

Programacion III

TP MacoWins

Autor: Florencia Skiba

Indice

Resumen

Requerimientos

Consideraciones

Entregables

Diagrama de clases

Diagrama de Objetos

Codigo parcial

# Resumen

En este trabajo se pretende obtener como resultado el diseño inicial de una tienda de ropa. El mismo podrá considerase como la versión 0.0.1 o mockup siguiendo metodología agile.

# Requerimientos

Basados en el enunciado del trabajo practico se identificaron los siguientes requerimientos básicos. Al basarse en los casos de uso de un solo rol/usuario se detectaron algunos Gaps que han sido desarrollados en la siguiente sección.

Requerimientos funcionales:

* Manejo de clientes, empleados y productos.
  + Deberá poder identificar al cliente con atributos como nombre, DNI, tipo de Cliente y dirección.
  + Deberá manejar los empleados de la tienda.
  + Deberá poder identificar articulo o producto de la tienda basado en atributos como nombre, tipo y precio base.
    - Tipo=Saco, Pantalón, Camisa.
* Deberá poder procesar todas las ventas/transacciones:
  + Obteniendo el precio al consumidor basado en la prenda utilizada en la transacción y tipo de operación (estado de la prenda) que podrá ser:
    - Nueva: No se aplicara ningun cambio al precio base.
    - Promoción: Se aplicara un descuento decidido por el empleado al momento de efectuar la venta.
    - Liquidación: Se aplicara un 50% de descuento sobre el precio base.
  + Tipo de pago
    - Efectivo: No se aplicara ningun cambio al precio de venta calculado por el total de los items.
    - Tarjeta: Se aplicara un recargo según la cantidad de cuotas seleccionadas (cantidad de cuotas \* un coeficiente fijo + 0.01 del valor de cada prenda).
* Permitirá generar informe diarios, mensuales y anuales de ventas. Para el reporte diario se deberá poder seleccionar el día especifico con formato dd/mm/aaaa.

# Consideraciones

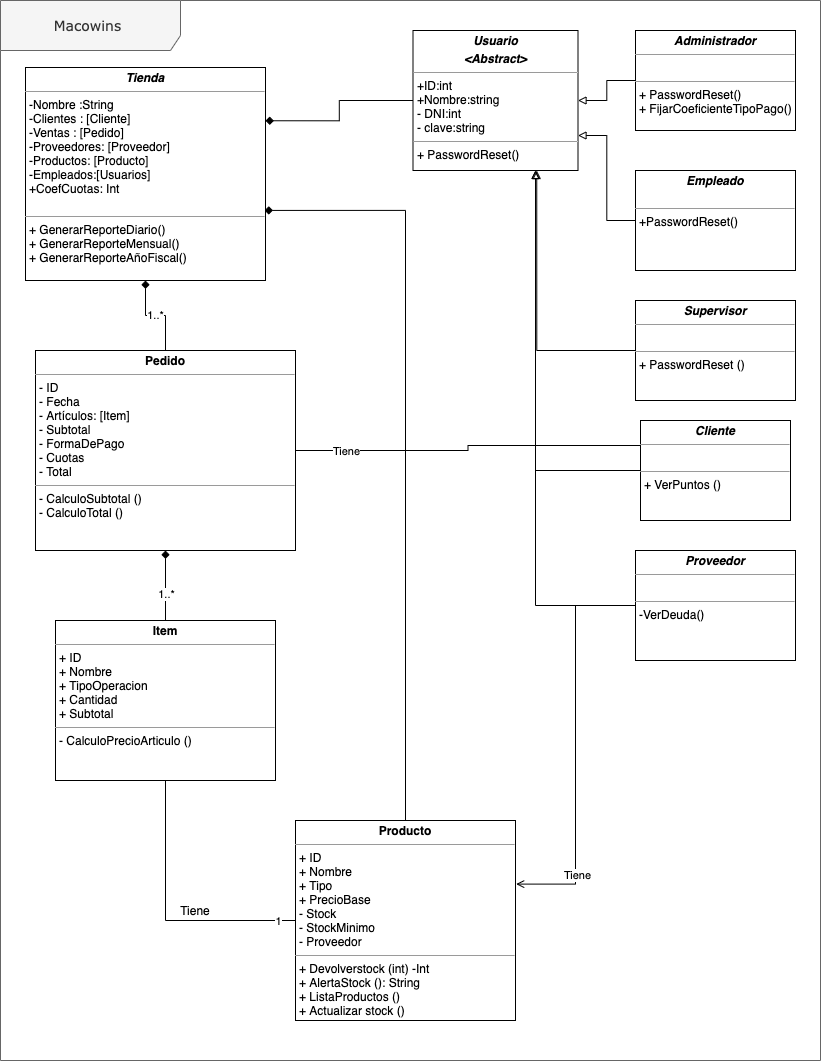
Aunque no surgen de los casos de usos y la entrevista con el usuario, es de suma importancia realizar las siguientes consideraciones o supuestos para un análisis y diseño de una solución integral que sea escalable.

* El proceso de manejo de inventario de productos y proveedores se analizará en una segunda fase de implementación. También se debe analizar tipo de solución y canales de venta lo cual impactara en el diseño final.
* Los usuarios serán identificados como empleados y supervisor en caso de necesitar separar los permisos a futuro.
* No se desarrollará el flujo de trabajo o actividades
* Puedo tener en una venta, productos a los cuales se le aplicara una promoción y otro precio full
* Asumimos que, bajo la modalidad de pago con tarjeta, las cuotas seguirán un modelo [1,2,3,6,12,18]
* El coeficiente fijo será una responsabilidad bajo el rol de administrador de la tienda. De esta forma se reducirá el mantenimiento de aplicación por parte del desarrollador.
* Para los clientes se agrega la opción de fidelización a través de un programa de puntos

# Entregables

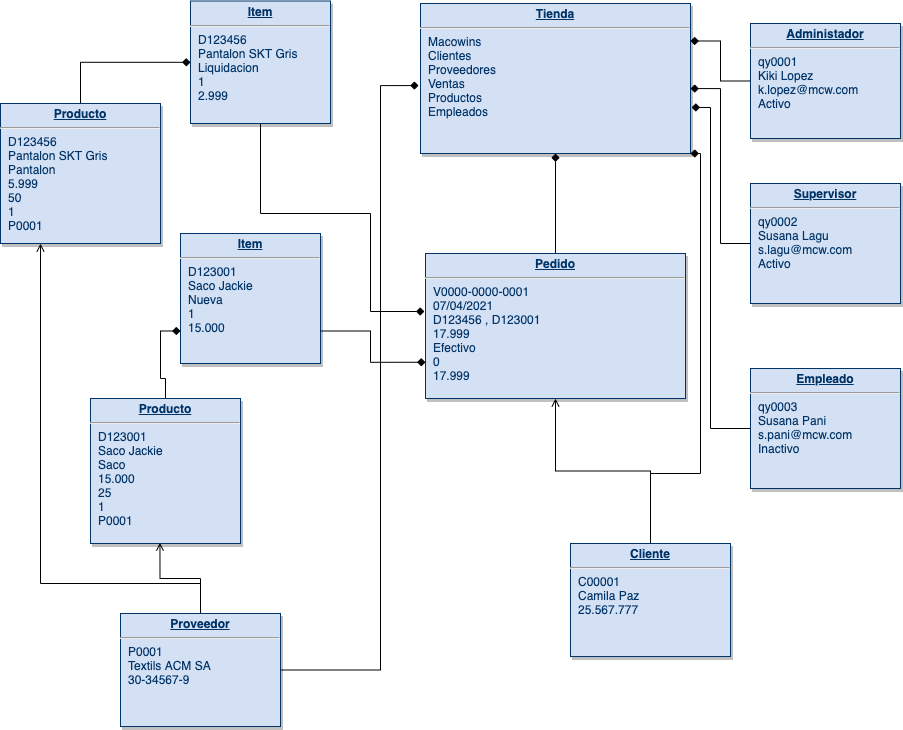
## Diagrama de clases

Basado en los anteriormente mencionado se entiende que el siguiente diagrama será utilizado como un primer borrador y luego del análisis de casos de uso de diferentes perfiles se obtendrá la solución integral.



## Diagrama de Objetos

El principal objetivo del diagrama de objetos es modelar los elementos que están presentes en un diagrama de clases, visualizando o graficando las relaciones entre ellos. También puede ser utilizado durante las entrevistas con nuestro cliente para confirmar o rechazar algún supuesto.



## Código Parcial

Clase Usuario

|  |
| --- |
| abstract class Usuario  { ID: Int;  DNI: Int:  Nombre: string (100);  Password: string(8);  Void PasswordReset()  { // a desarrollar//  }  } |

Clase Administrador

|  |
| --- |
| public class Administrador : Usuario  {  Pin: string(4);    Public override PasswordReset()  { // a desarrollar// }  } |

Nota: Las clases Cliente, Empleado y Supervisor serán similares a Administrador ya que heredan de la clase Usuario

Clase Item

|  |
| --- |
| Double CalculoPrecioArticulo ()  {  Double valor;  If item.Tipo\_operacion=”Promocion”  { desc\_usuario: double desc\_usuario;  desc\_usuario=”Ingrese descuento a aplicar por ser una promocion (0-100)”;  valor= (((Cantidad \* Producto.PrecioBase)\* (100-desc\_usuario))/100);  }  If not item.Tipo\_operacion=”Liquidacion”  { valor =((Cantidad \* Producto.PrecioBase)\* 50)/100);  }  If not item.Tipo\_operacion=”Efectivo”  { valor=((Cantidad \* Producto.PrecioBase);  }  return valor;  } |

Clase Pedido

|  |
| --- |
| Double CalculoSubtotal ()  { n: int  valor: double;    n= Articulos.Cantidad();  valor=0;  Desde i=1 hasta n  valor= valor+ Articulos[i].Subtotal;  return valor;  } |

|  |
| --- |
| Double CalculoTotal (Tienda.CoefCuotas: int)  { Si Pedido.FormaDePago=”Tarjeta”  { n, AD: Int:  n= Articulos.Cantidad;  Desde i=1 hasta n  AD= AD + Articulos[i].Subtotal\*0,01;    return (Pedido.Subtotal + (Pedido.Cuotas\*Tienda.CoefCuotas) + AD)  }  } |

Clase tienda

|  |
| --- |
| GenerarReporteDiario()  { FechaReporte: date;  I, n: int;  n=Ventas.cantidad();  Tot\_fac: double;    Tot\_fac=0;  fechaReport=”Ingrese fecha del cual necesita conocer total facturado”;    Desde i=1 hasta n  If Ventas[i].fecha=FechaReporte  { Tot\_fac= Tot\_fac + Ventas[i].Total }  Imprimir “Total facturado para el dia”, FechaReporte,” es de :”,Tot\_fac;  } |