

# ED

**Entornos de Desarrollo (ED)**

**Unidad de Trabajo 1 (UT1)**

**Tarea 04**

## **TA04 Proceso de desarrollo de una aplicación**

### **El problema**

El objetivo de esta actividad es aplicar tus conocimientos sobre el proceso de desarrollo de software a un ejemplo concreto, en el contexto de una reunión entre un cliente y el jefe de proyecto de un equipo de desarrollo.

### **¿Qué tiene que hacer tu equipo de trabajo?**

Imagina que eres parte del equipo de desarrollo de software para una tienda especializada en vender productos de jardinería. Tu tarea será describir detalladamente las fases del ciclo de vida de desarrollo de una aplicación destinada a satisfacer las necesidades de esta tienda. A continuación, se presenta una escena ficticia en la que el cliente y el jefe de proyecto mantienen una reunión para discutir los requerimientos y características del sistema que desean implementar.



## Leyenda

Requisitos funcionales

Requisitos no funcionales

Funcionalidades en base al personal (funcionales)

Funcionalidades en base al producto (funcionales)

### **Escena de la reunión:**

**Cliente:** “Nos gustaría desarrollar una aplicación para nuestra tienda de jardinería. Queremos algo sencillo, pero eficaz. Por supuesto, la aplicación debe ser de software libre, ya que preferimos no depender de licencias caras.”

**Jefe de proyecto:** “Claro, podemos trabajar en esa línea. ¿Qué **funcionalidades** le gustaría que la aplicación cubriera?”

**Cliente:** “Bueno, en primer lugar, necesitamos **generar facturas** para nuestras ventas de manera fácil y rápida. También sería útil llevar un **registro de las ventas realizadas por cada empleado**. Eso nos ayudará a controlar las comisiones y el rendimiento individual.”

**Jefe de proyecto:** “Entendido, una gestión por empleado es importante para monitorizar el trabajo. ¿Qué más necesita la tienda?”

**Cliente:** “El **stock**, es vital saber lo que tenemos en el almacén. El sistema debería poder **actualizarse inmediatamente** cada vez que se venda un algo. Además, sería ideal que la aplicación funcione con lectores de **código de barras y tarjetas de crédito**.”

**Jefe de proyecto:** “Podemos integrar esas funcionalidades. ¿Qué hay sobre la gestión de los precios de los productos?”

**Cliente:** “Sí, necesitamos poder **ajustar los precios** en cualquier momento. A veces realizamos promociones y rebajas y es importante que los cambios se reflejen inmediatamente en la aplicación.”

**Jefe de proyecto:** “Perfecto, lo tendremos en cuenta. Ahora bien, ¿cómo manejan la velocidad de respuesta del sistema?”

**Cliente:** “Preferiríamos que el **tiempo de respuesta fuera lo más rápido** posible, no queremos perder clientes por tener que esperar. Ah, y algo crucial: **no podemos permitir que se procesen dos ventas simultáneamente de un mismo artículo**, ya que eso podría generar errores en el stock.”

**Jefe de proyecto:** “Entendido. En cuanto al personal, ¿qué tipo de datos le gustaría almacenar?”

**Cliente:** “Nos gustaría registrar información importante, como el **DNI, nombre, apellidos, número de Seguridad Social**, etc”

**Jefe de proyecto:** “Eso se puede hacer fácilmente. Y para los **productos**, ¿qué tipo de información deben incluir en la base de datos?”

**Cliente:** “Lo habitual, el **código de producto, la marca, nombre comercial**, esas cosas.”

**Jefe de proyecto:** “Tomo nota de todo. tenemos que hablar de otros aspectos generales...”

**Cliente:** "Sí claro, veamos si este proyecto va a..."

## Desarrollo del proyecto

A partir de esta conversación, deberás diseñar una planificación del proyecto, detallando **cada una de las fases** de desarrollo necesarias para la creación de esta aplicación. Debes seguir las premisas estudiadas en la presente unidad de trabajo.



## **1.Requisitos del trabajo:**

### **III.Análisis de requisitos.**

Aquí se deben identificar las necesidades generales del cliente y planificar los primeros pasos del proceso. Sintetiza los requisitos **funcionales y no funcionales** del sistema, distinguiendo claramente entre los dos tipos. Escribe al menos 15 requisitos. Cumple con las características que deben tener los requisitos. Usa el **formato** adecuado según lo estudiado en la unidad. (DEBES CENTRARTE EN ESTA FASE)

### **IV. Diseño.**

Proporciona una propuesta de diseño del sistema, mencionando los módulos que interactuarán entre sí y seleccionando el **ciclo de vida de desarrollo o la metodología** que consideres más adecuada para este tipo de aplicación. Incluye al menos dos documentos de los que se generan en esta fase según los apuntes.

### **V. Codificación.**

Planifica el proceso de codificación, eligiendo el **lenguaje** de programación y las **herramientas** que utilizarás para obtener el código fuente, el código objeto y el ejecutable. Justifica la elección de estas herramientas

### **VI. Desarrollo de las fases restantes del ciclo de vida del software.**

Describe el objetivo de cada fase (pruebas, mantenimiento, ...) y explica cómo abordarías tú cada una de ellas en este proyecto.

## I. Análisis de requisitos

### Requisitos funcionales (RF):

- **RF01:** El empleado podrá generar facturas.
- **RF02:** El programa registrará las ventas realizadas por cada empleado.
- **RF03:** El sistema contará el stock de productos almacenados en inventario.
- **RF04:** Los empleados registrarán su información personal (DNI, nombre, apellidos y número de Seguridad Social).
- **RF05:** El sistema no permitirá que se procesen dos ventas simultáneamente de un mismo artículo
- **RF06:** El sistema permitirá buscar productos por nombre, categoría o código de barras.
- **RF07:** El usuario podrá consultar la disponibilidad de productos en tiempo real.
- **RF08:** El sistema mostrará recomendaciones de productos relacionados según la compra.

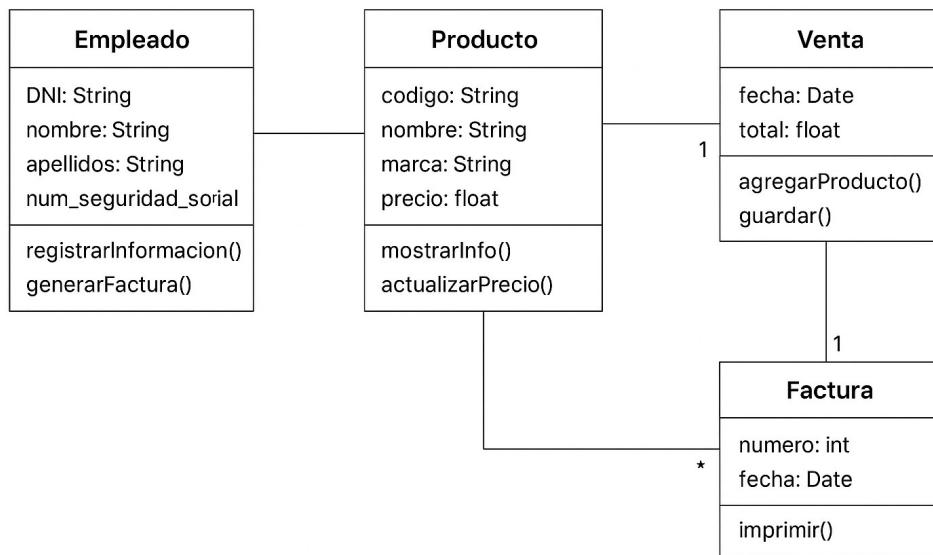
### Requisitos no funcionales (RNF):

- **RNF01:** El sistema se actualizará automáticamente sin interrumpir el servicio.
- **RNF02:** El sistema funcionará correctamente con lectores de código de barras y terminales de tarjetas de crédito.
- **RNF03:** El tiempo de respuesta del sistema será inferior a 2 ms por operación.
- **RNF04:** Las copias de seguridad del sistema se generarán automáticamente cada 24 horas.
- **RNF05:** El sistema mostrará la interfaz principal en menos de 3 segundos tras iniciar sesión.
- **RNF06:** La interfaz del sistema será intuitiva y fácil de usar por los empleados sin conocimientos técnicos.
- **RNF07:** El sistema será compatible con impresoras de tickets y facturas estándar del mercado.

## II. Diseño

### Diagrama de Clases (POO)

**Diagrama de Clases (poo)** en la que muestra gráficamente como sería el sistema del proyecto a crear:



En este ejemplo vemos como la entrada principal es el empleado cuyos datos a de ingresar concordando con lo dicho por el cliente (“Nos gustaría registrar información importante, como el DNI, nombre, apellidos, número de Seguridad Social, etc”), según esto, pasaría al producto a comprar que es compatible con lectores de código de barras (un requisito funcional del cliente), capaz de actualizar el precio en tiempo real (“El sistema debería poder actualizarse inmediatamente cada vez que se venda un algo.”) Finalmente es capaz de generar una factura que sera imprimida y realizar exitosamente la transacción guardando el día y el total de la venta.

## **Metodología escogida**

Para este tipo de proyecto, el ciclo de vida escogido (determinado por el ejercicio) sería un ciclo de vida completo si saltarnos ninguna fase:



Es decir, es un tipo de metodología basada en el modelo de ciclo de vida. Este tipo de metodología se caracteriza por ser menos flexible comparado con otros tipos de metodología (Metodología ágil), además consiste en realizar cada fase una tras otra, sin saltarse ninguna fase. Por el contrario, se puede llegar a detectar más fallos en la fase de implementación debido a la anterior etapa, la de diseño, ya que si esta ha sido realizada de una manera incorrecta, puede haber muchos problemas en la implementación de cada módulo separado.

El escoger esta metodología fue difícil, ya que existen otras metodologías más flexibles y sencillas, sin embargo, para un trabajo eficiente y estricto conforme a los requisitos elegidos por el cliente.

En conclusión, la metodología del modelo de ciclos de vida es la mejor opción para el proyecto sobre un programa dedicado a su utilización en el ámbito de un negocio (en concreto a una tienda de jardinería).

## **III. Codificación**

El cliente necesita un programa capaz de registrar las compras y los datos de los empleados, capaz de ser fluida y sencilla durante los procesos de la interfaz gráfica. En el tema de arquitectura (Elemento investigado independientemente), se escogería una cuyo propósito es poder utilizar el programa en distintos dispositivos (tanto máquinas físicas como móviles y ordenadores). En este caso sería una arquitectura cliente-servidor de tipo web híbrida (PWA o escritorio con backend RESTfull).

## **Backend**

El lenguaje a utilizar sería:

- **Python**- Excelente para desarrollo rápido, junto a un mantenimiento sencillo (para una tienda pequeña) y tiempos de respuestas muy rápidos. Además, es compatible con base de datos y lectores externos, ideal para guardar la información de los empleados, junto a la de las ventas y capaz de utilizar lectores para los productos.
- Alternativa: **Noje.js**- Perfecto si se quiere trabajar con JavaScript full-stack y también proporciona buen rendimiento con lectores externos.

## **Frontend**

Aquí, se necesita ser los más claro y sencillo para que el empleado sea capaz de comprender y utilizar de manera intuitiva:

- **React.js + Tailwind CSS**- Tipo de lenguaje con interfaz moderna, rápida y fácil de utilizar y personalizar. También es compatible con PWA (aplicación que es puede utilizar en tablets y ordenadores)

## **Base de datos**

Se necesita para el guardado o recopilación de información de la ventas y empleados:

- **PostgreSQL**- Lenguaje de base de datos compatible con la mayoría de sistemas operativos y lenguajes de programación.

## **Herramientas**

Otro tipo de herramientas para implementar en el proyecto son:

- **Github** (para gestionar el código y el despliegue del programa)
- **VSCODE** (para generar el código, ligero y compatible con todos los lenguajes)
- **PyTest** (para realizar pruebas automatizadas)
- **Cron jobs + scripts de PostgreSQL** (para crear backups diarios automáticos)

## **IV. Desarrollo de fases restantes del ciclo de vida del software**

Para el resto de las demás fases del ciclo de vida, las veremos por encima:

**Implementación:** Mediante lenguajes de programas, librerías y frameworks, se logrará implementar todas las partes del proyecto como el frontend, backend y la base de datos (hay que

detallar las entradas y salidas del programa). Fijándose en todos los detalles, puesto que esta parte es una de las más complejas para el producto final (Implementando documentación).

**Pruebas:** En esta fase se realizaran una serie de pruebas para verificar el funcionamiento del programa. Mediante la validación, se comprobara que la aplicación cumple con los requisitos acordado con el cliente, y durante la verificación, comprobaremos los errores del programa. Los tipos de pruebas realizados en esta fase pueden ser:

- Pruebas unitaria: Verificar si los módulos funcionan de manera independiente.
- Pruebas de integración: Comprobar que el sistema sea funcional cuando todos los módulos son integrados.
- Pruebas del sistema: Si el cliente lo experimenta y es aceptado por el.

**Explotación:** Instalando el software en un entorno real (por primera vez), se realizan varias prácticas con el programa de forma cotidiana. Se recopilan los errores que pueden aparecer durante el proceso sobre un documento de mantenimiento para que los programadores lo revisen y que corrijan los errores.

**Mantenimiento:** Se realizan cambios conforme a las distintas necesidades (errores, mejoras, adaptaciones). Durante este periodo, las documentación es esencial para dejar registrado los cambios realizados. Según el tipo de mantenimiento realizado se realiza de manera diferente la documentación:

- Correctivo: se corrigen defectos.
- Perfectivo: se mejora la funcionalidad o eficiencia.
- Evolutivo: se añade funcionalidades nuevas.
- Adaptativo: se adapta a nuevos entornos.

**Fase de retirada:** Para finalizar con el ciclo de vida de un proyecto, cuando ya el programa no es rentable para el cliente o ya no es capaz de mantenerlo o se clausura el programa, a la hora de retirar un programa hay que ser cautelosos, especialmente mientras se realiza el proceso de eliminación de datos, para poder exterminar definitivamente los datos más sensible (huella de datos).

Finalmente, se puede comenzar otro ciclo de vida para otro nuevo proyecto.

