# Sin título.png

# 1. Introducción:

## 1.1 Propósito:

Esto es un proyecto para la asignatura Modelado del Software que nos permitirá gestionar una tienda online de informática y tendrá como audiencia al profesor Antonio Navarro.

## 1.2 Alcance:

Nuestro proyecto consistirá en la creación de una aplicación que permita gestionar una tienda online de informática. La tienda online de informática será capaz de registrar clientes, productos, marcas, proveedores, realizar ventas y devoluciones.

La aplicación proporcionará al usuario la gestión de productos, gestión de clientes, gestión de marcas, gestión de proveedores, gestión de facturas.

## 1.3 Definiciones, acrónimos y abreviaturas

ID: Identificador que se le pone a la entidad al crearla, empezando por el 1.

Base de datos: Base de datos relacional.

Carrito: Lista de productos con un cliente asignado.

ListaTareas: Lista de tareas con sus horas asignadas a un empleado.

Baja lógica: Cambia el atributo “baja” del objeto, pero no elimina el fichero.

En rojo: La clave primaria de cada entidad. Esto es, el atributo principal que no puede repetirse.

“---”: No hay nada.

## 1.4 Referencias

- Los apuntes de la asignatura.

- http://solaris.fdi.ucm.es/

- Estándar IEEE 830-1998

## 1.5 Resumen:

El resto de la SRS contiene los requisitos a nivel de usuario y de sistema según el estándar 8.30 del IEEE, así como otra información sobre el proyecto.

En la primera parte nos encargamos de describir los requisitos a nivel de usuario, luego nos metemos con los requisitos a nivel de sistema, mostramos de una matriz de traza y otra matriz de dependencia las

# 2. Descripción General:

## 2.1 Perspectiva del producto:

Este proyecto se va a usar para la asignatura de Modelado del software.

## 2.2 Funciones del producto:

**GESTOR DE CLIENTES:**

- Alta cliente: rellenar campos: Correo, nombre y apellidos, dirección, teléfono e intención de ser VIP.

- Alta cliente VIP: rellenar campos: Correo, nombre y apellidos, dirección, teléfono y límite de la tarjeta de crédito.

- Baja cliente: Buscar al cliente por ID independientemente de si es VIP o no y lo elimina (BAJA LÓGICA).

- Mostrar cliente: Busca el cliente por ID y muestra sus atributos.

- Mostrar clientes: Muestra todos los clientes.

- Modificar cliente: Solo se pueden modificar el correo, nombre, apellidos, dirección y teléfono, pero nunca el ID.

**GESTOR DE PRODUCTOS**:

- Alta producto: Nombre del producto , stock y precio del producto.

- Baja producto: Busca el producto por su ID y lo elimina (BAJA LÓGICA).

- Mostrar producto: Busca el producto por ID y muestra sus atributos.

- Mostrar productos: Muestra todos los productos.

- Modifica producto: Puede cambiar todo excepto el ID del producto.

**GESTOR DE FACTURAS:**

- Alta carrito: Añade el cliente al carrito buscándolo por ID.

- Añadir producto carrito: Añadirá el ID del producto y el número de unidades al carrito de la compra.

- Elimina producto carrito: Elimina un producto del carrito.

- Genera factura: Toda factura tendrá al menos un carrito, una fecha generará un ID de factura. Valida la factura comprobando el ID de cliente, ID de productos y el stock.

- Mostrar factura: Busca la factura por ID y la muestra.

- Mostrar facturas: Muestra todas las facturas.

- Devolución: Elimina un producto de la factura ya creada. Necesitamos ID de producto, de factura y la cantidad que queremos devolver. La cantidad que queremos devolver tiene que ser menor o igual que la vendida.

**GESTOR DE EMPLEADOS:**

- Alta empleado prácticas: rellenar campos: dni, nombre y apellidos, dirección, teléfono, intención quedarse, sueldo y departamento.

- Alta empleado contratado: rellenar campos: dni, nombre y apellidos, dirección, teléfono, sueldo, años en la empresa y departamento.

- Baja empleado: Buscar al empleado por ID independientemente de si esta de practicas o no y lo elimina (BAJA LÓGICA).

- Mostrar empleado: Busca el empleado por ID y muestra todos sus datos.

- Mostrar empleados: Muestra todos los empleados.

- Modificar empleados: Solo se pueden modificar el dni, nombre, apellidos, dirección y teléfono, sueldo, años en la empresa y la intención de quedarse, pero nunca el ID.

**GESTOR DE DEPARTAMENTOS**:

- Alta departamento: rellenar campos nombre y teléfono.

- Baja departamento: Buscar el departamento por ID y lo da de baja (BAJA LÓGICA).

- Mostrar departamento: Busca el departamento por ID y muestra todos sus datos.

- Mostrar departamentos:: Muestra todos los departamentos.

- Modificar departamento: Solo se pueden modificar el nombre, teléfono pero no el ID.

**GESTOR DE TAREAS:**

- Alta tarea: rellenar campos nombre, descripción y horas.

- Baja tarea: Buscar la tarea por ID y lo da de baja (BAJA LÓGICA).

- Mostrar tarea: Busca tarea por ID y muestra todos sus datos.

- Mostrar tareas:: Muestra todas las tareas.

- Modificar tarea: Solo se pueden modificar el nombre, la descripción y las horas, pero no el ID.

- Asignar empleado: Busca la tarea por ID y le añade un empleado.

- Desasignar empleado: Busca la tarea por ID y le quita el empleado.

## 2.3 Características del usuario:

Profesor de Modelado del Software de la Universidad Complutense de Madrid.

## 2.4 Restricciones:

Se utilizará en ordenadores en forma de aplicación de escritorio.

**2.5 Supuestos y dependencias:**

Nuestra aplicación no dependerá de otras aplicaciones.

**2.6 Requisitos futuros**

Nuestra aplicación no tendrá requisitos futuros.

# 3. Requisitos específicos

**3.1 Interfaces externas:**

**3.1.1 Interfaces de usuarios:**

Esta interfaz consta de un conjunto de ventanas y botones con las que interactúa el usuario.

**3.1.2 Interfaces de software:**

La entrada de datos sólo será válida si tienen los siguientes datos de entrada.

*Entidad Cliente*

El nombre y los apellidos serán cadenas de caracteres. El correo consistirá en una cadena de caracteres.

No pueden existir 2 clientes con el mismo correo e ID. La dirección será una cadena de caracteres.

La intención de ser VIP es un booleano.

A cada cliente creado el sistema le asignará un ID de cliente.

*Entidad Cliente VIP*

El nombre y los apellidos serán cadenas de caracteres.

El correo consistirá en una cadena de caracteres.

No pueden existir 2 clientes con el mismo correo e ID.

La dirección será una cadena de caracteres.

El límite de tarjeta será un booleano.

A cada cliente creado el sistema le asignará un ID de cliente que será un entero.

*Entidad Producto*

El nombre del producto será una cadena de caracteres. No pueden existir 2 productos con el mismo nombre e ID. La marca será una cadena de caracteres.

El proveedor será una cadena de caracteres. El precio es un número real con dos decimales. El stock es un número entero.

A cada producto creado el sistema le asignará un ID del producto que será un entero.

*Entidad Facturas:*

El carrito tendrá un ID de cliente que será un entero El carrito tendrá un ID de producto que será un entero

El carrito tendrá unas unidades de producto que será un entero La fecha será una cadena de caracteres

A cada factura creada por el sistema le asignará un ID de la factura que será un entero

*Entidad empleado prácticas:*

El nombre, apellidos, dirección, teléfono y dni del empleado serán una cadena de caracteres.

El sueldo del empleado será un float.

El ID del departamento será un entero.

El ID del empleado será un entero asignado por el sistema.

La intención de quedarse es un boolean.

No pueden existir dos empleados con el mismo ID y con el mismo dni.

La lista de tareas tendrá un ID que identifica a la tarea correspondiente y un entero que indica la duración de la tarea. También tendrá el ID del empleado asignado.

*Entidad empleado fijo:*

El nombre, apellidos, dirección, teléfono o dni del empleado serán una cadena de caracteres.

Los años en la empresa es un entero.

El ID del departamento será un entero.

El sueldo del empleado será un float.

El ID del empleado será un entero asignado por el sistema.

No pueden existir dos empleados con el mismo id o con el mismo dni.

La lista de tareas tendrá un ID que identifica a la tarea correspondiente y un entero que indica la duración de la tarea. También tendrá el ID del empleado asignado.

*Entidad departamento:*

El nombre y el teléfono serán una cadena de caracteres.

El ID del departamento será un entero asignado por el sistema.

No pueden existir dos departamentos con el mismo nombre.

*Entidad tareas:*

El nombre y la descripción serán cadena de caracteres.

No pueden existir dos tareas con el mismo ID o nombre.

El ID de la tarea es un entero asignado por el sistema.

**3.2 Requisitos a nivel de Sistema:**

*Gestor de clientes: 3.2.1*

ID: 3.2.1.1

Función: altaCliente

Prioridad: Alta

Estabilidad: Alta

Descripción: Añade un nuevo cliente a la base de datos

Entrada: Atributos de cliente excepto el ID

Salida: Nuevo cliente con un ID creado por el sistema

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ---

Postcondición: Alta del cliente con correo de entrada distinto al correo de los clientes existentes

Efectos laterales: -----

ID: 3.2.1.2

Función: bajaClientes

Prioridad: Alta

Estabilidad: Alta

Descripción: Elimina un cliente de la base de datos

Entrada: ID del cliente a eliminar

Salida: Baja lógica del cliente

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID introducido existente en ficheros de cliente

Postcondición: Eliminación lógica del cliente que coincida con el id de entrada

Efectos laterales: ---

Id: 3.2.1.3

Función: mostrarCliente

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra datos de un cliente

Entrada: ID de cliente

Salida: Atributos de cliente.

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de cliente existente

Postcondición: ---

Efectos laterales: ---

Id: 3.2.1.4

Función: mostrarClientes

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra datos de todos los clientes.

Entrada: ---

Salida: Atributos de todos los clientes

Origen: Teclado

Destino: Sistema/pantalla/interfaz

Necesita: Base de datos

Precondición: ---

Postcondición: ---

Efectos laterales: ---

id: 3.2.1.5

Función: modificarCliente

Prioridad: Alta

Estabilidad: Alta

Descripción: Modifica los atributos de un cliente excepto el ID

Entrada: ID de cliente

Salida: Cliente modificado

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de cliente existente

Postcondición: Cliente modificado y nuevo correo distinto al resto de correos existentes

Efectos laterales: ---

Id: 3.2.1.6

Función: altaClienteVIP

Prioridad: Alta

Estabilidad: Alta

Descripción: Da de alta a un cliente VIP

Entrada: Atributos de cliente VIP excepto el ID

Salida: ID de cliente

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ---

Postcondición: Alta de cliente VIP con correo de entrada distinto al correo de los clientes existentes

Efectos laterales: ---

ID: 3.2.1.7

Función: consultarClienteFactura

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra los IDs de las facturas que tiene un cliente y tienen un precio total superior al introducido

Entrada: ID de cliente y precio total de la factura

Salida: IDs de las facturas que tiene un cliente y tienen un precio total superior al introducido

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ---

Postcondición: ---

Efectos laterales: -----

*Gestor de productos: 3.2.2*

Id: 3.2.2.1

Función: altaProducto

Prioridad: Alta

Estabilidad: Alta

Descripción: Da de alta un producto

Entrada: Atributos del producto a excepción del ID

Salida: ID de producto

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición:

Postcondición: Producto dado de alta con nombre del producto distinto de nombres existentes

Efectos laterales: ---

Id: 3.2.2.2

Función: bajaProducto

Prioridad: Alta

Estabilidad: Alta

Descripción: Da de baja un producto

Entrada: ID del producto a dar de baja

Salida: Baja lógica de un producto

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de producto existente

Postcondición: Producto dado de baja (lógica)

Efectos laterales: ---

Id: 3.2.2.3

Función: mostrarProducto

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra los datos de un producto

Entrada: ID de producto

Salida: Atributos del producto

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de producto existente

Postcondición: ---

Efectos laterales: ---

Id: 3.2.2.4

Función: mostrarProductos

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra los atributos de todos los productos

Entrada: ---

Salida: Datos de todos los productos

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ---

Postcondición: ---

Efectos laterales: ---

Id: 3.2.2.5

Función: modificarProducto

Prioridad: Alta

Estabilidad: Alta

Descripción: Modifica el producto asociado a la ID (Nombre, precio, stock y si está dado de baja)

Entrada: ID de producto

Salida: Producto modificado

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de producto existente, si el producto está dado de baja no se puede modificar.

Postcondición: Producto modificado con nuevo nombre distinto a nombres existentes.

Efectos laterales: ---

ID: 3.2.2.6

Función: consultarProductoFactura

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra los IDs de las facturas en la que está el producto que tienen un precio total superior al introducido

Entrada: ID de producto y precio total de la factura

Salida: IDs de las facturas en las que está el producto y tienen un precio total superior al introducido

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición:---

Postcondición: ---

Efectos laterales: ---

*Gestor de facturas 3.2.3*

ID: 3.2.3.1

Función: altaCarrito

Prioridad: Alta

Estabilidad: Alta

Descripción: Añade el cliente al carrito

Entrada: ID del cliente

Salida: Cliente añadido al carrito

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ---

Postcondición: Cliente añadido al carrito

Efectos laterales: ---

ID: 3.2.3.2

Función: añadirProductoCarrito

Prioridad: Alta

Estabilidad: Alta

Descripción: Añade un producto al carrito

Entrada: ID del producto y unidades del producto

Salida: Producto añadido o, si ya estaba en el carrito, aumentar la cantidad

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ---

Postcondición: Producto añadido al carrito

Efectos laterales: ---

ID: 3.2.3.3

Función: eliminaProductoCarrito

Prioridad: Alta

Estabilidad: Alta

Descripción: Elimina un producto del carrito

Entrada: ID del producto

Salida: Producto eliminado

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de producto existente en el carrito

Postcondición: Producto eliminado del carrito

Efectos laterales: ---

ID: 3.2.3.4

Función: generaFactura

Prioridad: Alta

Estabilidad: Alta

Descripción: Da de alta una factura

Entrada: Carrito

Salida: ID de factura

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: Cliente existente no dado de baja, productos existentes y en stock

Postcondición: Nueva factura creada

Efectos laterales: ---

ID: 3.2.3.5

Función: mostrarFactura

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra los atributos de una factura

Entrada: ID de la factura a mostrar

Salida: Los datos de la factura

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de la factura existente

Postcondición: ---

Efectos laterales: ---

ID: 3.2.3.6

Función: mostrarFacturas

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra los atributos de las facturas

Entrada: ---

Salida: Las facturas por pantalla

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ---

Postcondición: ---

Efectos laterales: ---

ID: 3.2.3.7

Función: Devolución

Prioridad: Alta

Estabilidad: Alta

Descripción: Devuelve un producto a la tienda

Entrada: Atributos de la factura

Salida: Producto devuelto

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de factura y del producto de la factura existente y cliente dado de alta

Postcondición: Unidades del producto a devolver menores o iguales a las vendidas

Efectos laterales: ---

*Gestor de empleados: 3.2.4*

ID: 3.2.4.1

Función: altaEmpleado

Prioridad: Alta

Estabilidad: Alta

Descripción: Añade un nuevo empleado a la base de datos

Entrada: Atributos de empleado excepto el ID

Salida: Nuevo empleado con un ID creado por el sistema

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: DNI de entrada distinto al DNI de los empleados existentes

Postcondición: Alta de un nuevo empleado

Efectos laterales: -----

ID: 3.2.4.1

Función: altaEmpleadoParcial

Prioridad: Alta

Estabilidad: Alta

Descripción: Añade un nuevo empleado a tiempo parcial a la base de datos

Entrada: Atributos de empleado parcial excepto el ID

Salida: Nuevo empleado con un ID creado por el sistema

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: DNI de entrada distinto al DNI de los empleados existentes

Postcondición: Alta de un nuevo empleado a tiempo parcial

Efectos laterales: -----

ID: 3.2.4.3

Función: bajaEmpleados

Prioridad: Alta

Estabilidad: Alta

Descripción: Elimina un empleado de la base de datos

Entrada: ID del empleado a eliminar

Salida: Baja lógica del empleado

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID introducido existente en base de datos de empleado

Postcondición: Eliminación lógica del empleado que coincida con el ID del empleado introducido y desasignación de sus tareas

Efectos laterales: ---

Id: 3.2.4.4

Función: mostrarEmpleado

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra datos de un empleado

Entrada: ID de empleado

Salida: Atributos de empleado

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: DNI de empleado existente

Postcondición: ---

Efectos laterales: ---

Id: 3.2.4.5

Función: mostrarEmpleados

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra datos de todos los empleados

Entrada: ---

Salida: Atributos de todos los empleados

Origen: Teclado

Destino: Sistema/pantalla/interfaz

Necesita: Base de datos

Precondición: ---

Postcondición: ---

Efectos laterales: ---

id: 3.2.4.6

Función: modificarEmpleado

Prioridad: Alta

Estabilidad: Alta

Descripción: Modifica los atributos de un empleado.

Entrada: DNI de empleado

Salida: Empleado modificado

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: DNI de empleado existente

Postcondición: Empleado modificado.

Efectos laterales: ---

ID: 3.2.4.7

Función: consultarNominaDepartamento

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra el salario total de un departamento

Entrada: ID del departamento

Salida: Nómina de departamento mostrada

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID del departamento existente

Postcondición: ---

Efectos laterales: -----

*Gestor de Departamentos: 3.2.5*

ID: 3.2.5.1 Función: altaDepartamento

Prioridad: Alta

Estabilidad: Alta

Descripción: Añade un nuevo departamento a la base de datos

Entrada: Atributos de departamento excepto el ID

Salida: Nuevo departamento con un ID creado por el sistema

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: Nombre de entrada distinto al nombre de los departamentos existentes

Postcondición: Alta de un nuevo departamento

Efectos laterales: -----

ID: 3.2.5.2

Función: bajaDepartamento

Prioridad: Alta

Estabilidad: Alta

Descripción: Elimina un departamento de la base de datos

Entrada: ID del departamento a eliminar

Salida: Baja lógica del departamento

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: Nombre introducido existente en base de datos de departamento

Postcondición: Eliminación lógica del departamento que coincida con el nombre de entrada y que no tenga empleados activos

Efectos laterales: ---

Id: 3.2.5.3

Función: mostrarDepartamento

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra datos de un departamento

Entrada: ID de departamento

Salida: Atributos de departamento

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: Nombre de departamento existente

Postcondición: ---

Efectos laterales: ---

Id: 3.2.5.4

Función: mostrarDepartamentos

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra datos de todos los departamentos

Entrada: ---

Salida: Atributos de todos los departamentos

Origen: Teclado

Destino: Sistema/pantalla/interfaz

Necesita: Base de datos

Precondición: ---

Postcondición: ---

Efectos laterales: ---

id: 3.2.5.5

Función: modificarDepartamento

Prioridad: Alta

Estabilidad: Alta

Descripción: Modifica los datos de un departamento (Nombre, teléfono y lista de tareas)

Entrada: ID del departamento

Salida: Departamento modificado

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de departamento existente

Postcondición: Nuevo nombre de departamento distinto a uno ya existente

Efectos laterales: ---

*Gestor de Tareas : 3.2.6*

ID: 3.2.6.1

Función: altaTareas

Prioridad: Alta

Estabilidad: Alta

Descripción: Añade una nueva tarea a la base de datos

Entrada: Atributos de tarea excepto el ID

Salida: Nueva tarea con un ID creado por el sistema

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición:--

Postcondición: Alta de una nueva tarea y nombre de entrada distinto al nombre de las tareas existentes

Efectos laterales: -----

ID: 3.2.6.2

Función: bajaTarea

Prioridad: Alta

Estabilidad: Alta

Descripción: Elimina una tarea de la base de datos

Entrada: Nombre de la tarea a eliminar

Salida: Baja lógica de la tarea

Origen: Por teclado

Destino: Sistema

Necesita: Base de datos

Precondición: Nombre introducido existente en base de datos de la tarea

Postcondición: Eliminación lógica de la tarea que coincida con el nombre de entrada

Efectos laterales: ---

Id: 3.2.6.3

Función: mostrarTarea

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra datos de una tarea

Entrada: Nombre de una tarea

Salida: Atributos de una tarea

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: Nombre de una tarea existente

Postcondición: ---

Efectos laterales: ---

Id: 3.2.6.4

Función: mostrarTareas

Prioridad: Alta

Estabilidad: Alta

Descripción: Muestra datos de todas las tareas

Entrada: ---

Salida: Atributos de todas las tareas

Origen: Teclado

Destino: Sistema/pantalla/interfaz

Necesita: Base de datos

Precondición: ---

Postcondición: ---

Efectos laterales: ---

id: 3.2.6.5

Función: modificarTarea

Prioridad: Alta

Estabilidad: Alta

Descripción: Modifica los atributos de una tarea excepto el ID

Entrada: Nombre de tarea

Salida: Tarea modificada

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: Nombre de tarea existente

Postcondición: Nombre de tarea modificada distinto de los existentes

Efectos laterales: ---

ID: 3.2.6.6

Función: asignarEmpleadoTarea

Prioridad: Alta

Estabilidad: Alta

Descripción: Añade una tarea a la lista

Entrada: ID de la tarea, ID del empleado y horas para llevarla a cabo

Salida: Tarea añadida o, si ya estaba en la lista, aumentar la cantidad

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ID de tarea existente, ID del empleado existente

Postcondición: Tarea añadida a la lista. Si la tarea existe, aumentamos el número de horas

Efectos laterales: ---

ID: 3.2.3.3

Función: desasignarEmpleadoTarea

Prioridad: Alta

Estabilidad: Alta

Descripción: Elimina una tarea de la lista

Entrada: ID de la tarea

Salida: Tarea eliminada

Origen: Teclado

Destino: Sistema

Necesita: Base de datos

Precondición: ---

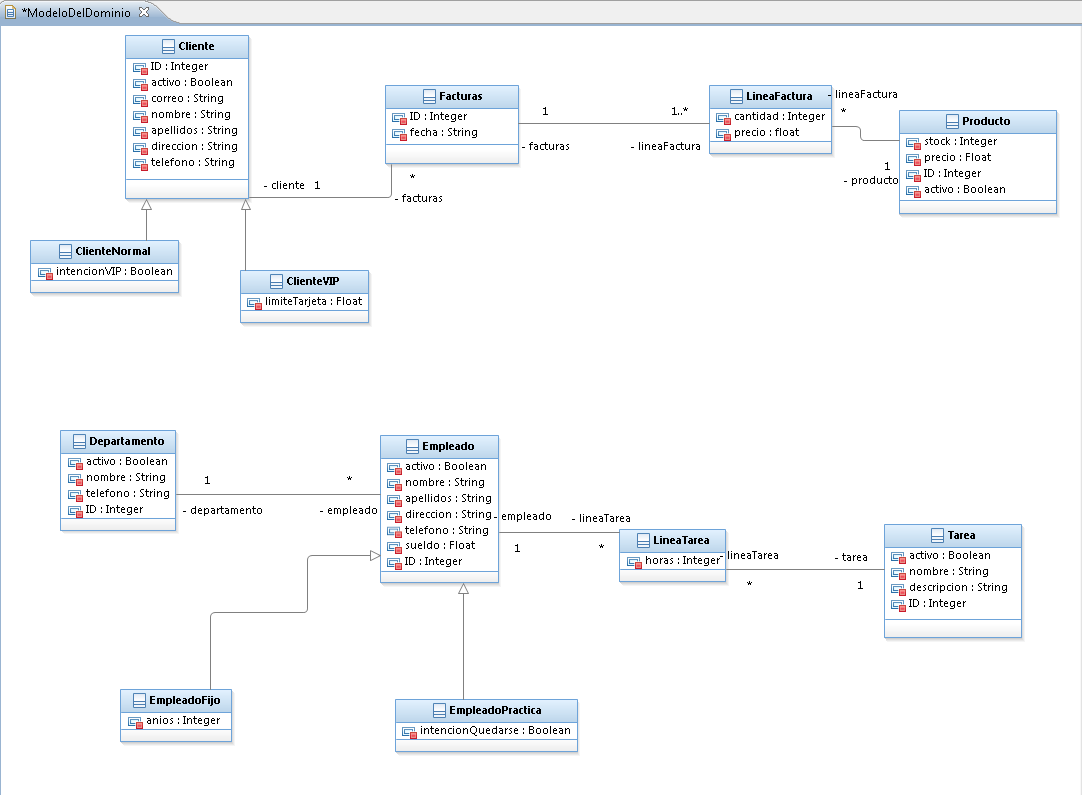
Postcondición: Tarea eliminada de la lista

Efectos laterales: ---

**3.3 Requisitos de rendimiento:**

Nuestra aplicación tendrá un único terminal y será usado por 7 usuarios, 6 que son los miembros del equipo y otro usuario que será el profesor Antonio Navarro.

**3.4 Requisitos lógicos de la base de datos**:



**3.5 Restricciones de diseño:**

El lenguaje utilizado es Java, junto con UML, tecnología multicapa y una base de datos para la persistencia de datos.

También tendremos gestión de concurrencia, gestión de la transaccionalidad y gestión de la carga dinámica.

Los patrones utilizados son: Service to worker, Transfer object, Data Access Object, Queries, Business Object, Domain Store (implementado con JPA 2.x.).

**3.6 Atributos del sistema software:**

Esta aplicación contará con RTFs para asegurar el funcionamiento correcto del proyecto. El programa será capaz de ir incluyendo actualizaciones y mejoras de software según se acuerden en el equipo.

También tendremos un control de versiones para asegurar el desarrollo del proyecto y evitar pérdida de datos.

La aplicación está desarrollada en Eclipse.

La disponibilidad de esta aplicación será de 24h/día