

Creación de la Interfaz para el trabajo con la lista

Crearé una lista, que le permitirá que una vez que este atendiendo a un cliente o paciente, pueda llenar todos sus datos, insertarlos, eliminarlos y moverse por la lista.

***Tomar en cuenta, que deberá cambiar los nombres de los controles, según el nombre que usted haya puesto en la propiedad name de cada control.

***Mantenga siempre una copia de su proyecto

1- Creación de la Interfaz:

1.- En el panel de atención, adicionara los datos que faltan para trabajar con el cliente o paciente, si es que fuera necesario, además adicionará los botones Agregar, Eliminar, Ir al primero, al ultimo, al anterior, al siguiente quedando la interfaz de la siguiente forma:

Para la clínica veterinaria:
<ul style="list-style-type: none">• Los títulos a adicionar serán:• Adicione dos etiqueta; una para costo total, donde cambie la propiedad text a Costo total, que estará en dependencia de la cantidad de mascotas que trajo y la otra para saber si ha pagado o no, adicione un control groupBox, y adicione dos controles radiobutton, uno para seleccionar si y otro para seleccionar no.• Adicione los botones, para insertar, eliminar, ir al primero, ir al ultimo, anterior y siguiente, deberá cambiar la propiedad name, poniendo btnInsertarL, btnEliminarL, btnPrimero, btnUltimoL, btnAnteriorL y btnSiguienteL, respectivamente. <p>Quedando la interfaz de la siguiente forma:</p>

Creación de la clase para trabajar con la lista:

Crearé la clase lista según su trabajo, adicionando dicha clase y su código a la namespace que tiene ya creado, para el trabajo con las clases:

Para la veterinaria: Crearé dos clases, la clase `datos_atencion` y clase `lista_Atencion_consultorio`

public ref class datos_atencion

{

private:

DateTime fechaAtencion;

double costo;

bool pagado;

datos^ dueno;

public:

datos_atencion(DateTime fechaAtencion, double costo, bool pagado, datos^ dueno) {

this->fechaAtencion = fechaAtencion;

this->costo = costo;

this->pagado = pagado;

this->dueno = dueno;

}

datos^ getdueno() { return dueno; }

DateTime getfechaAtencion() { return fechaAtencion; }

double getcosto() { return costo; }

```
bool getpagado() { return pagado; }

void setdueno(datos^ _dueno) { dueno = _dueno; }
void setfechaAtencion(DateTime _fechaAtencion) { fechaAtencion = _fechaAtencion; }
void setcosto(int _costo) { costo = _costo; }
void setpagado(bool _pagado) { pagado = _pagado; }
};

public ref class lista_Atencion_veterinaria
{
private:
    List <datos_atencion^>^ Lista; //defino la lista

public:
    lista_Atencion_veterinaria() {
        Lista = gcnew List<datos_atencion^>();
    }

    void Agregar(datos_atencion^ c) {
        Lista->Add(c);
    }

    datos_atencion_1Primero(){
        if (Lista->Count > 0)
            return Lista[0];
        else
            return nullptr;
    }

    datos_atencion^ Ultimo(){
        if (Lista->Count > 0)
            return Lista[Lista->Count - 1];
        else
            return nullptr;
    }

    datos_atencion^ Anterior(String^ nombreBuscado){
        for (int i = 0; i < Lista->Count; i++) {
            if (Lista[i]->getdueno()->getDueño()->Equals(nombreBuscado,
StringComparison::InvariantCultureIgnoreCase)) {
                if (i > 0)
                    return Lista[i - 1]; // anterior
            }
        }
    }
}
```

```
        else
            return nullptr; // no hay anterior
        }
    }
    return nullptr; // no se encontró la mascota
}

datos_atencion^ Siguiente(String^ nombreBuscado){
for (int i = 0; i < Lista->Count; i++) {
if (Lista[i]->getdueno()->getDueño()->Equals(nombreBuscado,
StringComparison::InvariantCultureIgnoreCase)) {
        if (i < Lista->Count - 1)
            return Lista[i + 1]; // siguiente
        else
            return nullptr; // no hay siguiente
    }
}
return nullptr; // no se encontró la mascota
}

int ObtenerCantidad() {
    return Lista->Count;
}

List<datos_atencion^>^ ObtenerLista() {
    return Lista;
}

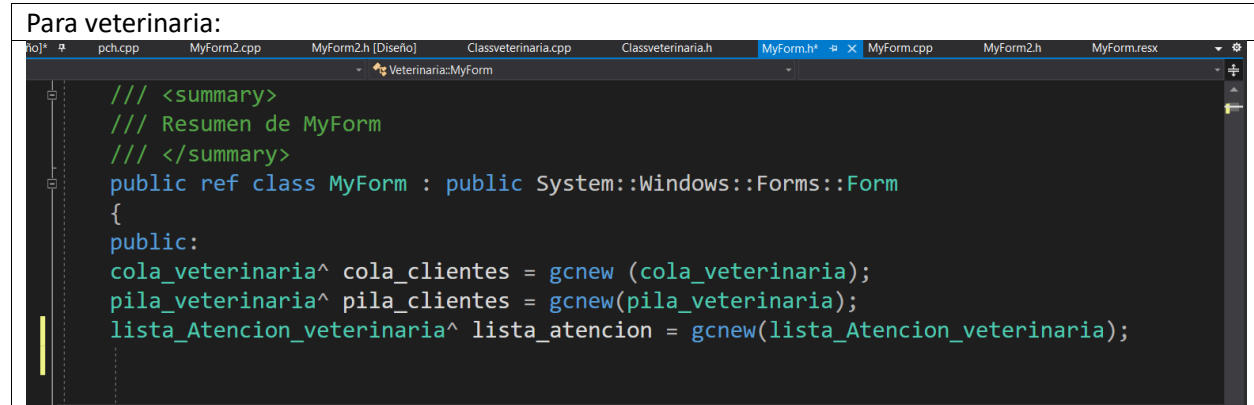
bool Eliminar(String^ nombreBuscado){
for (int i = 0; i < Lista->Count; i++) {
if (Lista[i]-> getdueno()->getDueño()->Equals(nombreBuscado,
StringComparison::InvariantCultureIgnoreCase)) {
        Lista->RemoveAt(i); // elimina
        return true;
    }
}
return false;
}

};
```

Trabajando con el objeto lista:

1.- Adicionar la definición del objeto lista

Para veterinaria:



```

/// <summary>
/// Resumen de MyForm
/// </summary>
public ref class MyForm : public System::Windows::Forms::Form
{
public:
cola_veterinaria^ cola_clientes = gcnew (cola_veterinaria);
pila_veterinaria^ pila_clientes = gcnew(pila_veterinaria);
lista_Atencion_veterinaria^ lista_atencion = gcnew(lista_Atencion_veterinaria);

```

2.- Adicionar código al evento clic de cada boton: insertar, eliminar, primero, ultimo, anterior y siguiente.

Boton Insertar

Para la Clinica Veterinaria:

```

if (txtANombre->Text != "") {
bool pagado;
double pago = Double::Parse(txtATotal->Text);
String^ texto = txtACantidad->Text;
int numero = Int32::Parse(texto);
if (radioButtonSI->Checked == true)      pagado = true;
if (radioButtonNO->Checked == true) pagado = false;

datos^ nuevo = gcnew datos(txtANombre->Text, txtAContacto->Text, txtAHora->Text,numero);
datos_atencion^ nueva = gcnew datos_atencion_1(dateTimePicker1->Value, pago, pagado,nuevo );
lista_atencion->Agregar(nueva);
clear_ventana();
atendido = true;
}

```

Boton Primero**Para la Clinica Veterinaria:**

```
datos_atencion^ primero;  
primero = lista_atencion->Primero(); //Devuelve el primero de la lista  
if (primero != nullptr) {  
    txtANombre->Text = primero->getdueno()->getDueño();  
    txtAContacto->Text = primero->getdueno()->getContacto();  
    txtAHora->Text = primero->getdueno()->getHora_llegada();  
    txtATotal->Text = primero->getcosto().ToString();  
    if(primero->getpagado()) {  
        radioButtonSI->Checked = true;  
        radioButtonNO->Checked = false }  
    else {  
        radioButtonSI->Checked = false;  
        radioButtonNO->Checked = true;  
    }  
}
```

Boton Ultimo**Para la Clinica Veterinaria:**

```
datos_atencion^ ultimo;  
ultimo = lista_atencion->Ultimo(); //Devuelve el primero de la lista  
if (ultimo != nullptr) {  
    txtANombre->Text = ultimo->getdueno()->getDueño();  
    txtAContacto->Text = ultimo->getdueno()->getContacto();  
    txtAHora->Text = ultimo->getdueno()->getHora_llegada();  
    txtATotal->Text = ultimo->getcosto().ToString();  
    if (ultimo->getpagado()) {  
        radioButtonSI->Checked = true;  
        radioButtonNO->Checked = false;  
    }  
    else {  
        radioButtonSI->Checked = false;  
        radioButtonNO->Checked = true;  
    }  
}
```

Boton Anterior**Para la Clinica Veterinaria:**

```
datos_atencion^ anterior;
anterior = lista_atencion->Anterior(txtANombre->Text); //Devuelve el anterior a uno de la lista
if (anterior != nullptr) {
txtANombre->Text = anterior->getdueno()->getDueño();
txtAContacto->Text = anterior->getdueno()->getContacto();
txtAHora->Text = anterior->getdueno()->getHora_llegada();
txtATotal->Text = anterior->getcosto().ToString();
if (anterior->getpagado()) {
radioButtonSI->Checked = true;
radioButtonNO->Checked = false;
}
else {
radioButtonSI->Checked = false;
radioButtonNO->Checked = true;
}
}
```

Boton Siguiente**Para la Clinica Veterinaria:**

```
datos_atencion^ siguiente;
siguiente = lista_atencion->Siguiente(txtANombre->Text); //Devuelve el anterior a uno de la lista
if (siguiente != nullptr) {
txtANombre->Text = siguiente->getdueno()->getDueño();
txtAContacto->Text = siguiente->getdueno()->getContacto();
txtAHora->Text = siguiente->getdueno()->getHora_llegada();
txtATotal->Text = siguiente->getcosto().ToString();
if (siguiente->getpagado()) {
radioButtonSI->Checked = true;
radioButtonNO->Checked = false;
}
else {
radioButtonSI->Checked = false;
radioButtonNO->Checked = true;
}
}
```

Boton Eliminar**Para la Clinica Veterinaria:**

```
if (lista_atencion->Eliminar(txtPNombre->Text)) {  
    MessageBox::Show("Se eliminó con éxito", "Éxito", MessageBoxButtons::OK,  
        MessageBoxIcon::Information);  
}  
else {  
    MessageBox::Show("No se encontró ese nombre", "Error", MessageBoxButtons::OK,  
        MessageBoxIcon::Warning);  
}
```