Departament de Llenguatges i Sistemes Informàtics

Tema 8. Model de capes

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Escola Politècnica Superior
Universitat d'Alacant

Introducció: Disseny de Components d'Accés a Dades

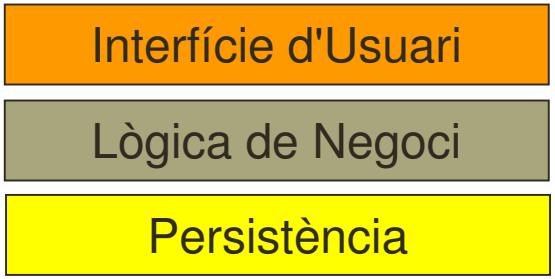
- Comencem el disseny decidint com accedir i representar les dades de negoci associats de la nostra aplicació
- Aquest tema proveeix una guia que ajuda a triar la manera més apropiada d'exposar, representar i fer persistent les dades d'una aplicació

Arquitectura de Capes

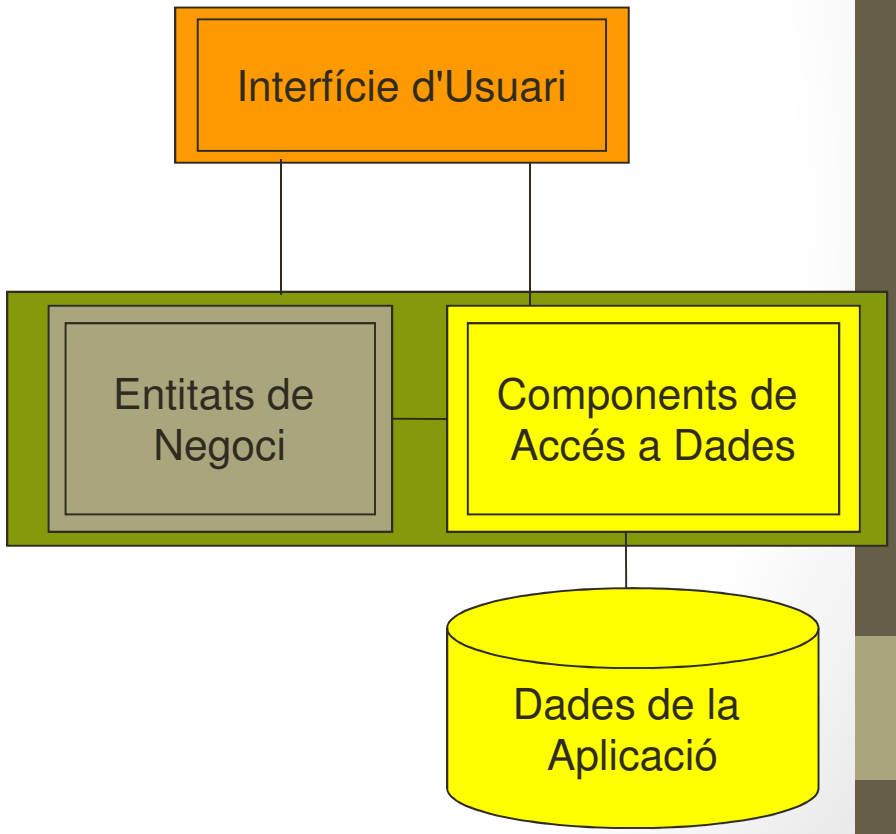
- Patró d'arquitectura [Buschmann] que estableix la distribució d'una aplicació en divisions lògiques desenvolupades i mantingudes com a mòduls independents, fins i tot en plataformes diferents.
-
- Una arquitectura de 3 capes aquesta divideixi en:
  - **Interfície d'usuari**, components que interactuen amb l'usuari final
  - **Lògica de negoci**, contenen les regles de negoci de la nostra aplicació
  - **Persistència**, conté l'accés i emmagatzematge de les dades

Arquitectura d'una Aplicació

3 Capes Lògiques



Configuració de Components



Entitats de Negoci (EN)

- Components que representen entitats de negoci del món real, p.i. un producte, una comanda
- Contenen normalment la informació d'una classe de domini amb els seus atributs, operacions i restriccions. Encara que poden representar una composició de classes
- Tenen associat un CAD (Component d'Accés a Dades) que li proporciona l'accés i el mapatge a les dades
- Poden ser representats de múltiples maneres, p.i. classes personalitzades, DataSets, XML, etc.

Representació d'una EN

```
public class ENProducto
{
// Campos privats per mantenir el
// estat de l'Entitat Producte
private int idProducto;
private String nombre;
private String cantidadPorUnidad;
private decimal precioUnitario;
private int unidadesStock;
private int stockMinimo;
// Propietats publiques per exposar el
// estat del producte
public int IdProducto
{
get { return idProducto; }
set { idProducto = value; }
}
public String Nombre {
  get { return nombre; }
  set { nombre = value; }
}
public String
CantidadPorUnidad
{
get { return
cantidadPorUnidad; }
set { cantidadPorUnidad =
value; }
}
public decimal
PrecioUnitario
{
get { return precioUnitario; }
set { precioUnitario = value;
}}
…
```

Representació d'una EN (II)

```
// Mètodes que realitzen algun processament
public void IncrementarPrecioUnidadPor (decimal cantidad)
{
      precioUnitario += cantidad;
}

public short UnidadesSobreElNivelMinimo
{
      get { return (short)(unidadesStock - stockMinimo); }
}
}//Fi de Classe
```

Components d'Accés a Dades

- Els Components d'Accés a Dades (CADs) encapsulen la tecnologia d'accés a dades i la BD a la resta de l'aplicació
- Proporciona un interfície senzill que permet recuperar les dades de la BD i salvar una entitat de negoci en la BD
- Els CADs també contenen qualsevol lògica de negoci necessària per aconseguir les operacions relacionades amb les dades

Operacions d'un CAD

- Un CAD hauria de proveir els mètodes per realitzar les següents tasques sobre la BD:
  - Crear registres en la BD
  - Llegir registres en la BD i retornar les entitats de negoci al component invocante
  - Actualitzar registres en la BD, usant entitats de negoci proporcionades pel component invocante
  - Esborrar registres de la BD

- Aquests mètodes són anomenats CRUD, acrònim de “Create, Read, Update and Delete”

Operacions d'un CAD (II)

- Els CAD poden contenir també mètodes que realitzen algun filtre. Per exemple, un CAD pot tenir un mètode per trobar el producte més venut en un catalogo durant un mes

- Un CAD accedeix a una única BD i encapsula les operacions relacionades amb una única taula o un grup de taules vinculades de la BD

- Per exemple, podreu definir un CAD que controli les taules Demanats i les LineasDePedidos

Exemple de CAD en .NET

CAD per a la classe Client

```
public class ClienteCAD
{
private String conexion;
publica ClienteCAD()
{
// Adquireix la cadena de connexió des d'un únic lloc
}
public ENCliente dameCliente (String id)
{
// Codi per recuperar un tipus DataSet contenint les dades del Client
}
public String Crear (String nombre, String direccion, String ciudad,
String pais, int codPostal){
```

Exemple de CAD (II)

```
// Codi per crear un client basat en els parametros escalessis
// Retorna l'ANEU del client en aquest mètode.
 }
 public void Actualizar (ENCliente clienteActualizado)
 {
 //Codi per actualitzar la BD, basat en el les dades del client enviats
com un paràmetre de tipus ClienteDataSet
 }
 public void Borrar (String id)
 {
 // Codi per esborrar el client amb l'ANEU especificat
 }
 public DataSet dameClientesPorCiudad (string ciudad)
 {
 // Codi per recuperar clients usant un criteri de cerca.
 }}
```

Mètode CAD: BorrarCliente

```
// Mètode per recuperar el Nom del Client
public void BorrarCliente( String clienteID )
{
SqlConnection conn = null;
// Encapsula tot l'accés a dades dins del try
String comando = “Delete from Cliente where id = “+ clienteID;
try
{
conn = new SqlConnection(conexion);
conn.Open();
SqlCommand cmd = new SqlCommand(comando, conn );
```

Método CAD: BorrarCliente (II)

```
 cmd.ExecuteNonQuery();
}
catch (SqlException sqlex)
{
// Embolica l'excepció actual en una excepció mes rellevant
throw new CADException (“Error borrando el cliente: " + clienteID, sqlex );
}
catch (Exception ex)
{
// Captura la condició general i la reenvía.
throw ex;
}
finally
{
  if(conn != null) conn.Close(); // S'assegura de tancar la connexió.
}}
```

Mètode CAD: ObtenerClientesPorCiudad

```
// Métot per recuperar els clients d'una determinada ciutat
public DataSet ObtenerClientesPorCiudad( String ciudad )
{
SqlConnection conn = null;
DataSet dsClientes = null;
// Encapsula tot l'accés a dades dins del try
string comando = “Select * from Cliente where ciudad = “+ ciudad;
try
{
  conn = new SqlConnection(conexion);
  SqlDataAdapter sqlAdaptador = new SqlDataAdapter (comando, conn);
```

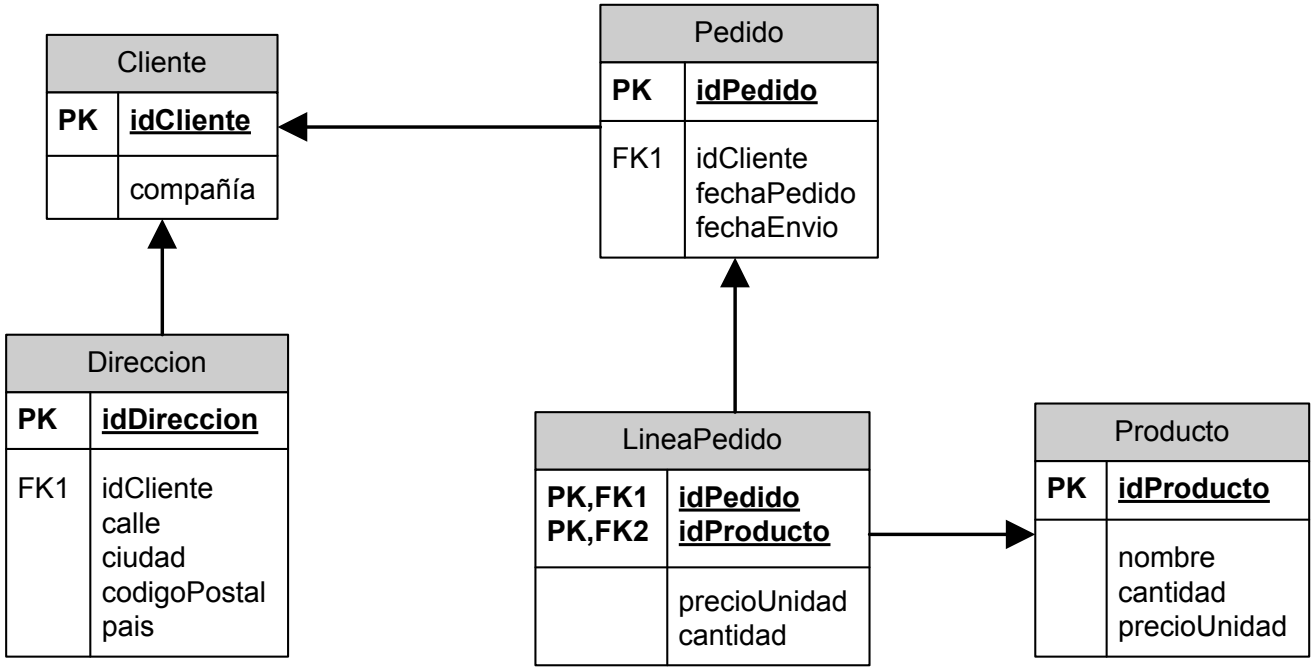
Mètode CAD: ObtenerClientesPorCiudad

```
 dsClientes = new DataSet();
 sqlAdaptador.Fill (dsClientes);
 return dsClientes;
}
catch (SqlException sqlex)
{
  throw new CADException (“Error en la consulta de clients per ciutat: " +
clienteID, sqlex );
}
catch (Exception ex)
{
// Captura la condició general i la reenvía.
throw ex;
}
finally
{
if(conn != null) conn.Close(); // S'assegura de tancar la connexió.
}}
```

De Relacional a Entitat de Negoci

- Una BD conté múltiples taules amb relacions i hem de decidir com mapear les taules en diferents EN
- Quan es defineix les EN s'ha de considerar “com” s'usarà la informació en l'aplicació
- És millor identificar el nucli d'En què encapsulen la funcionalitat de l'aplicació, abans que definir una EN per cada taula

De Relacional a Entitat de Negoci



Base de dades reduïda d'una aplicació d'una Tenda de Venda al detall

De Relacional a Entitat de Negoci

- Les requisits funcionals mínims d'una tenda són:
  - Obtenir informació sobre el Client, incloent les seves adreces
  - Obtenir la llista de comandes per a un client
  - Obtenir la llista d'articles per a una comanda en particular
  - Enviar una nova comanda
  - Obtenir o actualitzar la informació d'un producte o col·lecció de productes

De Relacional a Entitat de Negoci

- Per completar aquests requisits, podem fer-ho definint tres EN lògiques que controlin l'aplicació:
  - Un Client que contindrà les seves adreces
  - Una Comanda que contindrà les seves línies de comanda
  - I un Producte

De Relacional a Entitat de Negoci

- Para cada EN, definim un CAD que serà definit com segueix:
  - ClienteCAD: Aquesta classe proveeix els serveis per recuperar i modificar les dades de les taules Client i Adreça
  - PedidoCAD: Aquesta classe proveeix els serveis per recuperar i modificar les dades de les taules Demanat i LineaPedido
  - ProductoCAD: Aquesta classe proveeix els serveis per recuperar i modificar les dades de la taula Producte

De Relacional a Entitat de Negoci: Recomanaci

- Pren-te el teu temps per analitzar i modelar les EN de la teva aplicació, en lloc de definir una EN per cada taula
- Basa't en les composicions i herències UML per compondre objectes complexos
- No defineixis EN separades per representar taules molts-a-molts. Aquestes relacions poden ser implementades mitjançant col·leccions en les EN implicades

De Relacional a Entitat de Negoci: Recomanaci

- Definir tots els mètodes que retornen un tipus concret d'Entitat de Negoci en un sol CAD
  - Per exemple, si s'estan recuperant totes les comandes d'un determinat client, implementar la funció en PedidoCAD cridada ***ObtenerPedidosPorCliente*** que retorni les comandes filtrant per un idCliente
  - Contràriament, si estàs recuperant tots els clients que han demanat un específic producte, implementa la funció en ClienteCAD ***ObtenerClientesPorProducto***

Altres tasques que pot realitzar un CAD

- Els CADs poden realitzar altres tasques en la seva implementació:
  - Controlar la seguretat i autorització
  - Realitzar la paginació de dades
  - Realitzar Transaccions d'entitats complexes
  - Invocar a procediments emmagatzemats