Método `Listar()` – Explicación paso a paso

Este documento explica en detalle qué hace el método `Listar()` en C#, que se utiliza para obtener desde la base de datos una lista de productos con su categoría.

---

1. Código completo del método

```
// Metodo que permite listar todos los productos desde la base de datos
public List<Producto> Listar()
{
    // Genera la variable de tipo Lista que contiene Productos
    List<Producto> lista = new List<Producto>();

    // Conecta a la base por medio de la "cadena de conexion"
    using (SqlConnection oconexion = new SqlConnection(Conexion.cadena))
    {
        // Capturador de errores por si falla la conexion a la BD
        try
        {
            StringBuilder query = new StringBuilder();

            // Traer la lista de productos con sus categorías (todos, sin
filtro).
            query.AppendLine("select
IdProducto,Codigo,Nombre,p.Descripcion,c.IdCategoria,c.Descripcion[DescripcionCategoria],p.Sto
from PRODUCTO p");

            // Consulta a la BD la tabla Producto y Categoria, relacionando
su IdCategoria
            query.AppendLine("inner join CATEGORIA c on c.IdCategoria =
p.IdCategoria");

            SqlCommand cmd = new SqlCommand(query.ToString(), oconexion);
            cmd.CommandType = CommandType.Text;

            oconexion.Open();

            // Lee el resultado del comando anterior
            using (SqlDataReader dr = cmd.ExecuteReader())
            {
                while (dr.Read())
                {
                    // crea un nuevo objeto de la clase Producto y lo agrega
a la lista
                    lista.Add(new Producto()
                    {
                        IdProducto = Convert.ToInt32(dr["IdProducto"]),
```

```
                        Codigo = dr["Codigo"].ToString(),
                        Nombre = dr["Nombre"].ToString(),
                        Descripcion = dr["Descripcion"].ToString(),
                        oCategoria = new Categoria()
                        {
                            IdCategoria = Convert.ToInt32(dr["IdCategoria"]),
                            Descripcion =
dr["DescripcionCategoria"].ToString()
                        },
                        Stock = Convert.ToInt32(dr["Stock"].ToString()),
                        Precio = Convert.ToDecimal(dr["Precio"].ToString()),
                        Estado = Convert.ToBoolean(dr["Estado"])
                    });
                }
            }
        }
        // Si se produce una excepción se crea una nueva lista de productos
vacía
        catch (Exception ex)
        {
            lista = new List<Producto>();
            // Idealmente acá se podría registrar el error: ex.Message
        }
    }

    return lista;
}
```

---

2. Idea general del método

El método `Listar()` se encarga de:

1. Conectarse a la base de datos.
2. Ejecutar una consulta `SELECT` que devuelve **todos los productos** con su categoría.
3. Recorrer los resultados.
4. Crear objetos `Producto` (con su objeto `Categoria` interno).
5. Agregar esos objetos a una lista.
6. Devolver la lista completa.

Es decir, es un método para **listar productos** desde la base de datos.

---

3. Creación de la lista de productos

```
List<Producto> lista = new List<Producto>();
```

- Se crea una lista vacía de tipo `Producto`.

- Cada fila que venga de la base de datos se transformará en un objeto `Producto` y se agregará a esta lista.
- Al final, el método devolverá esta lista con todos los productos.

---

4. Conexión a la base de datos

```
using (SqlConnection oconexion = new SqlConnection(Conexion.cadena))
{
    ...
}
```

- `SqlConnection` representa la conexión a SQL Server.
- `Conexion.cadena` contiene el *connection string* (servidor, base de datos, usuario, etc.).
- El `using` garantiza que la conexión se cierre y libere los recursos, aunque ocurra un error.

---

5. Manejo de errores con `try/catch`

```
try
{
    // Código que accede a la BD
}
catch (Exception ex)
{
    lista = new List<Producto>();
}
```

- Todo el acceso a la base de datos está dentro del `try`.
- Si ocurre cualquier error (fallo de conexión, error en el SQL, etc.), se captura en el `catch`.
- En el `catch` se reemplaza la lista por una lista vacía, de modo que el método siempre devuelva un valor válido.
- En una aplicación real, también se suele registrar el error (`ex.Message`).

---

6. Armado de la consulta SQL

```
StringBuilder query = new StringBuilder();

query.AppendLine("select
IdProducto,Codigo,Nombre,p.Descripcion,c.IdCategoria,c.Descripcion[DescripcionCategoria],p.Sto
from PRODUCTO p");
query.AppendLine("inner join CATEGORIA c on c.IdCategoria = p.IdCategoria");
```

Se usa un `StringBuilder` para construir el texto de la consulta SQL.

La consulta resultante es:

```
SELECT
    IdProducto,
    Codigo,
    Nombre,
    p.Descripcion,
    c.IdCategoria,
    c.Descripcion AS DescripcionCategoria,
    p.Stock,
    p.Precio,
    p.Estado
FROM PRODUCTO p
INNER JOIN CATEGORIA c ON c.IdCategoria = p.IdCategoria;
```

¿Qué hace esta consulta?

- Trae todos los registros de la tabla `PRODUCTO` (`p`).
- Hace un `INNER JOIN` con la tabla `CATEGORIA` (`c`) usando la columna `IdCategoria`.
- Devuelve, para cada producto:
- Sus datos (IdProducto, Código, Nombre, Descripción, Stock, Precio, Estado).
- Los datos de su categoría (IdCategoria y la descripción de la categoría).

Como no tiene cláusula `WHERE`, **lista todos los productos**.

---

7. Creación del `SqlCommand`

```
SqlCommand cmd = new SqlCommand(query.ToString(), oconexion);
cmd.CommandType = CommandType.Text;
```

- `query.ToString()` convierte el `StringBuilder` en un `string` con la consulta SQL completa.
- `oconexion` indica qué conexión a la base se va a usar para ejecutar este comando.
- `CommandType.Text` indica que el comando es **texto SQL directo** (no un stored procedure).

En otras palabras: se construye un comando que dice: *"ejecutá este SELECT en esta conexión"*.

---

8. Apertura de la conexión

```
oconexion.Open();
```

- Abre la conexión con SQL Server.
- A partir de este momento se puede ejecutar el comando (`cmd`).

---

9. Ejecución del comando y lectura de datos

```
using (SqlDataReader dr = cmd.ExecuteReader())
{
    while (dr.Read())
    {
        lista.Add(new Producto()
        {
            IdProducto = Convert.ToInt32(dr["IdProducto"]),
            Codigo = dr["Codigo"].ToString(),
            Nombre = dr["Nombre"].ToString(),
            Descripcion = dr["Descripcion"].ToString(),
            oCategoria = new Categoria()
            {
                IdCategoria = Convert.ToInt32(dr["IdCategoria"]),
                Descripcion = dr["DescripcionCategoria"].ToString()
            },
            Stock = Convert.ToInt32(dr["Stock"].ToString()),
            Precio = Convert.ToDecimal(dr["Precio"].ToString()),
            Estado = Convert.ToBoolean(dr["Estado"])
        });
    }
}
```

Explicación paso a paso

1. `cmd.ExecuteReader()` ejecuta la consulta `SELECT` y devuelve un `SqlDataReader` (`dr`).
2. `while (dr.Read())` va leyendo una fila por vez de los resultados.
3. En cada vuelta del `while` se crea un nuevo objeto `Producto`:
4. Se leen los valores de las columnas usando `dr["NombreColumna"]`.
5. Se convierten al tipo de dato correcto (`int`, `decimal`, `bool`, `string`).
6. Se crea un objeto `Categoria` (`oCategoria`) con:
    - `IdCategoria` → tomado de `dr["IdCategoria"]`.
    - `Descripcion` → tomado de `dr["DescripcionCategoria"]`.
7. El nuevo `Producto` se agrega a la lista:

```
lista.Add(new Producto() { ... });
```

Resultado: cada fila del resultado SQL se transforma en un objeto `Producto` en memoria.

---

10. Devolución del resultado

```
return lista;
```

- Si todo salió bien, `lista` contiene todos los productos.

- Si ocurrió algún error en el acceso a la base de datos, `lista` será una lista vacía (por el `catch`).

---

11. Resumen final

- **Propósito del método**: listar todos los productos con su categoría desde la base de datos.
- **Tecnologías usadas**:
- C# y ADO.NET (`SqlConnection`, `SqlCommand`, `SqlDataReader`).
- SQL Server (`SELECT`, `INNER JOIN`).
- **Flujo**:
- Crear lista de productos.
- Abrir conexión a la base.
- Armar consulta SQL con `INNER JOIN`.
- Crear `SqlCommand` y ejecutar `ExecuteReader()`.
- Recorrer resultados con `SqlDataReader`.
- Mapear filas a objetos `Producto` (con `Categoria`).
- Devolver la lista.

Este patrón de código es muy común cuando trabajás con ADO.NET de forma “manual” para acceder a una base de datos relacional desde C#.