ÍNDICE

INTRODUCCIÓN 3
DIAGRAMAS DE ENTIDAD-RELACIÓN 4
Diagrama de Entidad – Relación: DER Cheques 4
Diagrama de Entidad Relación: DER Clientes 5
Diagrama de Entidad Relación: DER Datos Generales 6
Diagrama de Entidad Relación: DER Definicion Estados 7
Diagrama de Entidad Relación: DER Mantenimiento 8
Diagrama de Entidad Relación: DER Pagos 9
Diagrama de Entidad Relación: DER Productos 10
Diagrama de Entidad Relación: DER Proveedores 11
Diagrama de Entidad Relación: DER Realización Trabajos 12
Documentos de Arquitectura 13
Estándares de codificación 13
Sobre los Paquetes 13
Sobre las Clases 13
Sobre los Atributos 14
Sobre Constantes 14
Sobre Variables 14
Sobre los Métodos 15
Estilo de Indentación 16
Estándares de Diseño 17
Sobre Ventanas 17
Sobre Mensajes de Error 18
PATRONES DE DISEÑO UTILIZADOS 19
Patrón Fachada 19
Patrón Singleton 20
Patrón Strategy 21
Diagrama de Despliegue 22
Requerimientos Mínimos del sistema 22
Diagramas de Transición de Estados 23
DTE Cheques 23
Cheque 23
DTE Clientes 24
Cuenta Cliente 24
Detalle Cuenta Cliente 24
Detalle Cuenta Trabajo 25
Detalle Pedido 26

Pedido Cliente 26
DTE Proveedores 27
Cuenta Proveedor.....27
Detalle Cuenta Proveedor 27
Pedido Proveedor 27

INTRODUCCIÓN

El Modelo de Diseño consiste en definir la estructura o arquitectura global del sistema.

Éste incluye decisiones acerca de la asignación de subsistemas a componentes hardware y software, y decisiones conceptuales y de política con el objeto de constituir un marco de trabajo acorde a las necesidades del desarrollo.

Presentamos aquí el modelo de datos, representado a través de los diagramas de entidad relación que han sido divididos en partes debido a la imposibilidad de representación en un solo diagrama. A continuación se representan los estándares de codificación y diseño establecidos, finalizando con el diagrama de despliegue del sistema de información.

DIAGRAMAS DE ENTIDAD-RELACIÓN

Diagrama de Entidad – Relación: DER Cheques

Diagrama de Entidad Relación: DER Clientes

Diagrama de Entidad Relación: DER Datos Generales

Diagrama de Entidad Relación: DER Definicion Estados

Diagrama de Entidad Relación: DER Mantenimiento

Diagrama de Entidad Relación: DER Pagos

Diagrama de Entidad Relación: DER Productos

Diagrama de Entidad Relación: DER Proveedores

Diagrama de Entidad Relación: DER Realización Trabajos

DOCUMENTOS DE ARQUITECTURA

Sobre el Desarrollo de Software

Lenguaje de Programación

El lenguaje de programación utilizado es java, mediante la plataforma jdk 1.6 de java en el IDE Netbeans 6.7 con plugin para modelado UML, fachada de persistencia para el acceso a base de datos.

Base de Datos

Se utiliza SQL Server 2000.

Reportes

Jasper Reports sobre java en IDE Netbeans 6.7.

Estándares de codificación

Sobre los Paquetes

- Nombre:
  - Primera letra del cada palabra en mayúscula y sin espacio
  - No se usarán acentos.
  - La letra "ñ" será remplazada por la letra "n".

Sobre las Clases

- Nombre:
  - Primera letra del cada palabra en mayúscula y sin espacio
  - No se usarán acentos.
  - La letra "ñ" será remplazada por la letra "n".
- Documentación:
  - *Fecha creación (dd/mm/aaaa)*
  - *Autor*
  - *Fecha y autor de última modificación*.
- Cláusulas "*import*"
  - Deben ubicarse entre la declaración del paquete y el cuerpo de la clase.
  - Sólo se deben importar las clases que se usen de un paquete.
- Cuerpo de la Clase
  - Orden de Declaración:
    1. Atributos
    2. Constantes

3. Variables Locales
4. Constructor por defecto (sin modificar)
5. Constructor por parámetro (con todos los parámetros si es necesario)
6. Constructor por copia
7. Métodos (variables, código)

- Redefinir el método *toString()* teniendo en cuenta que debe presentar los datos de la clase indicada.

Sobre los Atributos

- Nombre:
  - Primer letra de la primer palabra en minúscula, primera letra de cada palabra siguiente en mayúscula y sin espacio.
- Modificadores de Acceso:
  - Los atributos serán declarados privados.
- Tipo de Dato:
  - Fechas: deben ser del tipo *Calendar*.
  - Considerar el uso de *double* para datos numéricos que alcancen valores altos.
- Inicialización según tipo de dato:
  - Primitivos:
    - *int=0;*
    - *Float=0;*
    - *boolean=false;*
    - *double=0;*
    - *char="";*
    - *long=0;*
  - Si el tipo de datos es NO Primitivo, el atributo debe ser instanciado en el momento de ser declarado e inicializado sin datos. Ejemplo:
    - *String= "";*
    - *ArrayList coleccion= new ArrayList(0);*
- Por cada atributo persistente agregar los métodos getter y setter.

Sobre Constantes

- Declaradas en mayúsculas
- En lo posible deben ser declaradas "static", haciendo de esto modo un uso más eficiente de la memoria.

Sobre Variables

- Nombre:
  - Primer letra de la primer palabra en minúscula, primera letra de cada palabra siguiente en mayúscula y sin espacio.
- Prefijos:
  - Banderas: se declara con un prefijo "fl".
  - Contador: se declara con un prefijo "cont".
- Toda bandera y contador deben estar acompañados de una descripción que indique cual será su finalidad de uso.
- En lo posible deben ser declaradas "static", haciendo de esto modo un uso más eficiente de la memoria.

Sobre los Métodos

- Nombre:
  - Primer letra de la primer palabra en minúscula, primera letra de cada palabra siguiente en mayúscula y sin espacio.
- Modificadores de Acceso:
  - Los métodos públicos serán declarados antes de los privados.
- Cuerpo del Método:
  - Las variables locales serán declaradas al principio.
  - Entre cada línea escrita de código no deben existir líneas en blanco, exceptuando las necesarias para separar:
    - Declaración de variables
    - Estructuras o bloques Condicionales y De Iteración (*if,switch, while,do while, for*).
  - El código debe estar organizado visualmente con tabulaciones(no espacios)
  - Las funciones deben tener un solo punto de retorno
  - El flujo normal de las estructuras de iteración no debe alterarse mediante instrucciones *break* o *continue*.
  - Cada funcionalidad o conjunto de líneas con un nivel de complejidad alto debe estar precedido por una explicación.

- Todo método debe estar documentado con:

  "@return <*tipo-retorno*> <*descripción*>
  @param
  <*nombre-parámetro-1*> <*descripción*>
  <*nombre-parámetro-2*> <*descripción*>…
  <*nombre-parámetro-n*> <*descripción*> "

Estilo de Indentación

Usar un estilo lógico y consistente hace el código más legible, por ello:

- Todos los bloques de instrucciones delimitados por {…}, deben usarlos aún si éstos tienen una sola instrucción.
- La disposición de las llaves según el bloque de instrucción deberá ser:

Estructuras De Iteración

```
while ( ..... ) {
   ..... ;
   ..... ;
   ..... ;
}
```

```
for ( ..... ; ..... ; ..... ) {
   ..... ;
   ..... ;
   ..... ;
}
```

```
do {
   ..... ;
   ..... ;
   ..... ;
} while ( ..... ) ;
```

Estructuras Condicionales

```
if ( ..... ) {
   ..... ;
   ..... ;
   ..... ;
} else {
        if ( ..... ) {
   ..... ;
   ..... ;
   ..... ;
} else {
   ..... ;
   ..... ;
   ..... ;
}
        }
```

```
switch ( ..... ) {
   case ..... :{
          ..... ;
          break;
   }
   case ..... :{
          ..... ;
          break;
   }
   case ..... :{
          ..... ;
   }
  default:{
          ..... ;
   }
}
```

Estructura de Excepción

```
        try {
   .....;
   .....;
} catch (....) {
   ....;
} catch (....) {
   ....;
} finally {
   .... ;
}
```

Definición de Clases

```
..... {
   ... ..... ;
   ... ..... ;
   ... ..... ;
   ... ..... ;
} ;
```

Definición de Métodos

```
.... ... ..... ( .... , .... , .... ){
   ..... ;
   ..... ;
}
```

```
    .... ... ..... ( .... , .... , .... )
    throws ...,..... {
   ..... ;
   ..... ;
}
```

ESTÁNDARES DE DISEÑO

Sobre Ventanas

- Nombre Pantalla
  - Principal:

    PP_CU_<número de caso de uso con tres cifras>
  - Secundaria:

    PS<número pantalla sec.>_**CU**_<número de caso de uso con tres cifras>
- Nombre de los elementos de Interfaz:
  - El nombre debe ser significativo. La primera letra de cada palabra en mayúscula y sin espacio.
  - Al nombre de cada elemento debe anteponerse con el correspondiente prefijo según su tipo:
    - Combo: *cmb*
    - Label: *lbl*
    - Caja de texto: *txt*
    - Tablas: *tbl*
    - Txt Area: *txa*
    - Botones: *btn*
    - Radio Button: *rbtn*
    - Scrool Panel: *jsp*
    - Panel: *jp*
    - Lista: *lst*
    - Check Box: *ckb*
    - Button Group: *btng*
    - Linea de division: *ln*
    - JCalendar: *jcal*
    - JFreeChar: *jfc*
- Formato Fuente:
  - Fuente: Tahoma.
  - Tamaño: 11.
- Disposición de los elementos:
  - Se deberá usar el layout “Free Design”.
  - Para organización de los elementos todos deberán introducirse dentro de un panel principal, llamado “jpPrincipal”.
  - Los botones *Salir* (o *Aceptar* según corresponda) y Cancelar se dispondrán en ese orden, en el extremo inferior derecho de la pantalla.
- Sobre Ventana Principal y Secundaria:
  - Un JFrame debe ser la ventana principal de nuestra aplicación y sólo debe haber una.
  - Las ventanas secundarias de nuestra aplicación deben ser JDialog.

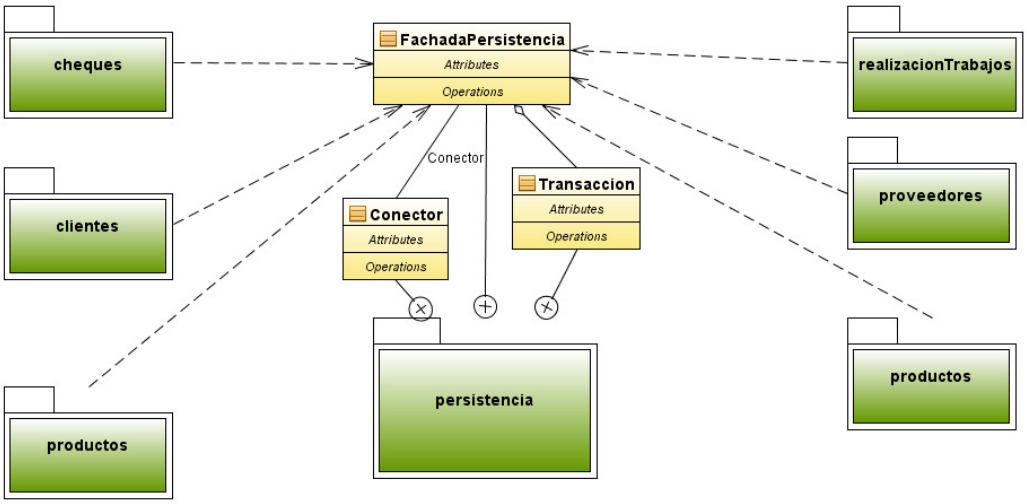
Sobre Mensajes de Error

- Se utilizarán componentes JOptionPane.
- Toda comunicación de errores debe proveer la siguiente información:
    - Qué es lo que ha ocurrido.
    - Cómo afecta al usuario el error.
    - Cuáles son las acciones posibles a partir de ahora.
    - Si el error es evitable en sucesivas ocasiones, qué se puede hacer para evitarlo.
    - A quien debe dirigirse el usuario para solucionar el problema.
- Toda información técnica del error debe poder guardarse en un archivo de texto. Este archivo no se generará automáticamente, sino que el usuario deberá poder elegir si se genera y donde y con qué nombre se guardará.
- Siempre que sea posible se adoptará una política preventiva de control de datos, realizando los controles necesarios para impedir que se introduzcan datos erróneos.

PATRONES DE DISEÑO UTILIZADOS

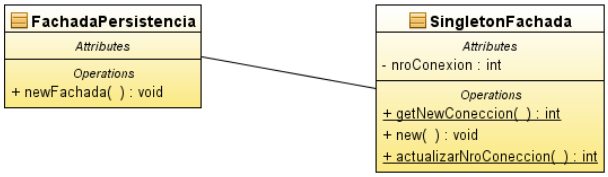
Patrón Fachada

Con respecto al acceso a base de datos se implemento este patrón con el objetivo de proporcionar una interfaz simple a un subsistema, proporcionar un punto de entrada a cada nivel del subsistema, y organizar las muchas dependencias entre clientes y clases que implementan una abstracción; con el objetivo de facilitar a los clientes el uso del subsistema de Base de Datos unificando el proceso de acceso.



Patrón Singleton

Con el objetivo de asegurar que una clase tenga una única instancia y asegurar un acceso global se implementa el patrón fachada. El mismo se implemento en todos aquellos casos donde se requiere una única instancia y también para su acceso. Como ejemplos a mencionar se implemento para la creación de pedidos de clientes y proveedores con el fin de poseer en ese momento una única instancia de los mismo. También se implemento el patrón singleton en la Fachada de persistencia para asegurar que halla una sola fachada creada, por consiguiente una sola coneccion y un único acceso a base de datos, y de esta manera evitar perdida de rendimiento y errores ocasionados por conecciones no cerradas a base de datos.



Patrón Strategy

Se definen una familia de algoritmos, encapsulada cada uno y permite intercambiarlos. Permite variar los algoritmos de forma independiente a los clientes que los usan.

De esta manera configuramos la clase ProductoEmpresa con uno de varios comportamientos posibles dependiendo del tipo de producto que se estableció, y evitando así una cantidad innumerable de secuencias Case.

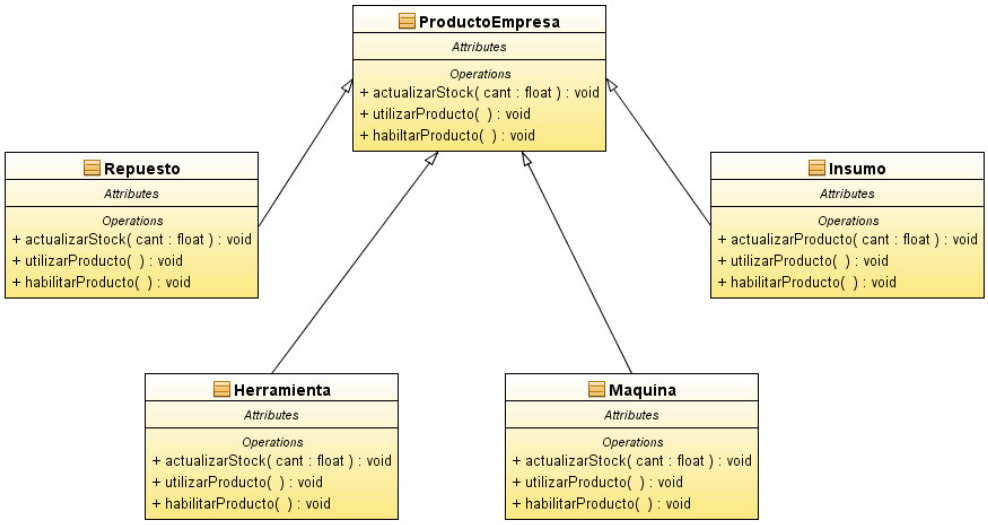
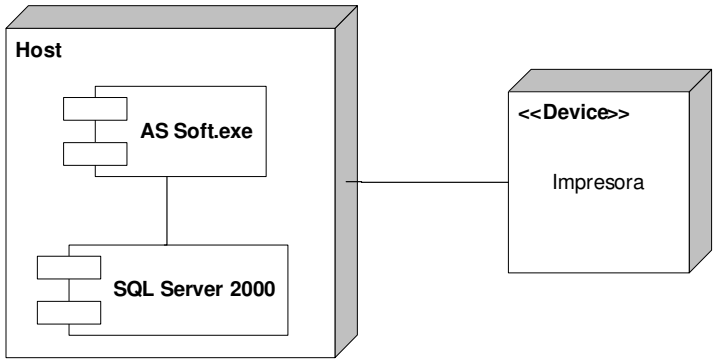


DIAGRAMA DE DESPLIEGUE



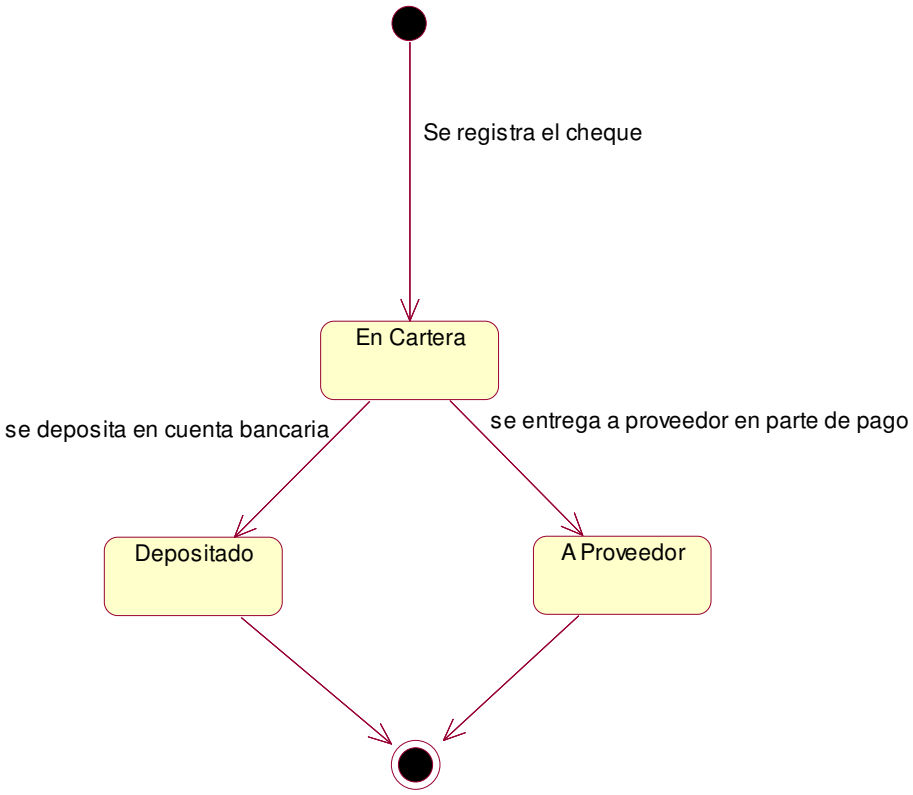
Requerimientos Mínimos del sistema

COMPONENTES	***DESCRIPCIÓN***
Procesador	ADM Sempron 1.6 GHz.
Memoria RAM	512 MB
Disco	SAMSUNG HM252JI
Tipo de Sistema Operativo	S.O 32 bits
Sistema Operativo	Windows XP o superior
Monitor	15 ''
Impresora	Multifunción HP FX 300

DIAGRAMAS DE TRANSICIÓN DE ESTADOS

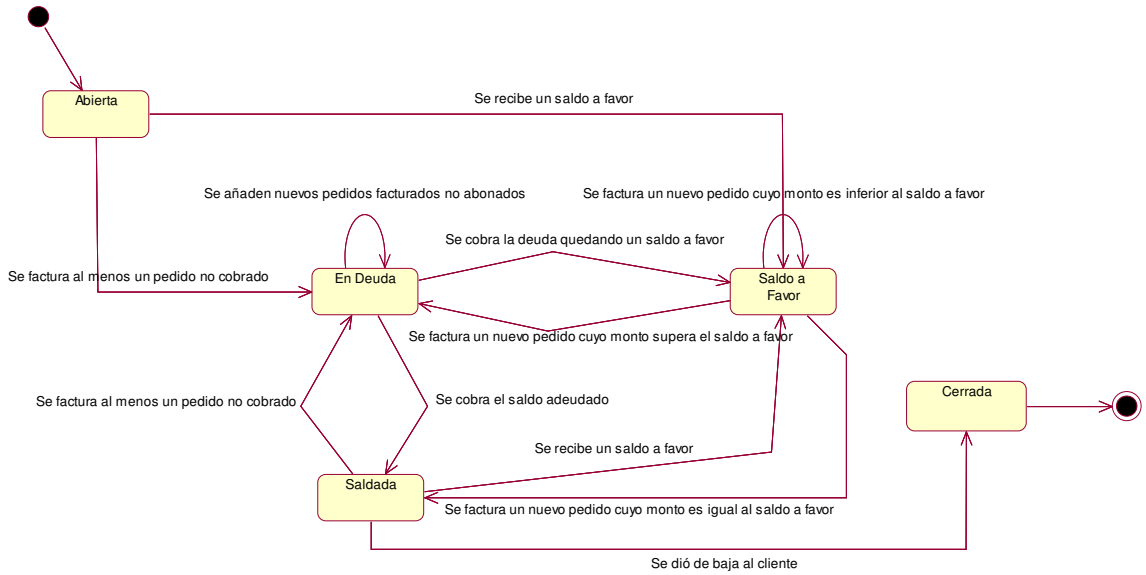
DTE Cheques

Cheque

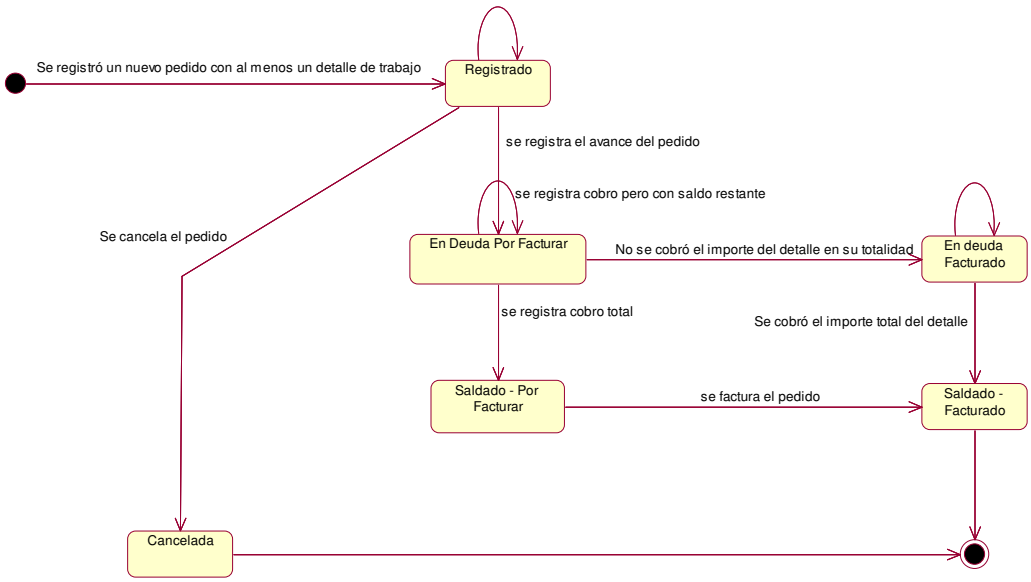


DTE Clientes

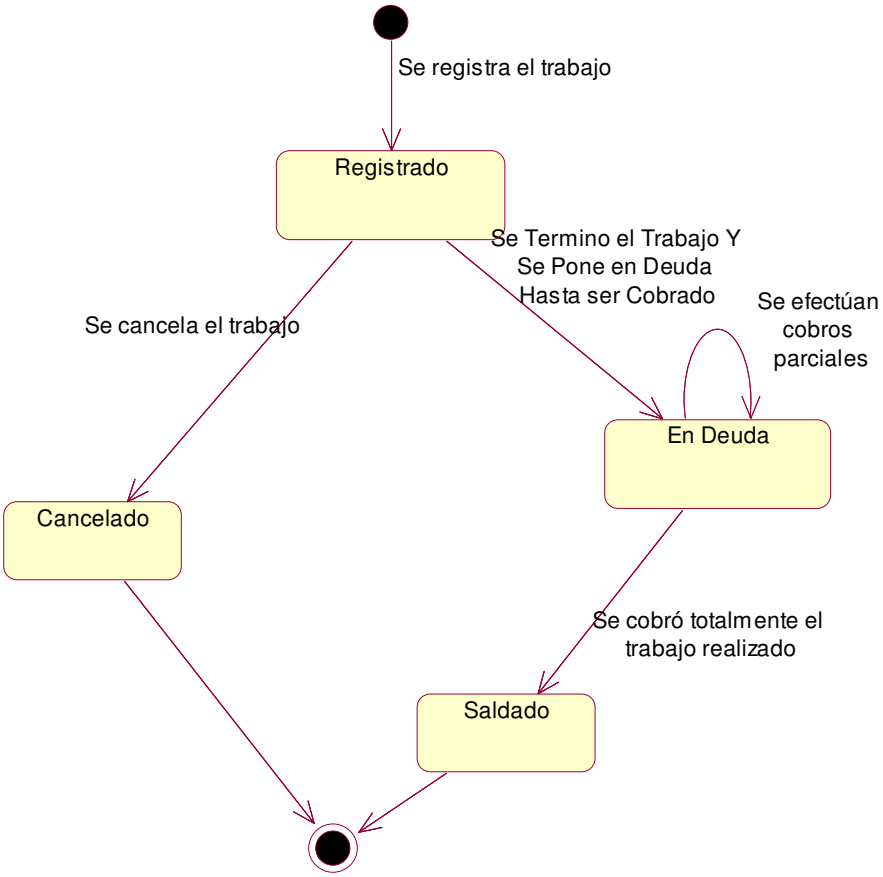
Cuenta Cliente



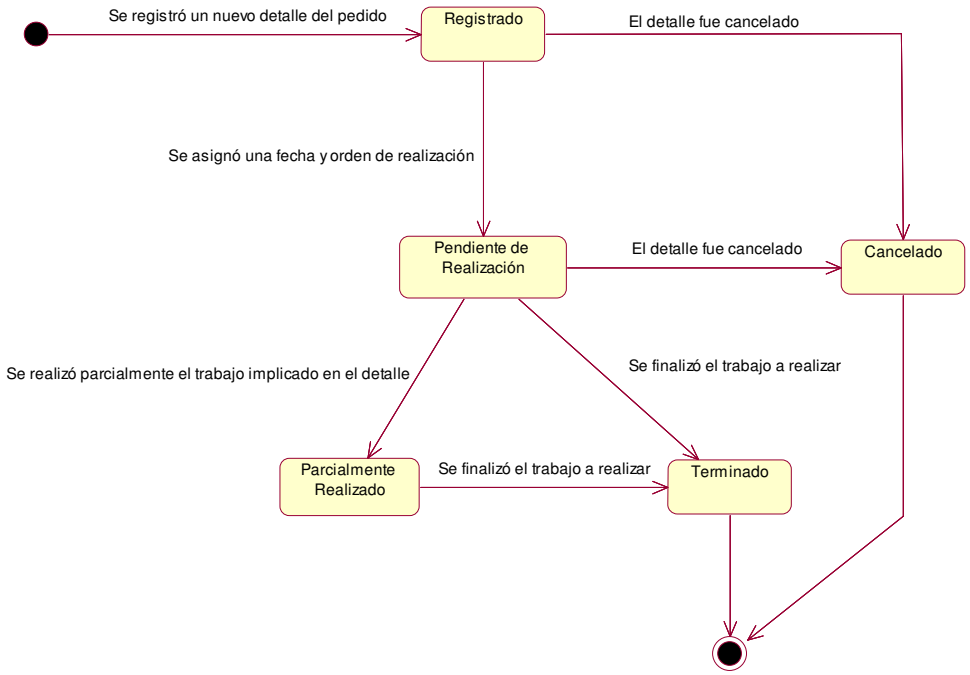
Detalle Cuenta Cliente



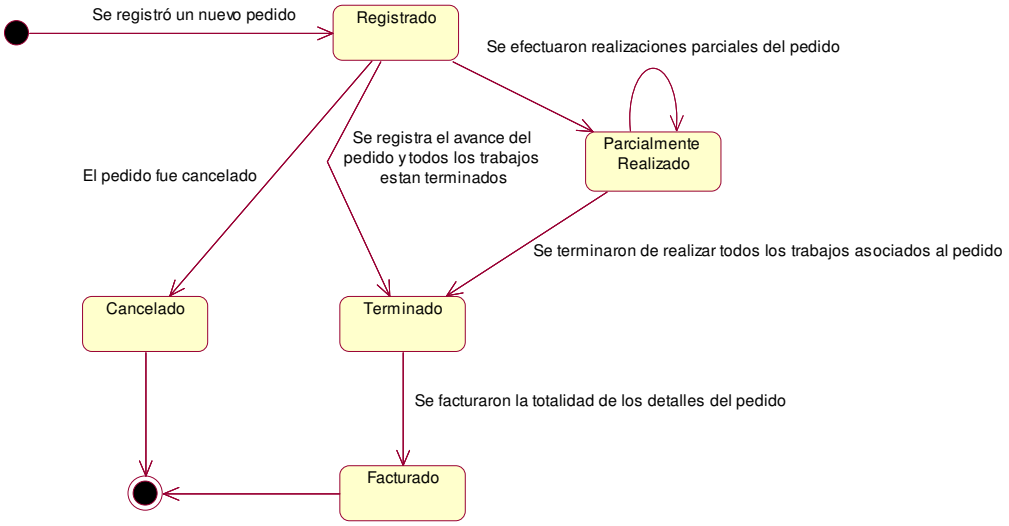
Detalle Cuenta Trabajo



Detalle Pedido

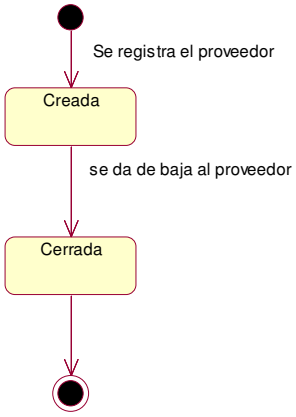


Pedido Cliente

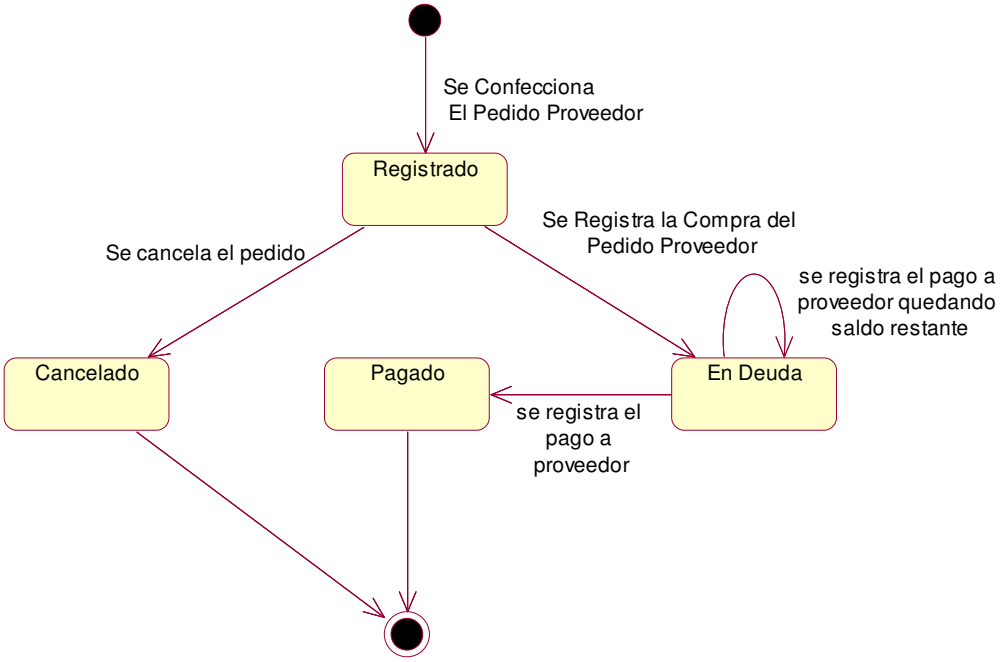


DTE Proveedores

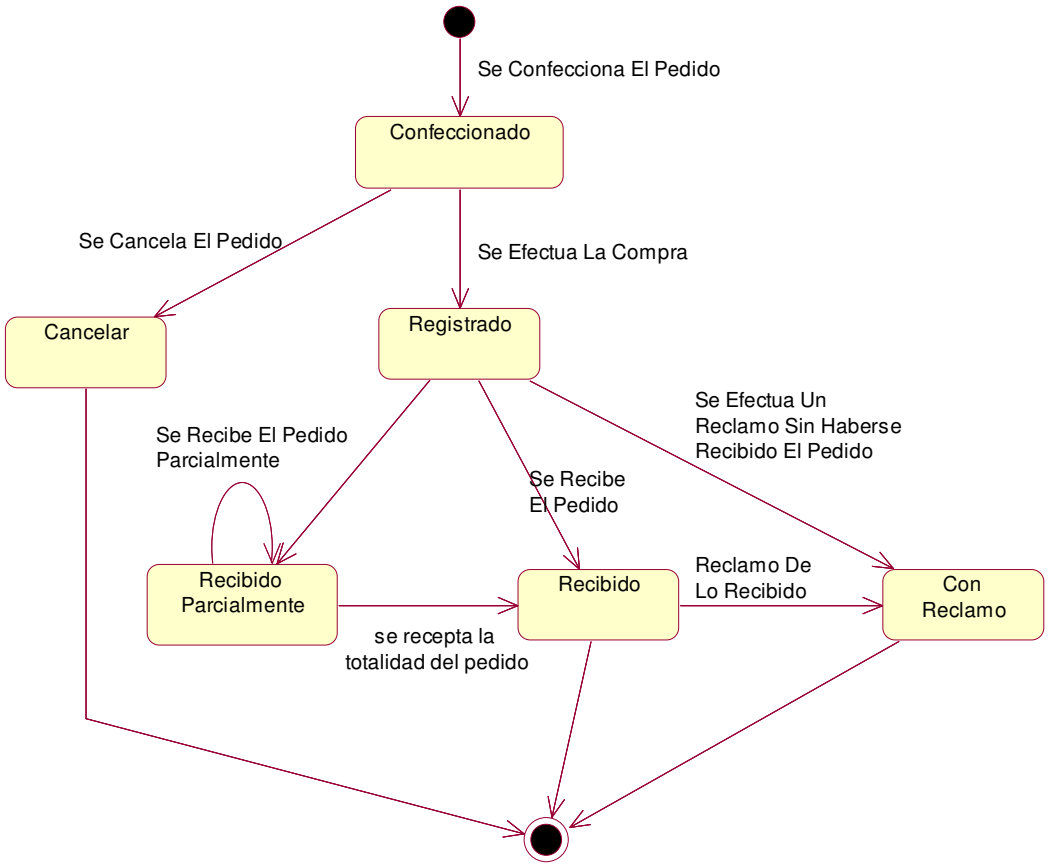
Cuenta Proveedor



Detalle Cuenta Proveedor



Pedido Proveedor


Se Confecciona El Pedido
Confeccionado
Se Cancela El Pedido
Se Efectua La Compra
Cancelar
Registrado
Se Efectua Un
Reclamo Sin Haberse
Recibido El Pedido
Se Recibe El Pedido
Parcialmente
Se Recibe
El Pedido
Recibido
Parcialmente
Recibido
Reclamo De
Lo Recibido
Con
Reclamo
se recepta la
totalidad del pedido