Departamento de Lenguajes y Sistemas Informáticos

Tema 8. Modelo de capas

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Escuela Politécnica Superior
Universidad de Alicante

Introducción: Diseño de Componentes de Acceso a Datos

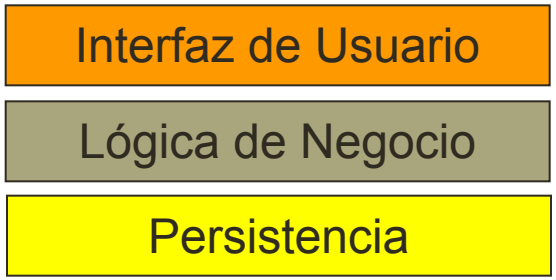
- Comenzamos el diseño decidiendo como acceder y representar los datos de negocio asociados de nuestra aplicación
- Este tema provee una guía que ayuda a elegir el modo más apropiado de exponer, representar y hacer persistente los datos de una aplicación

Arquitectura de Capas

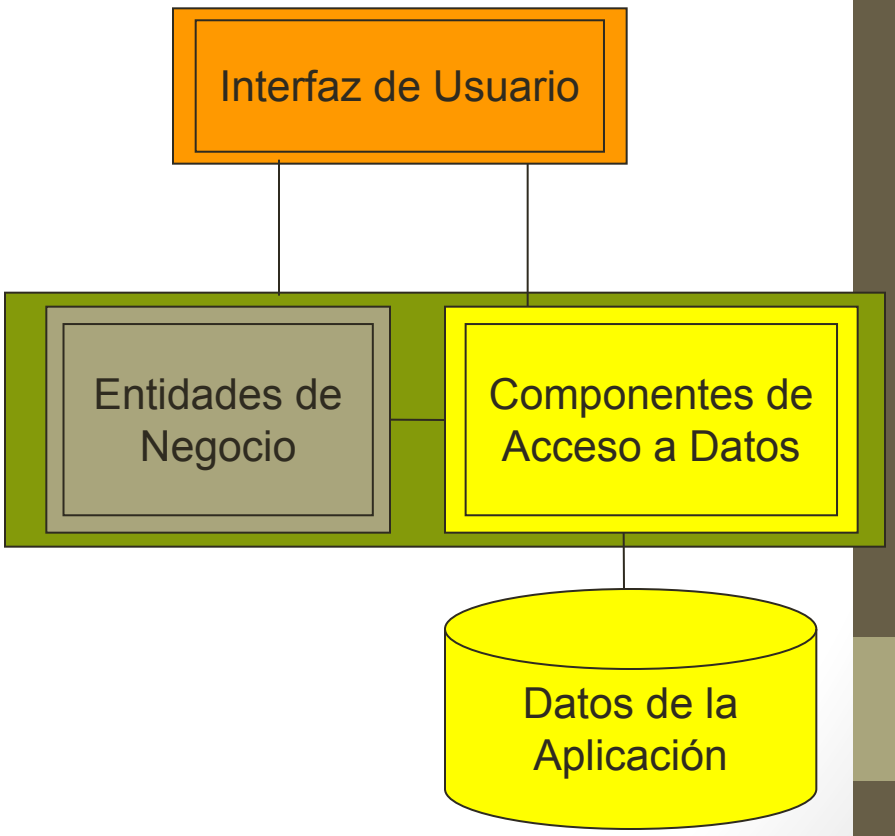
- Patrón de arquitectura [Buschmann] que establece la distribución de una aplicación en divisiones lógicas desarrolladas y mantenidas como módulos independientes, incluso en plataformas distintas.
-
- Una arquitectura de 3 capas esta divida en:
  - **Interfaz de usuario**, componentes que interactúan con el usuario final
  - **Lógica de negocio**, contienen las reglas de negocio de nuestra aplicación
  - **Persistencia**, contiene el acceso y almacenamiento de los datos

Arquitectura de una Aplicación

3 Capas Lógicas



Configuración de Componentes



Entidades de Negocio (EN)

- Componentes que representan entidades de negocio del mundo real, p.e. un producto, un pedido
- Contienen normalmente la información de una clase de dominio con sus atributos, operaciones y restricciones. Aunque pueden representar una composición de clases
- Tienen asociado un CAD (Componente de Acceso a Datos) que le proporciona el acceso y el mapeo a los datos
- Pueden ser representados de múltiples maneras, p.e. clases personalizadas, DataSets, XML, etc.

Representación de una EN

```
public class ENProducto
{
// Campos privados para mantener el
// estado de la Entidad Producto
private int idProducto;
private String nombre;
private String cantidadPorUnidad;
private decimal precioUnitario;
private int unidadesStock;
private int stockMinimo;
// Propiedades publicas para exponer el
// estado del producto
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;
}}
…
```

Representación de una EN (II)

```
// Métodos que realizan algún procesamiento
public void IncrementarPrecioUnidadPor (decimal cantidad)
{
    precioUnitario += cantidad;
}

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

Componentes de Acceso a Datos

- Los Componentes de Acceso a Datos (CADs) encapsulan la tecnología de acceso a datos y la BD al resto de la aplicación
- Proporciona un interfaz sencillo que permite recuperar los datos de la BD y salvar una entidad de negocio en la BD
- Los CADs también contienen cualquier lógica de negocio necesaria para alcanzar las operaciones relacionadas con los datos

Operaciones de un CAD

- Un CAD debería proveer los métodos para realizar las siguientes tareas sobre la BD:
  - Crear registros en la BD
  - Leer registros en la BD y devolver las entidades de negocio al componente invocante
  - Actualizar registros en la BD, usando entidades de negocio proporcionadas por el componente invocante
  - Borrar registros de la BD

- Estos métodos son llamados CRUD, acrónimo de “Create, Read, Update and Delete”

Operaciones de un CAD (II)

- Los CAD pueden contener también métodos que realizan algún filtro. Por ejemplo, un CAD puede tener un método para encontrar el producto más vendido en un catalogo durante un mes
- Un CAD accede a una única BD y encapsula las operaciones relacionadas con una única tabla o un grupo de tablas vinculadas de la BD
- Por ejemplo, podréis definir un CAD que controle las tablas Pedidos y las LineasDePedidos

Ejemplo de CAD en .NET

CAD para la clase Cliente

```
public class ClienteCAD
{
private String conexion;
publica ClienteCAD()
{
// Adquiere la cadena de conexión desde un único sitio
}
public ENCliente dameCliente (String id)
{
// Código para recuperar un tipo DataSet conteniendo los datos del Cliente
}
public String Crear (String nombre, String direccion, String ciudad, String pais, int codPostal){
```

Ejemplo de CAD (II)

```
// Código para crear un cliente basado en los parametros escalares
// Devuelve el ID del cliente en este método.
 }
 public void Actualizar (ENCliente clienteActualizado)
 {
 //Código para actualizar la BD, basado en el los datos del cliente
enviados como un parámetro de tipo ClienteDataSet
 }
 public void Borrar (String id)
 {
 // Código para borrar el cliente con el ID especificado
 }
 public DataSet dameClientesPorCiudad (string ciudad)
 {
 // Código para recuperar clientes usando un criterio de búsqueda.
 }}
```

Método CAD: BorrarCliente

```
// Método para recuperar el Nombre del Cliente
public void BorrarCliente( String clienteID )
{
SqlConnection conn = null;
// Encapsula todo el acceso a datos dentro 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)
{
// Envuelve la excepción actual en una excepción mas relevante
throw new CADException (“Error borrando el cliente: " + clienteID, sqlex );
}
catch (Exception ex)
{
// Captura la condición general y la reenvía.
throw ex;
}
finally
{
  if(conn != null) conn.Close(); // Se asegura de cerrar la conexión.
}}
```

Método CAD: ObtenerClientesPorCiudad

```
// Método para recuperar los clientes de una determinada ciudad
public DataSet ObtenerClientesPorCiudad( String ciudad )
{
SqlConnection conn = null;
DataSet dsClientes = null;
// Encapsula todo el acceso a datos dentro del try
string comando = “Select * from Cliente where ciudad = “+ ciudad;
try
{
  conn = new SqlConnection(conexion);
  SqlDataAdapter sqlAdaptador = new SqlDataAdapter (comando, conn);
```

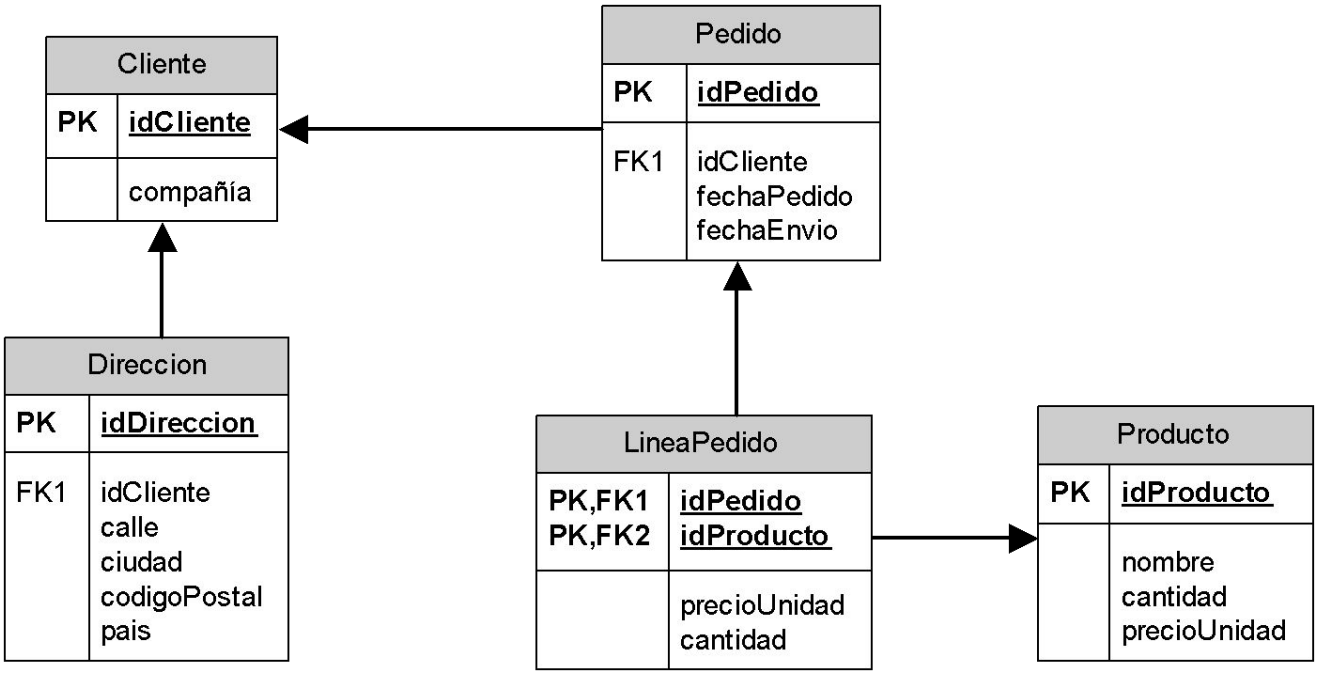
Método CAD: ObtenerClientesPorCiudad

```
 dsClientes = new DataSet();
 sqlAdaptador.Fill (dsClientes);
 return dsClientes;
}
catch (SqlException sqlex)
{
  throw new CADException (“Error en la consulta de clientes por ciudad: " +
clienteID, sqlex );
}
catch (Exception ex)
{
// Captura la condición general y la reenvía.
throw ex;
}
finally
{
if(conn != null) conn.Close(); // Se asegura de cerrar la conexión.
}}
```

De Relacional a Entidad de Negocio

- Una BD contiene múltiples tablas con relaciones y debemos decidir como mapear las tablas en diferentes EN
- Cuando se define las EN se debe considerar “como” se usará la información en la aplicación
- Es mejor identificar el núcleo de EN que encapsulan la funcionalidad de la aplicación, antes que definir una EN por cada tabla

De Relacional a Entidad de Negocio



Base de Datos reducida de una aplicación de una Tienda de Venta al por menor

De Relacional a Entidad de Negocio

- Las requisitos funcionales mínimos de una tienda son:
  - Obtener información sobre el Cliente, incluyendo sus direcciones
  - Obtener la lista de pedidos para un cliente
  - Obtener la lista de artículos para un pedido en particular
  - Enviar un nuevo pedido
  - Obtener o actualizar la información de un producto o colección de productos

De Relacional a Entidad de Negocio

- Para completar estos requisitos, podemos hacerlo definiendo tres EN lógicas que controlen la aplicación:
  - Un Cliente que contendrá sus direcciones
  - Un Pedido que contendrá sus líneas de pedido
  - Y un Producto

De Relacional a Entidad de Negocio

- Para cada EN, definimos un CAD que será definido como sigue:
  - ClienteCAD: Esta clase provee los servicios para recuperar y modificar los datos de las tablas Cliente y Dirección
  - PedidoCAD: Esta clase provee los servicios para recuperar y modificar los datos de las tablas Pedido y LineaPedido
  - ProductoCAD: Esta clase provee los servicios para recuperar y modificar los datos de la tabla Producto

De Relacional a Entidad de Negocio: Recomendaciones

- Tómate tu tiempo para analizar y modelar las EN de tu aplicación, en lugar de definir una EN por cada tabla
- Básate en las composiciones y herencias UML para componer objetos complejos
- No definas EN separadas para representar tablas muchos-a-muchos. Estas relaciones pueden ser implementadas mediante colecciones en las EN implicadas

De Relacional a Entidad de Negocio: Recomendaciones

- Definir todos los métodos que devuelven un tipo concreto de Entidad de Negocio en un solo CAD
  - Por ejemplo, si se están recuperando todos los pedidos de un determinado cliente, implementar la función en PedidoCAD llamada ***ObtenerPedidosPorCliente*** que devuelva los pedidos filtrando por un idCliente
  - Contrariamente, si estás recuperando todos los clientes que han pedido un específico producto, implementa la función en ClienteCAD ***ObtenerClientesPorProducto***

Otras tareas que puede realizar un CAD

- Los CADs pueden realizar otras tareas en su implementación:
  - Controlar la seguridad y autorización
  - Realizar la paginación de datos
  - Realizar Transacciones de entidades complejas
  - Invocar a procedimientos almacenados

Distribución capas en el proyecto

En la práctica en grupo…

- Una solución con 2 proyectos:
  - Proyecto Web (con referencia al proyecto de librería)
    - Contendrá la interfaz y validaciones
  - Proyecto de Librería de clases con las Capas EN y CAD
    - Carpeta EN
      - Clases EN para entidades del sistema
    - Carpeta CAD
      - Por cada EN una clase CAD para el acceso a datos