



# TALLER PROGRAMACION AVANZADA: ENTREGA FINAL

Integrantes: Juan Almonte ,Sean Castillo.

Correo Electronico:

juan.almonte@alumnos.ucn.cl,  
:sean.castillo@alumnos.ucn.cl.

Rut:22.168.387-0,20.542.721-K.

Paralelo:C1.

Profesor: Tomas Reiman.

Fecha:28 de abril del 2024.

# Índice

Introducción .....	2
Cambios en el diagrama de clase .....	2
Clases empleadas en el programa.....	3
Metódos.....	5
Retos .....	11
Horas Trabajadas .....	11
Conclusión.....	12



Además, introducimos una nueva clase para mantener un registro efectivo de los clientes. Esta lista de clientes nos permitió mantener un seguimiento claro de quiénes eran nuestros usuarios registrados, así como gestionar sus datos y proporcionarles un servicio más personalizado y eficiente.

### **Clases empleadas en el programa**

<b>Clases Empleadas para la lógica del programa</b>	
<b>Empleados-ListaEmpleado</b>	Encargada de la gestión y validación de personal.
<b>Videojuegos-ListaVideojuegos</b>	Gestiona inventario de todos los productos en Stock
<b>Cliente-ListaClientes</b>	Registro de nuevos clientes interesados en nuestros productos
<b>Estadísticas-ListaEstadisticas</b>	Encargada de la gestión de las ventas realizada y de diversas funciones de utilidad para los empleados
<b>Main</b>	Capa de interacción del usuario con los métodos internos .

Hablaremos a detalle que hace cada lista en el siguiente apartado

#### **Clase Utilizada: Empleados**

Los empleados son los principales usuarios del programa y necesitan una experiencia de uso fluida y sencilla que les permita realizar sus tareas de manera eficiente. Para garantizar la seguridad y la autenticación adecuada, hemos implementado validaciones específicas para cada empleado, las cuales están registradas en nuestra base de datos de credenciales.

La clase "ListaEmpleado" actúa como un repositorio de datos de los empleados. Su función principal es utilizar estos datos para llevar a cabo una de las operaciones más críticas del programa: el inicio de sesión, que es la puerta de entrada a las demás funciones disponibles.

Los atributos necesarios para un objeto de tipo Empleado incluyen el nombre de usuario, la contraseña, un código único y un bono asociado. Mientras tanto, los atributos de la clase ListaEmpleados comprenden un arreglo de objetos Empleado, el número máximo de empleados permitidos en la lista, la cantidad actual de empleados almacenados y el empleado activo en sesión en un momento dado.

### **Clase Utilizada: Videojuegos**

GameTech se ha destacado por traer títulos de calidad a nuestra ciudad, ofreciendo una selección cuidadosamente curada de juegos que han sido parte de la infancia de muchos. Para mantener esta tendencia y competir con las grandes multinacionales, es crucial contar con sistemas de vanguardia. Con este fin, hemos creado la clase Videojuego, diseñada para almacenar los datos más relevantes y mantener un inventario preciso de nuestros juegos.

Los atributos de la clase Videojuego incluyen un código único, nombre, precio, género, clasificación de edad, compañía, plataforma, género de descuento y porcentaje de descuento aplicable.

Para gestionar, mostrar y obtener datos de manera eficiente, hemos desarrollado la clase ListaVideojuegos. Esta clase está diseñada para administrar la colección de videojuegos, manteniendo un registro actualizado de los mismos. Sus atributos comprenden un arreglo de objetos Videojuego, que almacena la información detallada de cada juego, así como el número máximo y actual de videojuegos en la lista. Además, hemos incluido un atributo para gestionar la aplicación de descuentos aleatorios, que añade un elemento de sorpresa y atractivo para nuestros clientes.

### **Clase Utilizada: Clientes**

Reconocimos la necesidad de modificar la clase Clientes en relación con el informe anterior para garantizar el correcto funcionamiento del programa. Inicialmente, pensamos en utilizar solo un objeto Cliente sin un contenedor asociado, sin embargo, al proyectar el crecimiento futuro del programa, nos dimos cuenta de que era fundamental considerar escenarios en los que un usuario pudiera atender a múltiples clientes, cada uno con diferentes necesidades y registros.

Por lo tanto, decidimos introducir la clase ListaClientes, diseñada para gestionar y almacenar la información de los clientes de manera eficiente. Esta clase actúa como un

contenedor para los objetos Cliente, permitiéndonos mantener un registro organizado de los mismos. Sus atributos incluyen un arreglo de objetos Cliente, que almacena la información detallada de cada cliente, así como el número máximo y actual de clientes en la lista.

Los atributos de la clase Cliente comprenden el RUT, nombre completo, correo electrónico y estado del cliente. Estos datos son esenciales para identificar a cada cliente de manera única, gestionar su información y ofrecerles servicios personalizados, como ofertas especiales o impedir registros duplicados.

### **Clase Utilizada: Estadísticas**

Originalmente, planeamos calcular las estadísticas a través de múltiples cálculos y métodos relacionados con las clases mencionadas anteriormente. Sin embargo, nos dimos cuenta de que este enfoque causaba problemas durante la construcción del programa. Los métodos necesitaban invocar constantemente a estas clases, lo que no aprovechaba completamente su funcionalidad. Por lo tanto, decidimos introducir un gestor de información que centralizara los datos necesarios de las demás clases y permitiera la realización de métodos específicos. Para esta tarea, asignamos los métodos concretos a la clase ListaEstadísticas.

Los atributos de la clase Estadísticas incluyen datos relevantes para el análisis, como el videojuego ingresado, la plataforma utilizada, la condición de cliente premium, el dinero recaudado, el trabajador con más ventas y la comisión generada.

Por otro lado, la clase ListaEstadísticas actúa como un contenedor para los objetos Estadísticas. Sus atributos comprenden un arreglo de objetos Estadísticas, que almacena la información detallada de cada estadística, así como el número máximo y actual de elementos en la lista. Esta estructura nos permite gestionar y analizar eficientemente los datos estadísticos generados por el programa.

### **Metódos**

La utilización de métodos son una herramienta indispensable de la programación diseñados para realizar una variedad de funciones útiles en el programa.

### **Metódos generales:**

#### **Método buscar:**

El método buscar tiene la función lógica de encontrar datos solicitados por algoritmos siguiendo la estructura de iteraciones dentro de un ciclo “for” junto a la comprobación de si el dato solicitado concuerda con uno dentro de los distintos objetos presentes en el programa.

#### **Método “obtenerPosición”:**

El método obtener posición es complementario a los resultados del “método buscar” con la función de retornar la posición del dato solicitado por los algoritmos dentro del programa.

#### **Método agregar:**

El método agregar cumple con la función de agregar datos externos a los objetos usando iteraciones dentro de un ciclo “for”. Comprobando que los datos ingresados no se repitan dentro del mismos objetos, ya que, el enfoque del código busca que estos sean únicos.

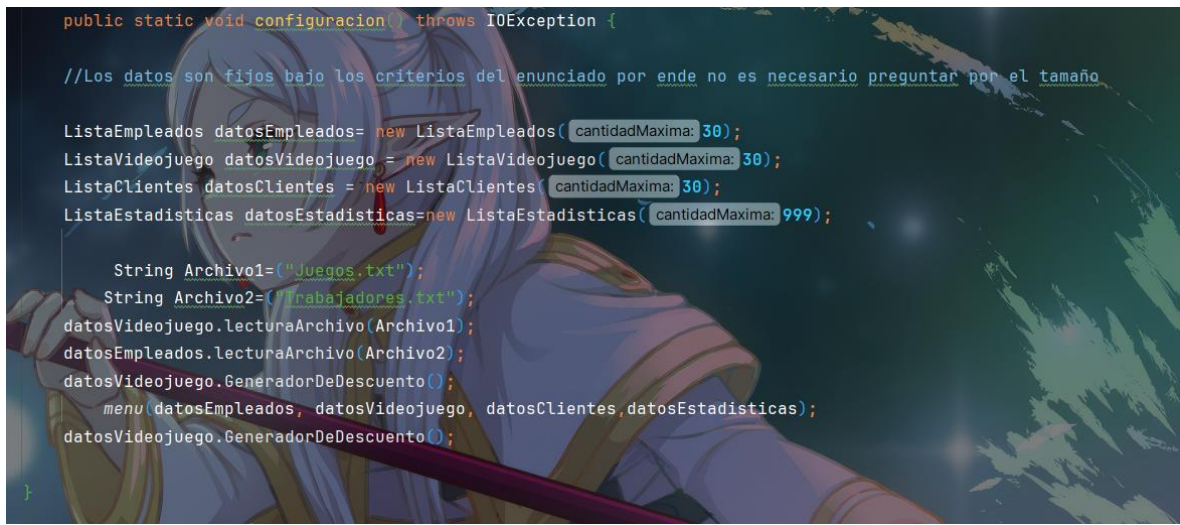
#### **Métodos específicos:**

##### **Método “lecturaDeArchivos”:**

El método “lecturaDeArchivos” tiene el propósito de leer un archivo de texto usando iteraciones dentro de un ciclo, consecuentemente añadirá los datos al objeto correspondiente complementándose con el método agregar descrito con anterioridad.

##### **Método “generadorDeDescuento”:**

El método “generadorDeDescuento” realiza una lógica similar al método buscar descrito con anterioridad, sin embargo, utiliza un ciclo de iteraciones “for” delimitado por una cantidad específica nombrada por los datos de videojuegos presentes para seleccionar un número aleatoriamente extrayendo su género y retornado el mismo para expresar que posee un descuento.



```

public static void configuracion() throws IOException {

    //Los datos son fijos bajo los criterios del enunciado por ende no es necesario preguntar por el tamaño

    ListaEmpleados datosEmpleados= new ListaEmpleados( cantidadMaxima: 30);
    ListaVideojuego datosVideojuego = new ListaVideojuego( cantidadMaxima: 30);
    ListaClientes datosClientes = new ListaClientes( cantidadMaxima: 30);
    ListaEstadisticas datosEstadisticas=new ListaEstadisticas( cantidadMaxima: 999);

    String Archivo1=("Juegos.txt");
    String Archivo2=("Trabajadores.txt");
    datosVideojuego.lecturaArchivo(Archivo1);
    datosEmpleados.lecturaArchivo(Archivo2);
    datosVideojuego.GeneradorDeDescuento();
    menu(datosEmpleados, datosVideojuego, datosClientes,datosEstadisticas);
    datosVideojuego.GeneradorDeDescuento();

}

```

### **Método “Configuracion”:**

inicializa el programa, cargando datos que son de vital relevancia para el funcionamiento adecuado del programa desde archivos de texto, aplicando descuentos a los videojuegos y mostrando el menú principal para que el usuario interactúe con el sistema. Se crean instancias de las clases ListaEmpleados, ListaVideojuego, ListaClientes y ListaEstadisticas para almacenar y gestionar los datos necesarios.

Después de aplicar descuento general a videojuegos, se muestra el menú principal para que el usuario pueda realizar acciones en el programa.

### **Método “Iniciar sesión”:**

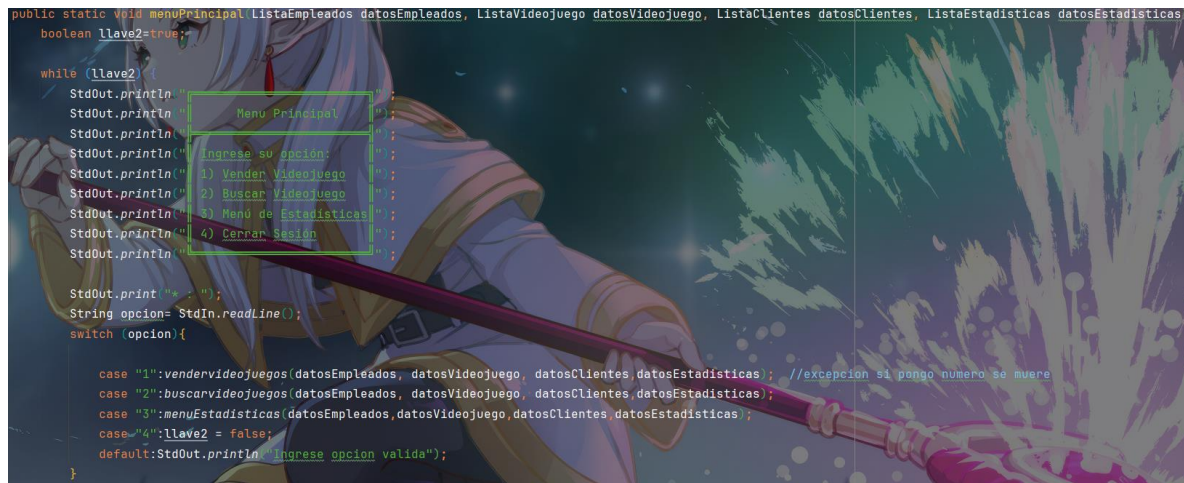
El método “iniciarSesion” se encarga de autenticar a los empleados para permitirles acceder al sistema. Aquí tienes un resumen de lo que hace cada parte del método.

- 1)Solicitud de credenciales: El método solicita al usuario que ingrese su nombre de usuario y luego busca si existe un empleado con ese nombre de usuario en la lista de empleados.
- 2) Verificación del nombre de usuario: Si se encuentra un empleado con el nombre de usuario proporcionado, se solicita al usuario que ingrese su contraseña.
- 3) Verificación de la contraseña: Se busca si la contraseña ingresada coincide con la contraseña del empleado encontrado anteriormente. Si coincide, se muestra un mensaje de bienvenida y se llama al método `menuPrincipal()` para mostrar las opciones del menú principal.

Este método asegura que solo los empleados con credenciales válidas puedan acceder al sistema, verificado por medio de la función equals que coincida con el registro y evitando que no coincidan el usuario y la contraseña.



## Método “menuPrincipal”:

The image shows a screenshot of a code editor with a background illustration of a character in a white and blue uniform, possibly a knight or a wizard, holding a sword. The code is in Java and defines a static method named menuPrincipal. It takes several parameters: ListaEmpleados, datosEmpleados, ListaVideojuego, datosVideojuego, ListaClientes, datosClientes, and ListaEstadisticas, datosEstadisticas. The method starts with a boolean variable llave2 set to true. It enters a while loop that continues as long as llave2 is true. Inside the loop, it prints a menu titled "Menu Principal" with four options: 1) Vender Videojuego, 2) Buscar Videojuego, 3) Menu de Estadisticas, and 4) Cerrar Sesión. It then prompts the user to "Ingrese su opción:" and reads the input. A switch statement handles the options: case "1" calls vendervideojuegos, case "2" calls buscarvideojuegos, case "3" calls menuEstadisticas, and case "4" sets llave2 to false. A default case prints "Ingrese opcion valida". The method ends with a closing brace.

```
public static void menuPrincipal ListaEmpleados datosEmpleados, ListaVideojuego datosVideojuego, ListaClientes datosClientes, ListaEstadisticas datosEstadisticas
boolean llave2=true;

while (llave2){
    StdOut.println("Menu Principal");
    StdOut.println("Ingrese su opción:");
    StdOut.println("1) Vender Videojuego");
    StdOut.println("2) Buscar Videojuego");
    StdOut.println("3) Menu de Estadisticas");
    StdOut.println("4) Cerrar Sesión");
    StdOut.println("");

    StdOut.print "* : ";
    String opcion= StdIn.readLine();
    switch (opcion){

        case "1":vendervideojuegos(datosEmpleados, datosVideojuego, datosClientes,datosEstadisticas); //excepcion si pongo numero se muere
        case "2":buscarvideojuegos(datosEmpleados, datosVideojuego, datosClientes,datosEstadisticas);
        case "3":menuEstadisticas(datosEmpleados,datosVideojuego,datosClientes,datosEstadisticas);
        case "4":llave2 = false;
        default:StdOut.println("Ingrese opcion valida");
    }
}
```

El método menuPrincipal controla el menú principal del sistema, mostrando opciones al usuario y respondiendo a sus selecciones, se utiliza un bucle while para mantener el menú activo mientras el usuario interactúa con él logrando así comunicar los diferentes subprogramas.

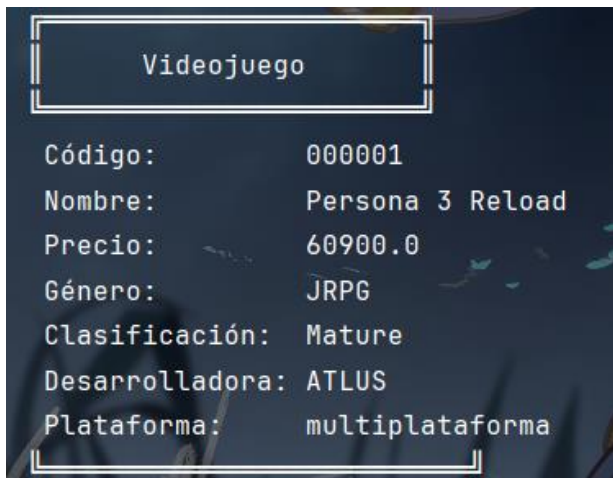
## Método “buscarvideojuegos”:

El método `buscarvideojuegos` se encarga de buscar un videojuego en la lista de videojuegos según el nombre o el código único ingresado por el usuario.

- 1.) Solicitud del dato de búsqueda Solicita al usuario que ingrese el nombre o el código único del videojuego que desea buscar.
- 2.) Búsqueda del videojuego: Utiliza los métodos `BuscarVideojuegoCodigo()` y `BuscarVideojuegoNombre()` de la clase `ListaVideojuego` para buscar el videojuego en la lista según el código único y el nombre respectivamente.
- 3.) Mostrar información del videojuego: La listaVideojuego se encargara de encontrar según los parámetros entregados la posición, si encuentra el videojuego se le devolverá en forma de int la posición para que se muestra su información, incluyendo el código único, nombre, precio, género, clasificación de edad, desarrolladora y plataforma.
- 4.) Volver al menú principal\*\*: Después de mostrar la información del videojuego, se vuelve a mostrar el menú principal para que el usuario pueda realizar otras acciones.
- 5) Manejo de casos de no encontrar el videojuego\*\*: Si no se encuentra el videojuego en la lista esta retornara un (-1) al regresar este número se cumplirá el caso y se mostrara un

mensaje indicando que no se encontró y se vuelve a llamar al método ``buscarvideojuegos()`` para que el usuario pueda intentar nuevamente.

Ejemplo:



#### **Método “VenderVideojuego”:**

El método ``vendervideojuegos`` se encarga de gestionar la venta de un videojuego.

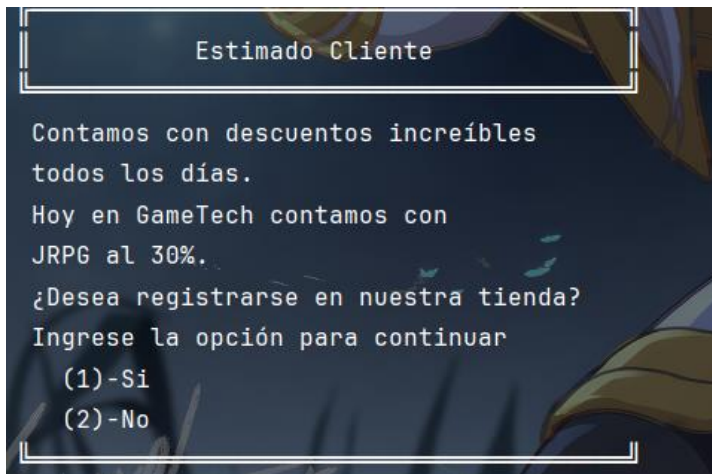
1)Solicitud de datos de entrada: Se solicita al usuario que ingrese el nombre del videojuego que desea vender y su RUT.

Se le solicita el RUT, a diferencia del nombre u otro dato personal debido a la característica que es único en la persona y nos servirá para identificar a los clientes.

2.) Búsqueda del videojuego: Se busca el videojuego en la lista de videojuegos según el nombre proporcionado por el usuario.

3)Mostrar información del videojuego: Si se encuentra el videojuego, se muestra su información.

4.) Verificación del cliente: Se verifica si el cliente está registrado en la tienda. Si no lo está, se le ofrece la opción de registrarse con el incentivo de obtener beneficios en descuentos increíbles creado de manera aleatoria.



5.) Confirmación de compra: Se pregunta al usuario si desea confirmar la compra del videojuego.

6.) Proceso de compra Se realiza la compra del videojuego si se confirma la compra y se ajusta el precio si hay un descuento aplicable. Se registra la venta en las estadísticas del sistema guardando estos datos en la lista de estadísticas para la gestión de datos.

Se creará un objeto de tipo estadística que almacenaremos con los datos mencionados anteriormente.

```

if (posicion3 != -1) {
    if (confirmar.equalsIgnoreCase("si")) {
        if (confirmar.equalsIgnoreCase("si")) {
            System.out.println("Se realiza la compra de: ");
            System.out.println(videojuegoPorTeclado);
            System.out.println(precioConDescuento);
            comision = precioConDescuento * 0.02;
            Estadisticas videojuegoComprado = new Estadisticas(videojuegoPorTeclado, plataforma, clientePremium: "normal", precioConDescuento,
                datosEmpleados.getEmpleadoActivo(), comision);
            registroVentas(datosEstadisticas, videojuegoComprado);
            System.out.println("Compra Realizada");
        }
    }
}

```

7.) Opciones adicionales: Se ofrece al usuario la opción de realizar otra compra o regresar al menú principal.

Este método facilita el proceso de venta de videojuegos, guiando al usuario a través de cada paso y registrando las transacciones de manera adecuada siendo uno de los métodos más relevantes.

### Método “menuEstadisticas”:

El método menuEstadisticas() muestra un menú de opciones relacionadas con las estadísticas del sistema. Aquí tienes un resumen de su funcionamiento:

Mostrar menú de estadísticas: Se muestra un menú con varias opciones relacionadas con las estadísticas, como el videojuego más vendido, la plataforma con más ventas, etc.

Leer opción del usuario: Se solicita al usuario que ingrese una opción y se lee utilizando el método `readLine()`.

Este método proporciona una forma conveniente para que el usuario acceda a diferentes estadísticas del sistema y obtenga información relevante sobre el rendimiento del negocio, se realizaran cálculos para determinar datos relevantes para el negocio.

## **Retos**

El principal reto encontrado durante el desarrollo del taller se puede expresar como la adaptación y comprensión de conocimiento básico visto en la asignatura de “Programación” a la nueva metodología presentada en el ramo “Programación avanzada” respecto a la programación orientada a objetos.

Se nos han presentados diferentes implementaciones que no hemos quedado satisfechos debido al tiempo acotado del taller que pudieron hacer un programa robusto.

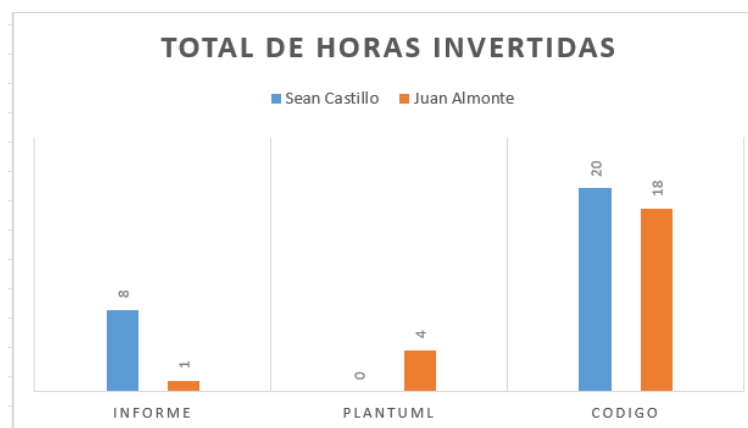
Entre las siguientes:

Implementación de la validación del formato del RUT.

La implementación de la lógica interna de los métodos asociado a estadísticas.

## **Horas Trabajadas**

En el siguiente grafico de barras se puede ver la distribución de horas empleadas en el trabajo.



Juan Almonte se centró en la creación de los diagramas, mientras que Sean Castillo se encargó de redactar los informes basados en los diagramas y el código desarrollado. Para facilitar la colaboración, se dividió el trabajo de manera autónoma y presencial. La colaboración autónoma se realizó a través de plataformas en línea como GitHub y aplicaciones de mensajería como Discord y WhatsApp. Cuando se requería trabajo en equipo presencial, se utilizaban herramientas como Parsec para permitir la comunicación y la colaboración eficientes entre los dos equipos. Esta estrategia permitió una distribución equitativa del trabajo y una coordinación efectiva a pesar de las limitaciones de tiempo y ubicación.

### **Conclusión**

Los cambios realizados en el diagrama de clases fueron fundamentales para mejorar la estructura y la funcionalidad del programa GameTech. Al introducir nuevas clases y optimizar las existentes, pudimos abordar las limitaciones identificadas y garantizar una implementación eficiente de los requisitos del cliente.