el que se puede indicar el dinero en efectivo con el que comenzamos el día.
  - Visualización de los movimientos de caja hechos en el día, diferenciando los movimientos en tarjeta de los movimientos en efectivo.
  - Cierre de caja diario: arqueo de caja.
  - Extracción e inserción de dinero en efectivo para cobros o pagos que son independientes a las ventas diarias.

- **Gestor de empleados**
  - **Almacenamiento de empleados: DNI, nombre, apellidos, número de la seguridad social, fecha de nacimiento, teléfono y localidad.**
  - **Consulta de ventas detalladas de los empleados.**
  - Modificación, eliminación de empleados.

- **Gestor de facturas**
  - Visualización de histórico de facturas
  - **Generar facturas a partir de ventas ya realizadas.**
  - Modificación de facturas. Solo se podrán modificar datos referidos al cliente al que ha sido generada, pero no podremos modificar datos de la venta en sí.

- **Gestor de almacén**
  - **Control de stock en almacén:** podremos añadir o quitar (mermar) la cantidad de productos que ya están creados en la base de datos que tenemos en stock disponibles para su venta.
  - Control de stock mínimo que avise cuando nos estamos quedando sin stock de un determinado producto para adquirir más.

- **Gestor de productos**
  - **Almacenar información de los mismos: marca, nombre comercial, precio y cantidad** (de estas dos últimas solo podremos generar un precio y una cantidad inicial. Para poder modificar estos datos tendremos que hacerlo desde el gestor de precios y el gestor de almacén).
  - Modificación y eliminación de productos.

- **Gestor de precios**
  - Visualización de los precios de los productos que tenemos dados de alta en la base de datos.
  - **Modificación de precios.**

A título personal, creo que en este apartado de diseño. Un factor que no se tuvo en cuenta en la fase de análisis es la **gestión de clientes.** Ya que por ejemplo, las facturas que generemos deben estar asignadas a nuestros clientes, los cuales deberíamos poder almacenar y tener acceso a ellos para gestionarlos. Con lo cual, me parecería acertado reunirnos con el cliente y volver a la fase de análisis antes de continuar.

Documento de diseño del software.

El siguiente paso consistirá en desarrollar el documento de diseño del software, en el cual indicaremos las entidades y relaciones de la base de datos. Es el momento de elegir el gestor de bases de datos y el lenguaje de programación, así como la IDE que utilizaremos para desarrollar el software, aunque ya hemos adelantado esta información al principio.

Para ello tendremos en cuenta la preferencia del cliente por utilizar software libre, pero también el lenguaje de programación que vamos a utilizar, que funcione bien con el gestor.

En este caso el gestor que recomendaría para este proyecto es PostgreSQL, un potente gestor gratuito y open source, que funciona muy bien con un buen número de lenguajes de programación orientados a objetos, entre ellos Java, que es el lenguaje de programación que he seleccionado para este proyecto. Además el JRE encargado de interpretar el programa, también es software libre, cumpliendo con los requisitos del cliente.

Plan de pruebas.

También generaremos en este punto el documento de plan de pruebas de integración, el cual desarrollaremos más adelante en la fase de pruebas.

Una vez terminado el documento de diseño del software ya podemos pasar a la siguiente fase.

Codificación.

Tal y como hemos indicado en el documento de diseño del software, el lenguaje de programación que utilizaremos para este proyecto es Java. Los motivos por los que nos decantamos por este lenguaje son varios.

Permite modularización y es orientado a objetos, con lo cual podremos reutilizar código de otros proyectos anteriores o incluso reutilizar código de este proyecto en otros futuros.

También hemos tenido en cuenta que gracias a estas dos características y a que es un lenguaje de alto nivel, de fácil comprensión para los programadores, su desarrollo y mantenimiento en el futuro será sencillo.

Además, Java es un lenguaje intermediario y eso nos ofrece la ventaja de poder ejecutar el software en cualquier equipo, sea cual sea su sistema operativo.

Ya solo nos falta elegir el programa con el que codificaremos el software del cliente, nos decantaremos por Netbeans, una IDE de open source.

A medida que vamos codificando nuestro programa, no podemos olvidarnos de ir añadiendo comentarios en el propio código y generando documentación que nos facilite la comprensión y modificación del código en el futuro.

Compilación.

El proceso de compilación en java genera distintos archivos de código objeto, a partir de nuestro propio código fuente, lo cual facilita la programación, ya que se puede ir probando las distintas partes del programa individualmente. Luego el JRE (Java Runtime Environment) es el encargado de interpretar el código fuente y ejecutarlo.

Pruebas.

Una vez hemos terminado de codificar el software pasamos a la fase de pruebas, en esta fase podremos volver a la fase de codificación en el caso de que las pruebas no den los resultados esperados. El objetivo no es otro que buscar posibles fallos en la codificación del software y solventarlos, además de dejarlo bien documentado.

Documento de procedimiento de pruebas.

Iremos desarrollando este documento a medida que vamos haciendo pruebas unitarias a las diferentes partes que componen el software.

Documento de procedimiento de pruebas de integración.

Cuando hayamos dado por finalizadas las pruebas unitarias, procederemos a la integración del software completo y terminaremos de generar este documento, el cual comenzamos en la fase de diseño.

Explotación.

Llega la hora de probar el software en los equipos del cliente. Para lo cual, procederemos a la instalación del mismo con el personal de la empresa presente. El objetivo que perseguimos es ofrecer al cliente información detallada de este proceso, así como solventar las dudas que puedan surgir sobre el funcionamiento del mismo.

Una vez instalado el software, configuraremos todos los parámetros necesarios para que esta funcione correctamente (sistema, periféricos, redes, etc…). Finalizada la configuración del software la empresa ya puede proceder a la explotación del mismo.

Mantenimiento.

Es muy probable que sigamos en contacto con nuestro cliente para realizar tareas de mantenimiento, para ello pactaremos o ya habremos pactado anteriormente un servicio de mantenimiento, en él atenderemos a las necesidades de nuestro cliente, ya sea para realizar mejoras en el software (modificaciones perfectivas) o para corregir errores que se detectan en la fase de explotación (correctivas).

El cliente además puede necesitar ampliar las funcionalidades del software y pedirnos modificaciones evolutivas.

Con el tiempo es probable que, con el cambio y la evolución del hardware del mercado, tendremos que realizar modificaciones adaptativas al software.

Documentación.

A lo largo de todas las fases del proyecto hemos ido documentando todas y cada una de las fases del mismo. Esto no solo ha sido útil para ayudarnos a desarrollar el software a lo largo de todos los ciclos de vida del proyecto, sino que además nos será útil en un futuro para consultar cualquier duda sobre el mismo (ya sea para hacer modificaciones en el mismo, para reutilizar partes del código en otros proyectos, etc.). La documentación, una vez terminado el proyecto, estará dividida en 3:

- Guía técnica, en el que se incluirá el diseño, la codificación y las pruebas del programa. Facilitarán en el futuro cualquier modificación a realizar sobre la aplicación.
- Guía de uso, en la cual se incluirá documentación que guíe a los usuarios finales para resolver problemas, requisitos de instalación, funcionalidad…
- Guía de instalación, en la que los responsables informáticos tendrán la documentación necesaria para poner la aplicación en marcha, explotación.