**Capítulo I**

Nota al pie. El muchacho lo hace con visual basic.

Usa PictureBox con autosize como los botones para minimizar y cerrar el formulario. Yo uso labels ya que son así: X y \_

Con LineShape introduce una barra horizontal (en este caso). Se pueden manejar a precisión las coordenadas.

Linklabel es una etiqueta que aparece así como los hipervínculos que todos queremos.

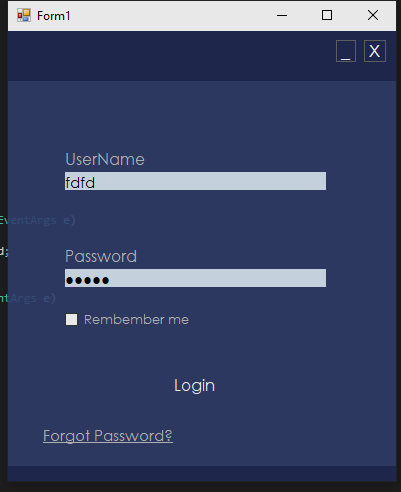
La propiedad StartPosition dice donde se va a inicializar el form en relación a la pantalla.

Opacidad

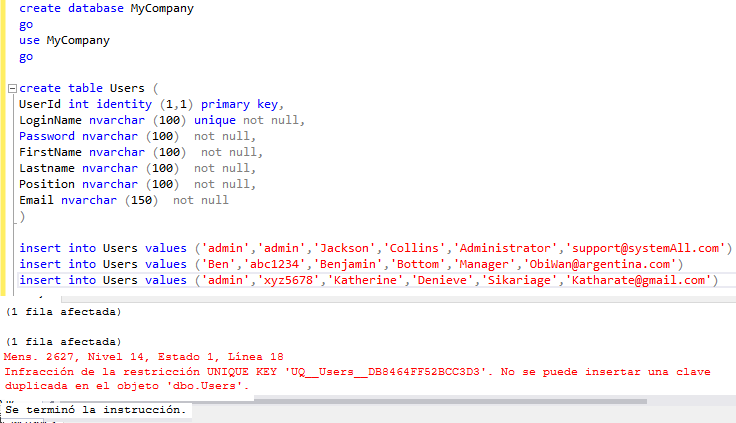
this.Opacity = 0.9;

Modo contraseña

txtPassword.UseSystemPasswordChar = true;



**Capítulo 2**



Cree esa table. Se ve que a LoginName le mete unique. El cartelito rojo del printeo señala que ‘admin’ no puede estar repetido.

Afirma que lo ideal es tener los datos de la conexión a BBDD en app.config

Hacer un bloque using haría innecesario cerrar la conexión, así como limpiar los parámetro del comando.

Hasta ahora se venía trabajando con un repositorio distinto al del Cap I. Ahora se lo incorpora, clickeando en el proyecto de presentación. Creo que lo que se importa es “Form1.Cs”. El diseñador y los recursos se agregan automáticamente.

MessageBox Yes/No

private void label11\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Are you sure you want to close the aplicattion", "Warning",

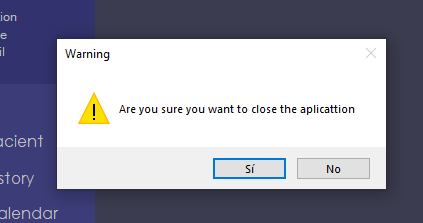
MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)

{

this.Close();

}

}





Tras explicar esto. Crea un 4 proyecto: Common y ahí crea una carpeta “Cache”. La hace estática

Aclara que todo esto puede hacerse en la capa de dominio.

Sus propiedades las hace públicas.

Todas las capas van a agregarla como referencia.

namespace Common

{

public static class ActiveUser

{

public static int idUser;

public static string firstName;

public static string email;

public static string lastName;

public static string position;

}

}

Así queda de momento. Eso se llena cuando se lee la BBDD:

public bool Login (string user, string pass)

{

using (SqlConnection sqlconexion = GetSqlConnection())

{

sqlconexion.Open();

SqlCommand comando = new SqlCommand();

comando.Connection = GetSqlConnection();

comando.CommandText = "select \* from Users where LoginName=@user and Password=@pass";

comando.Parameters.AddWithValue("@user", user);

comando.Parameters.AddWithValue("@pass", pass);

comando.CommandType = CommandType.Text;

SqlDataReader sqlDataReader = comando.ExecuteReader();

comando.Parameters.Clear();

if (sqlDataReader.Read())

{

ActiveUser.idUser = sqlDataReader.GetInt32(0);

//Ese número al final hace referencia al número de columna de la tabla

ActiveUser.firstName = sqlDataReader.GetString(3);

ActiveUser.lastName = sqlDataReader.GetString(4);

ActiveUser.position = sqlDataReader.GetString(5);

ActiveUser.email = sqlDataReader.GetString(6);

return true;

}

else return false;

}

}

FormBorderStyle Cambiando eso, le sacas el borde al Formulario.

Sacando eso no podría moverse el form. Para eso hay que agregar algo “tocable” que se pueda usar para eso.

Se utiliza el evento MouseDown, MouseMove y MouseUp.

// bool mouseDown = false;

int m, mx, my;

private void panel1\_MouseUp(object sender, MouseEventArgs e)

{

// mouseDown = false;

m = 0;

}

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

/\* if (mouseDown == true)

{

this.Location = Cursor.Position;

} \*/

if (m == 1)

{

this.SetDesktopLocation(MousePosition.X - mx, MousePosition.Y - my);

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

// mouseDown = true;

m = 1;

mx = e.X;

my = e.Y;

}

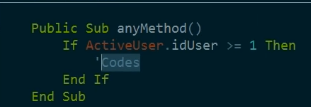
}

En verde la forma más fácil pero menos bella. Lista para correr la buena aunque mas complicada.

Dice que es buena práctica hacer una tabla relacionada, en la que figuren los cargos, y no que esté todo en la misma.

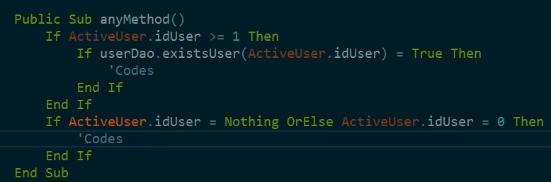
Seguridad

La Common es la mas importante para la seguridad.



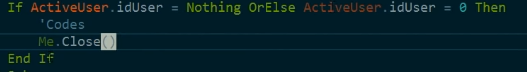
Esto es casi una introducción.

Crea un método cualquiera, el cual ejecuta su código si es que hay cargado algún Id de usuario, lo cual significa que en la capa de presentación se tuvo que loguear con username y password.



Acá suma 2 cosas mas. Anida otro if, el cual busca en la BBDD si existe dicho número de usuario.

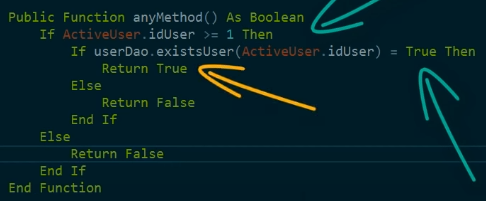
Por otro lado, si el idUser es 0, es decir no cargó nada, se podría enviar un mail al administrador.



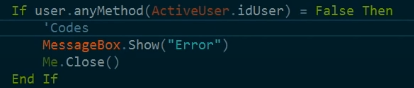
Pone eso es el form de Main Menu. Creo que es igual a lo primero. Si no hay idUser, te cierra el form.

Llama a estos métodos los mas “básicos e importante” de las páginas web.

¿Qué pasa si el hacker puede tocar el caché?

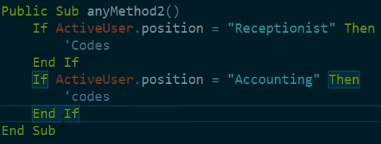


Acá se fija si el Id introducido por el hacker en cache existe en la BBDD o no, si existe retorna true(Todo esto en dominio).



Luego se llama a ese método desde la capa de presentación.

Siguiendo en el dominio, aunque es algo distinto. Es definir el acceso en función del cargo ocupado:



Dice que hacerlo de esta manera puede generar errores y además, si que quiere modificar algo es muy trabajoso.

Por eso apela a la capa common y crea ahí un struct para todos los cargos:

namespace Common.Cache

{

public struct Positions {

public const string administrator= "Administrator";

public const string receptionist= "Receptionist";

public const string accounting = "Accounting";

}

}

1 Prueba de seguridad

Modifico el orden en que se ejecuta la app, para que saltee la carga de usuario y contraseña. Tal cual podría hacer un hacker.

static class Program

{

/// <summary>

/// Punto de entrada principal para la aplicación.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

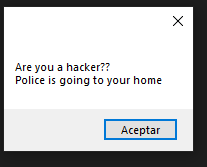
Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new FormMainMenu());

//Application.Run(new Form1());

}

}



Se abre esa ventanita, tras lo cual se cierra la app. Eso es porque en el form load del main menú agregué este código:

if (ActiveUser.idUser.ToString() == null || ActiveUser.idUser <= 0)

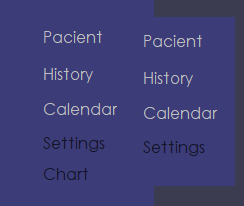
{

MessageBox.Show("Are you a hacker??" + Environment.NewLine + "Police is going to your home") ;

this.Close();

}

Permisos de seguridad



Así se la puede hacer caber a una recepcionista:

En el caso de la izquierda toqué el enabled, en el de la derecha toqué un enabled y un visible.

private void ManagePermissions()

{

if (ActiveUser.position == Positions.receptionist)

{

lblChart.Visible = false;

lblSettings.Enabled = false;

//Ahí le quita acceso a alguien logueado como recepcionista a ver el chart.

}

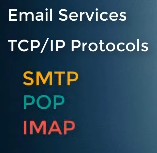
}

Este método hay que llamarlo en formLoad.

Dice que si la empresa es muy grande, pueden crearse formularios distintos (Main Menu dice) para cada una de las áreas. Cada área tendría, quizás varias posiciones distintas.

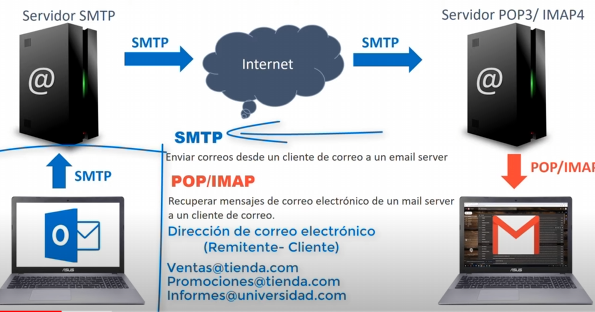
También señala la posibilidad de que cada área tengas sus propias capas. Habla de microservicios y rápidamente se las toma, el muy cagón del orto.

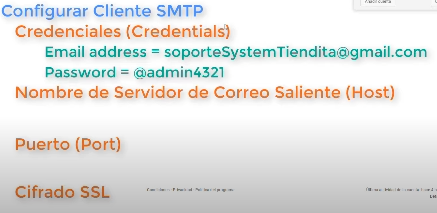
**Capítulo 4**



El que interesa acá es el SMTP. Se envía correo desde la app al usuario.

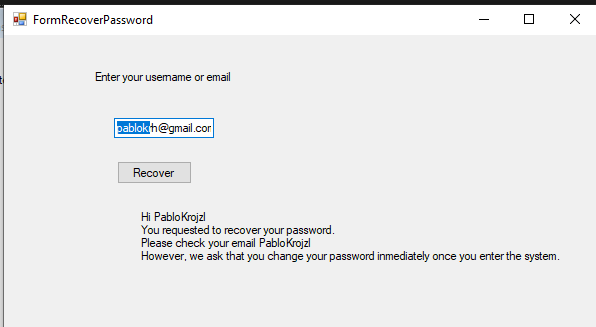
Hace falta un correo para enviar correos.



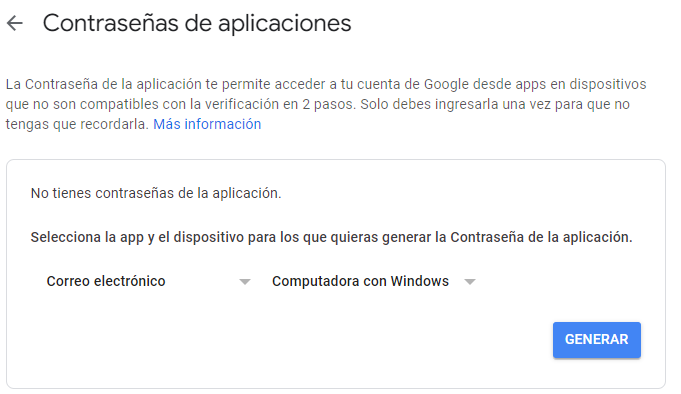


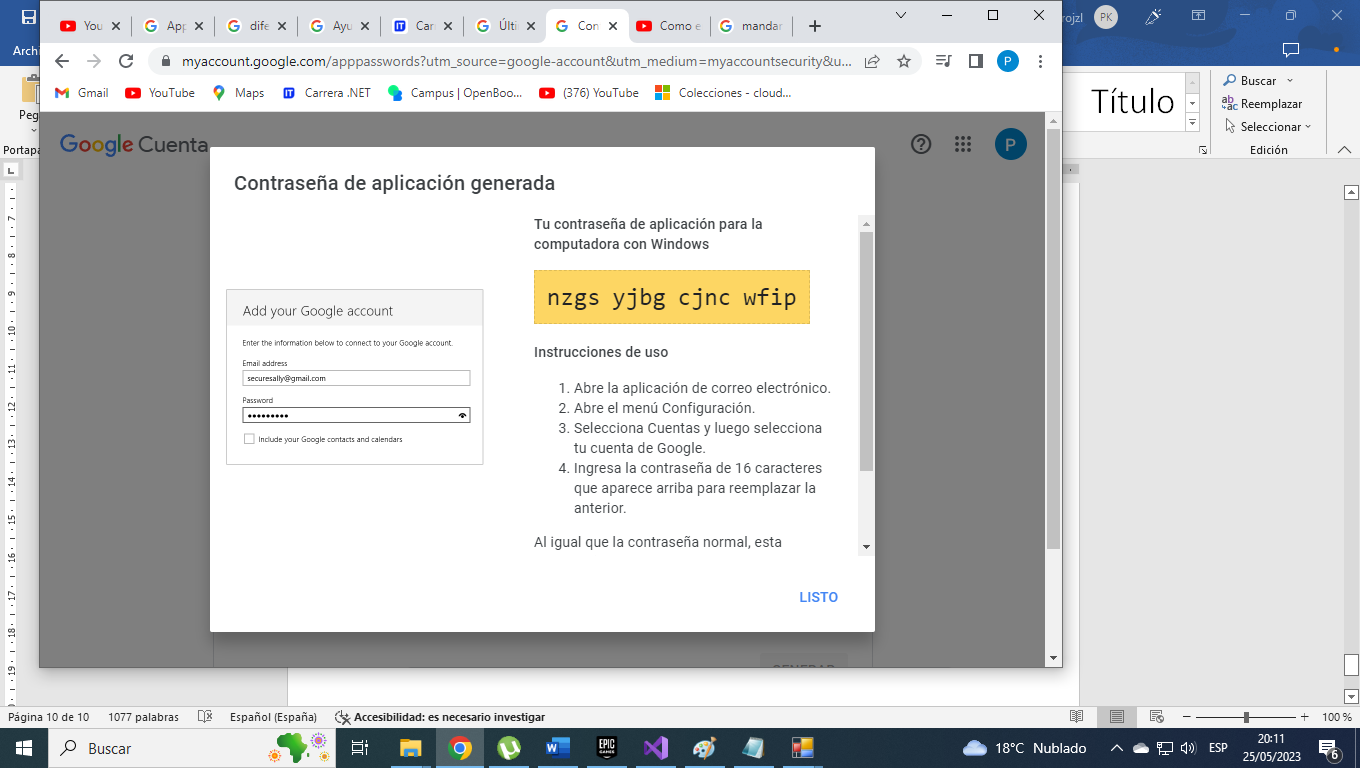
Hacen falta esos 4 datos para configurar el SMTP.

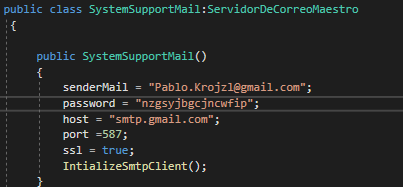
A la vez, hay que hacer otras configuraciones. Con Gmail hay que ir a cuenta de Google: Seguridad:Acceso de aplicaciones poco seguras=>Hay que activarlo.



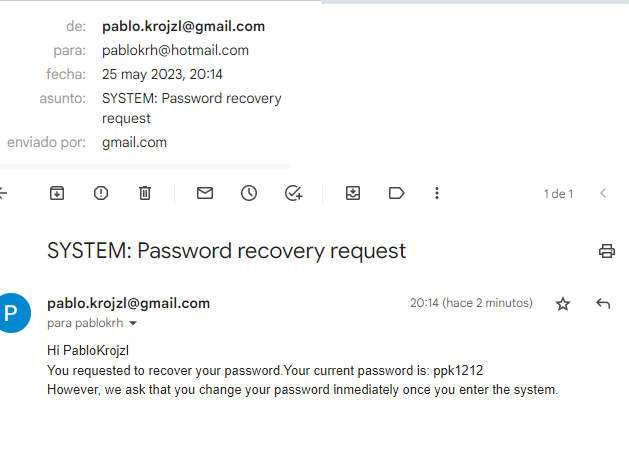
Google:tocando en seguridad en 2 pasos llegué a “Contraseñas de Aplicaciones”.







Ahí la puse… Y llegó el mail



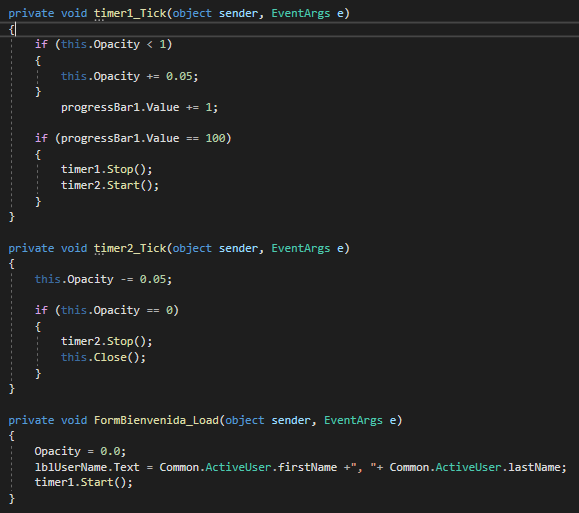
Capítulo 5

Crea una pantalla de presentación tras el login.

Crea un nuevo formulario en capa de presentación

Por si las moscas, las imágenes las maneja con pictureBox y con la propiedad Zoom, si no se va todo al carajo.

Agrega 2 timer. En la propiedad intervalo le pone 30. Le primero va a hacer que aparezca gradualmente el formulario (fade) y el segundo hará lo mismo pero para cerrarse.



Agrega una progressBar, reemplazando al contador.

Para darle estilo visual a la progressBar no alcanza con modificar sus propiedades. Hay que tocar la clase program:

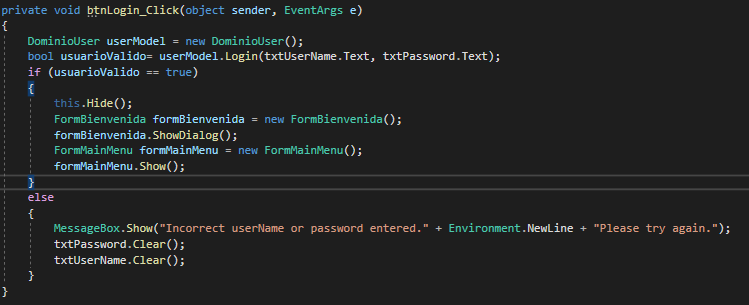


Pero dice que al deshabilitar eso pueden generarse problemas de incompatibilidad.

Remplaza la progressbar por CircularProgressBar, que no está en WF salvo que se descarguen paquete:



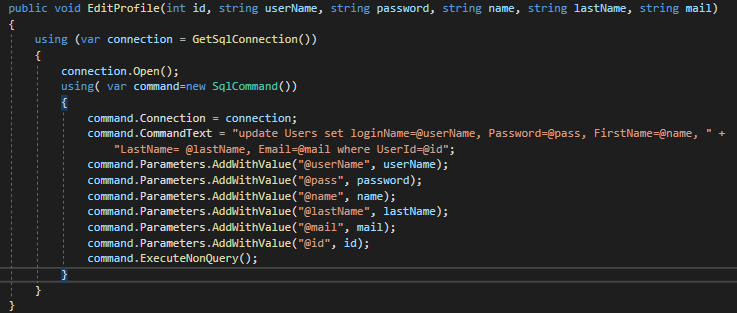
Así queda el formulario login. Primero se esconde ese, después se llama a formbienvenida como ShowDialog y ahí mismo se llama al form Menú como .Show.



Capítulo 6

Editar perfil/Contraseña

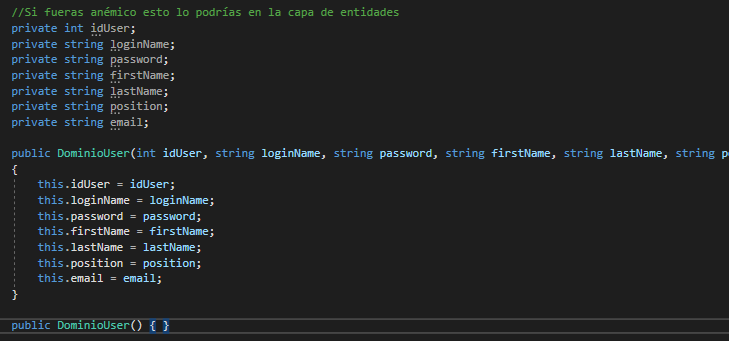
Agrega en capa de acceso a datos:



Luego, se va al modelo de usuario de la capa de dominio.

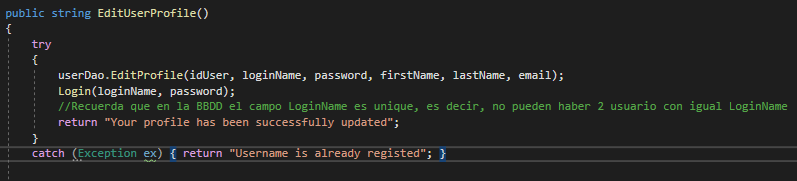
Reitera que en la capa de dominio se aplica la lógica de negocio.

Arquitectura. RJ code dice que crear una capa de entidades es “Modelo anémico”, eso estaría en contra de ciertos ppios, pero no estaría mal tampoco.



En dominio user, pone los atributos del objeto. Deja el constructor por defecto y agrega otro, al que se le pasan todos sus atributos.

Ahí abajo agrega este método:



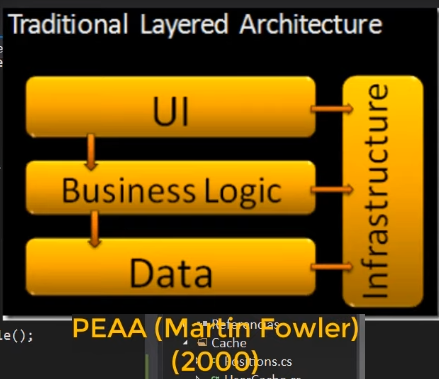
Agrega otro formulario en capa de presentación.

Aclara que, puntualmente en la edición del perfil, habría que realizar muchas verificaciones. No las desarrolla.

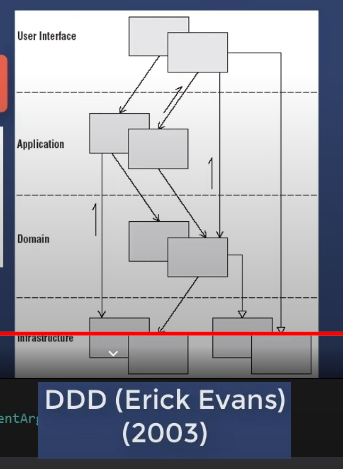
Aclara también, que en la capa de acceso a datos no se deberían generar los mensajes que después se muestran en la interfaz gráfica, eso debiera hacerse en la capa de dominio. En la capa de presentación también pueden crearse. Desde la capa de acceso a datos se puede devolver algún booleano u otro valor, que después si, se convierta en un msg.

También, como ve que hay bastante cosa que se repite en los métodos que conectan con la BBDD, afirma que podría usarse una clase abstracta y que hereden de allí.

RJ code dice que se basó en esto para hacer la app de sistema de gestión de servicios de salud mental. (al final del video 6 muestra una versión mas avanzada de la que hice yo).



Creo que la que viene es la que usó en la versión avanzada. Afirma que es para empresas un poco mas grandes que la arquitectura anterior, que es la mas usada, en empresas mas pequeñas.



Recomienda libros

PATRONES DE ARQUITECTURA DE APLICACIONES EMPRESARIALES. FOWLER, MARTIN

DISEÑO DIRIGIDO POR EL DOMINIO. EVANS, ERICK.

DICE TAMBIÉN QUE EN LA CAPA DE PRESENTACIÓN:

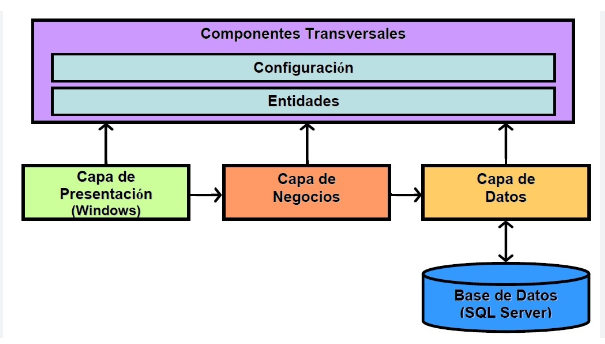
SE PUEDE USAR MVP (CON WINDOWS FORMS).

SE PUEDE USAR MODELO-VISTA. VISTA-MODELO. (WPF).

APP WEB SE PUEDE USAR MVC.

TODO ESTO PARA APPS DE GRAN TAMAÑO CON LÓGICA EMPRESARIAL COMPLEJA.

Arquitectura



Pongo esto acá para señalar que entidades refiere por ejemplo a Cliente, Empleado. Se crea eso en un proyecto, al cual van a referir todos los otros proyectos.