Universidad Tecnológica Nacional
Facultad Regional Avellaneda

Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:				Fecha:		04/08/2022				
Nombre:				Docente(2):						
División:	2º			Nota(2):						
Legajo:				Firma(2):						
Instancia(1):	**PP**		**RPP**		**SP**		**RSP**	X	**FIN**	

(1) Las instancias validas son: 1er Parcial (**PP**), Recuperatorio 1er Parcial (**RPP**), 2do Parcial (**SP**), Recuperatorio 2do Parcial (**RSP**), Final (**FIN**). Marque con una cruz.
(2) Campos a ser completados por el docente.

IMPORTANTE:

- ***2 (dos) errores en el mismo tema anulan su puntaje.***
- **La correcta documentación y reglas de estilo de la cátedra serán evaluadas.**
- Colocar sus datos personales en el nombre del proyecto principal, colocando: Apellido.Nombre.Departamento. Ej: Pérez.Juan.2D. No sé corregirán proyectos que no sea identificable su autor.
- **TODAS** las clases deberán ir en una Biblioteca de Clases llamada Entidades.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.
- Colocar nombre de la clase (en estáticos), **this** o **base** en todos los casos que corresponda.
- Colocar el nombre del docente con el cual cursó.

---

*TIEMPO MÁXIMO PARA RESOLVER EL EXAMEN **90** MINUTOS.*

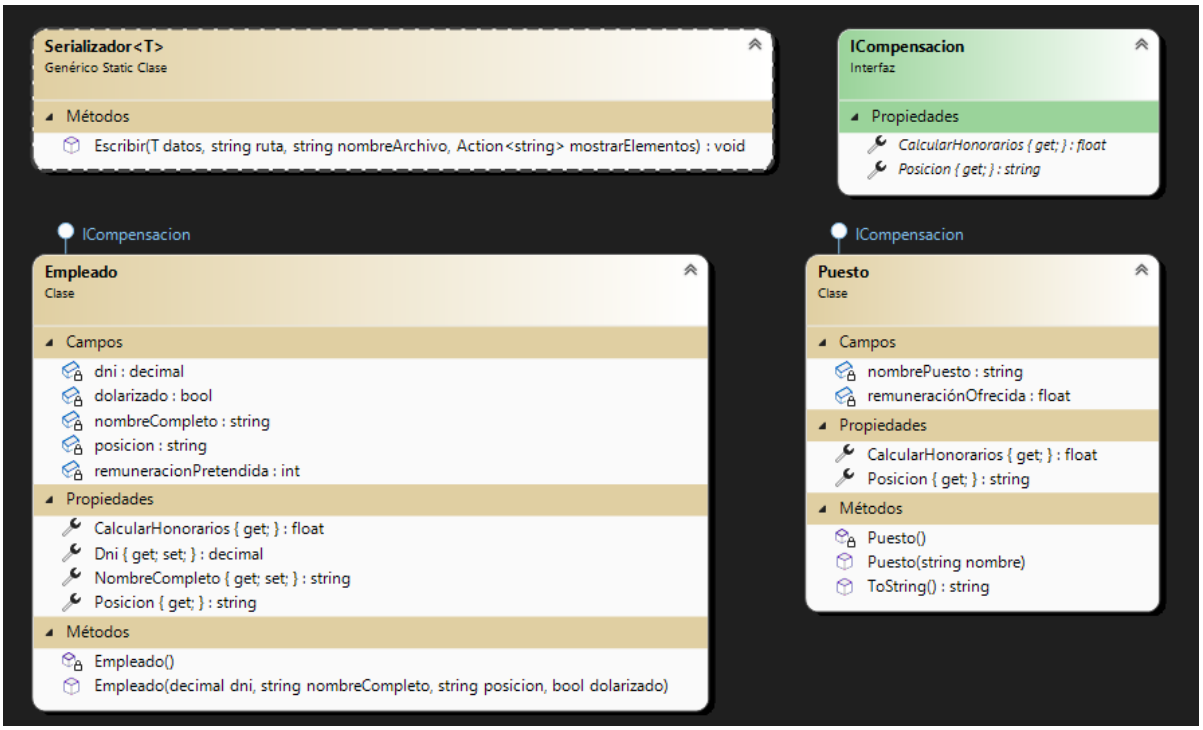
Introducción

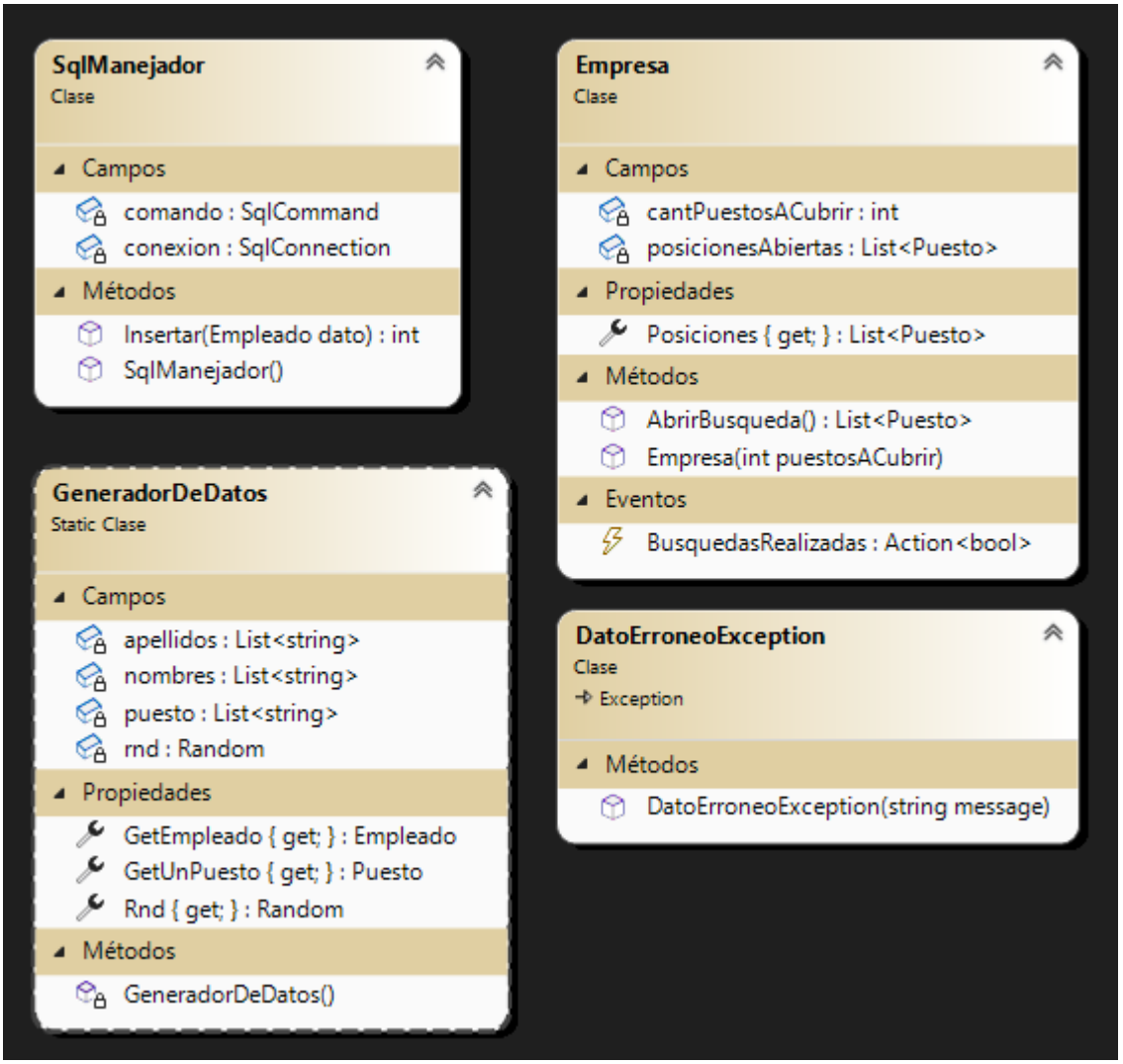
Junto con este documento, recibieron una aplicación a la cual hay que terminar las funcionalidades detalladas más adelante. Tener en consideración que hay funcionalidades ya desarrolladas que **no se deben cambiar.**

De todas formas,si algo les genera duda, pregunten.

Secciones de la aplicación

Biblioteca de clases





Interfaz ICompensacion:

- La interfaz **ICompensacion** deberá implementarse en las clases Empleado y Puesto.
    - En **Empleado**:
        - **CalcularHonorarios**: Consultará si el sueldo del empleado está dolarizado o no. Si está dolarizado, lo convertirá a Dólares (lo dividirá por 300).
        - **Posición**: Retorna el atributo Posición.
    - En **Puesto**:
        - **CalcularHonorarios**: Si el sueldo supera un monto de 200.000, devolverá solo el 75% del sueldo, dado que el resto fue retenido por impuestos.
        - **Posición**: Retorna el nombre del puesto;

Empleado:

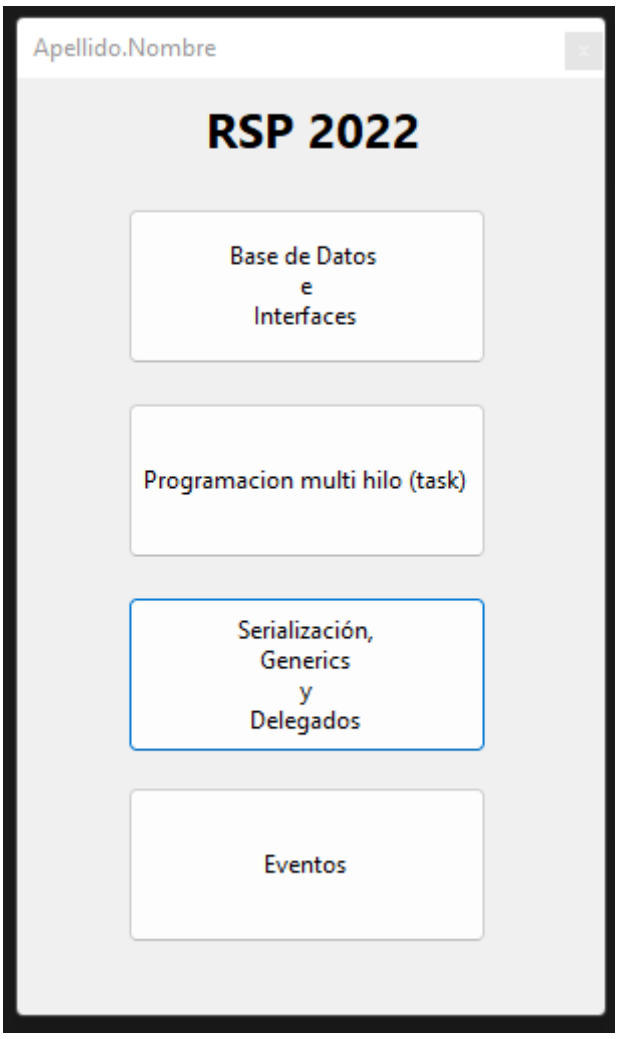
- Su constructor privado le dará un valor de entre 100000 y 300000 al atributo **remuneracionPretendida**, utilizando la propiedad Rnd de la clase **GeneradorDeDatos** (Esto ya esta codeado, solo utilizarlo).

Puesto:

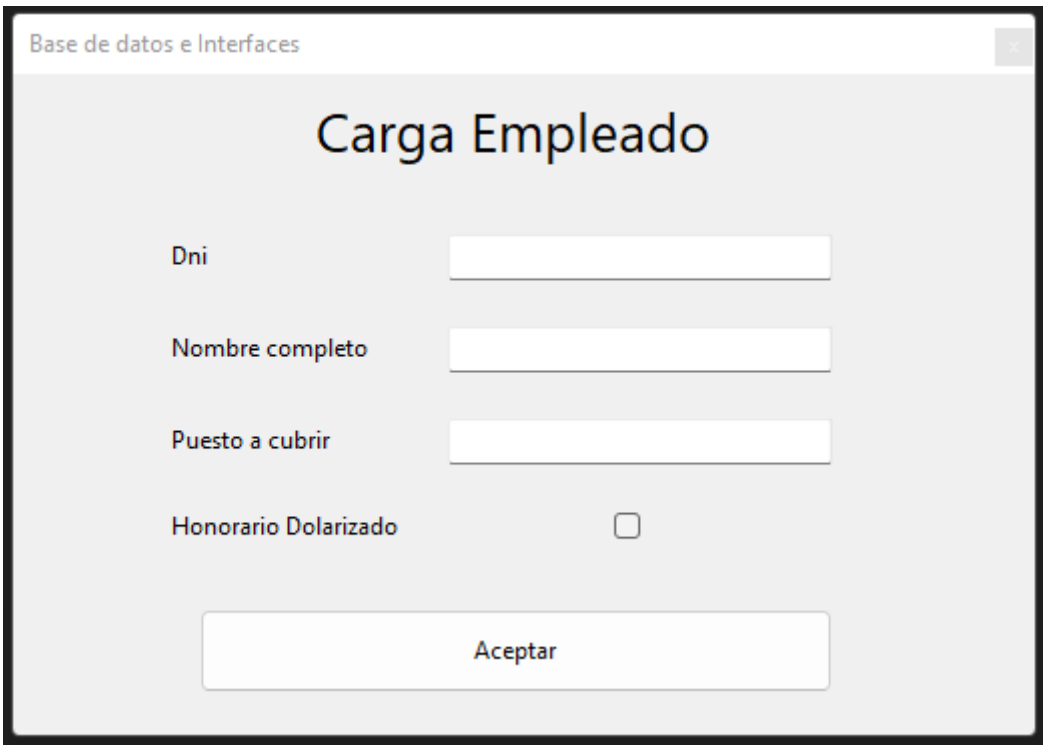
- Su constructor privado le dará un valor de entre 100000 y 250000 al atributo **remuneraciónOfrecida**, utilizando la propiedad Rnd de la clase **GeneradorDeDatos** (Esto ya esta codeado, solo utilizarlo).

Menú principal

- En el menú principal de la aplicación solo deben reemplazar el “Apellido.Nombre” por sus propios datos.



Base de datos e Interfaces



Este módulo deberá insertar un Empleado en la base de datos (Al final de este documento encontrarán el script para crear la tabla en la base de datos).

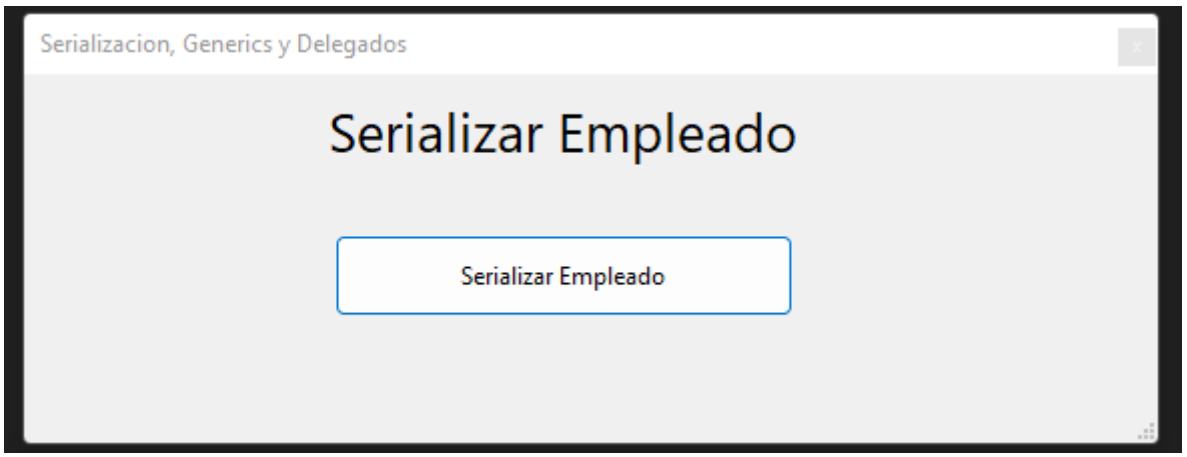
A tener en cuenta:

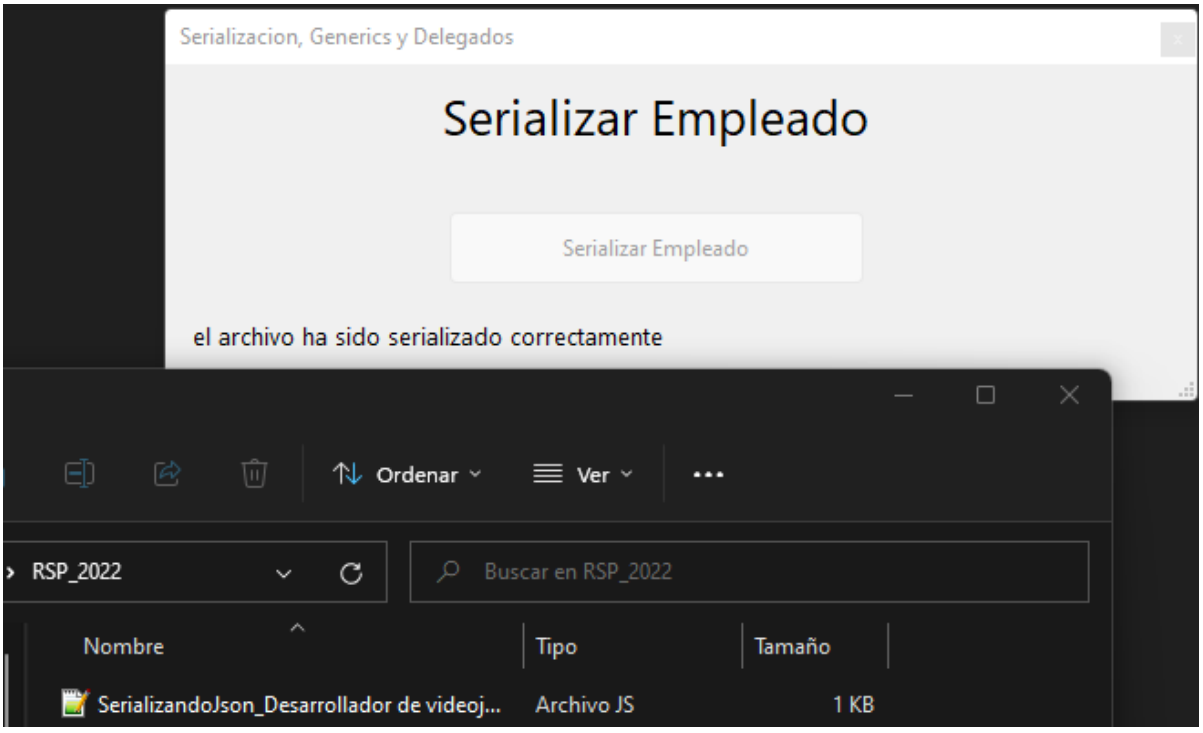
- Deberán validar que los datos sean correctos. Es decir:
    - Dni: Deberá tener un valor entre 10.000.000 y 45.000.000
    - Nombre: No puede ser ni nulo ni vacío.
    - Si el tilde está checkeado, será dolarizado el sueldo de esa persona.

- La validación no se hará en el form. Será en el método de **instancia** Insertar de la clase **SqlManejador** (también será de instancia). Si algún valor es incorrecto, se arrojará una excepción de tipo **DatoErroneoException**, que será capturada en formulario mostrando en un messagebox el mensaje.

Serialización, Generics y Delegados

Este módulo lo que hará es serializar un Empleado y mediante un delegado, desactivar el botón para serializar otro empleado y mostrar en pantalla la leyenda “El empleado ha sido serializado correctamente”.





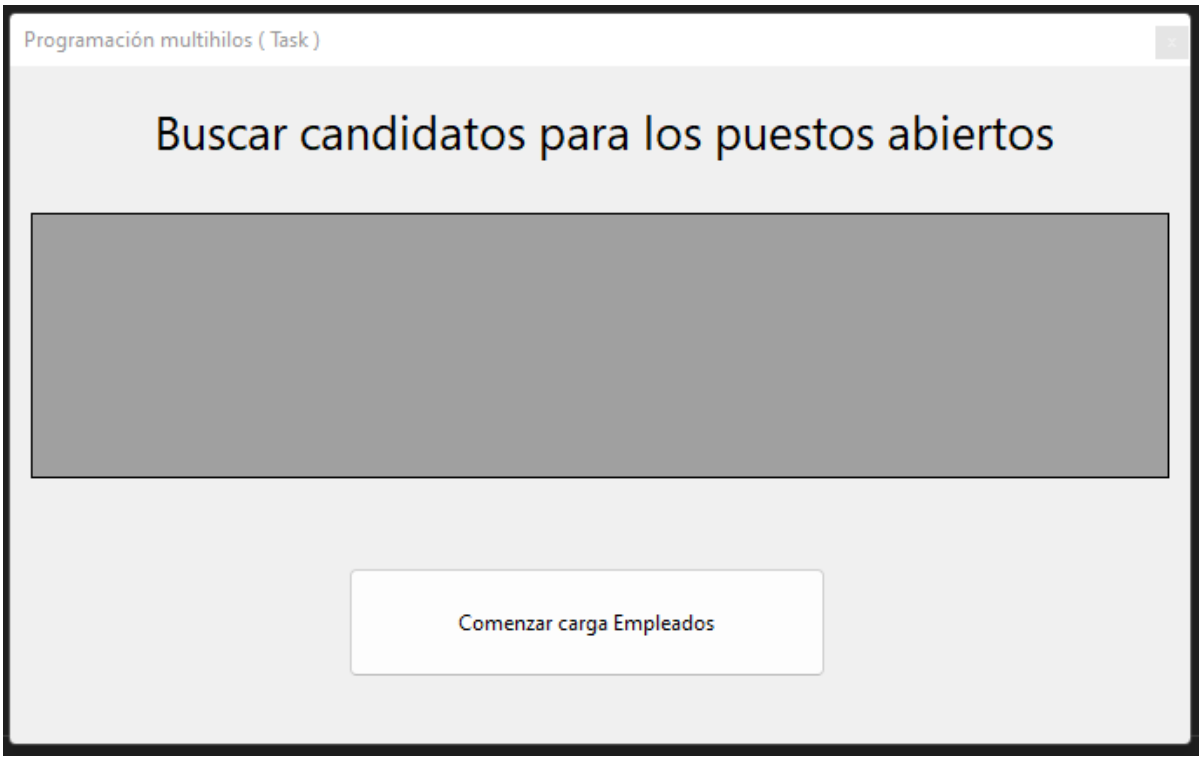
Programación MultiHilos (Task)

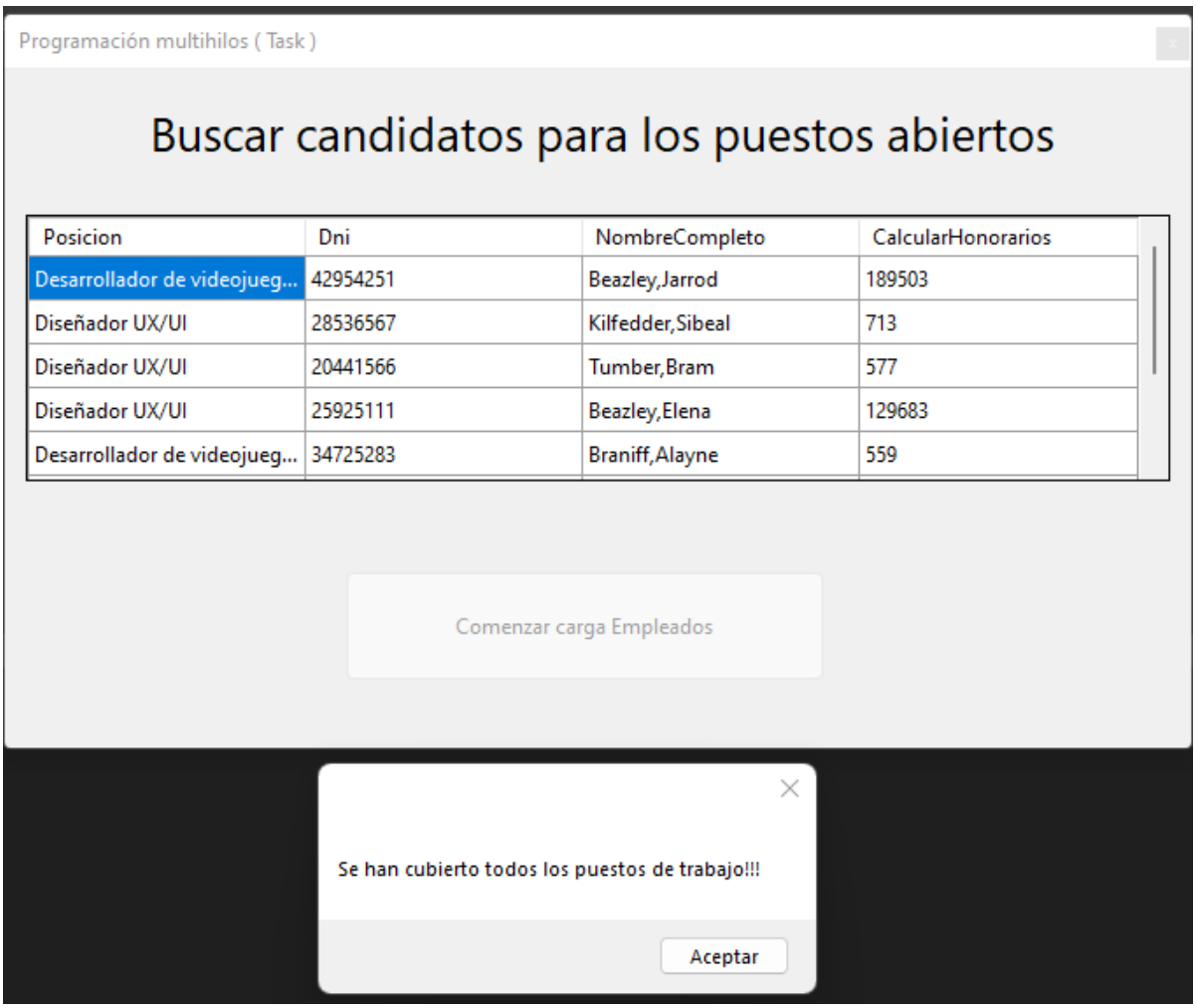
El botón comenzar carga simulará un grupo de empleados que se postulan a los puestos abiertos, hasta que todas las posiciones hayan sido cubiertas. Una vez cubiertas, se desactivará la búsqueda.

Dicho método hará el proceso de::

- Obtendrá un Empleado random.

- Comprobará si las posiciones abiertas tienen una búsqueda para la posición de ese empleado,y si el sueldo pretendido del empleado está dentro de lo que ofrece el puesto.
- Agregará a ese empleado a la lista de postulantes.
- Actualizar el datagrid con la lista de postulantes.
- Repetirá este proceso cada 2 segundos hasta que se cancele la tarea presionando el botón “Cancelar Carga…”



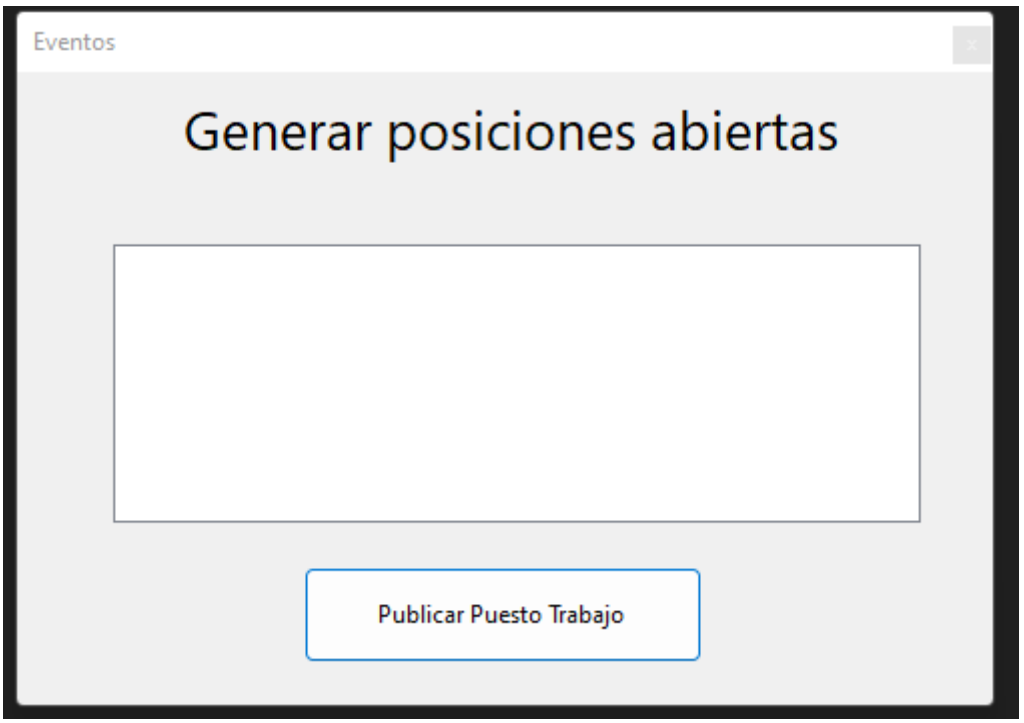


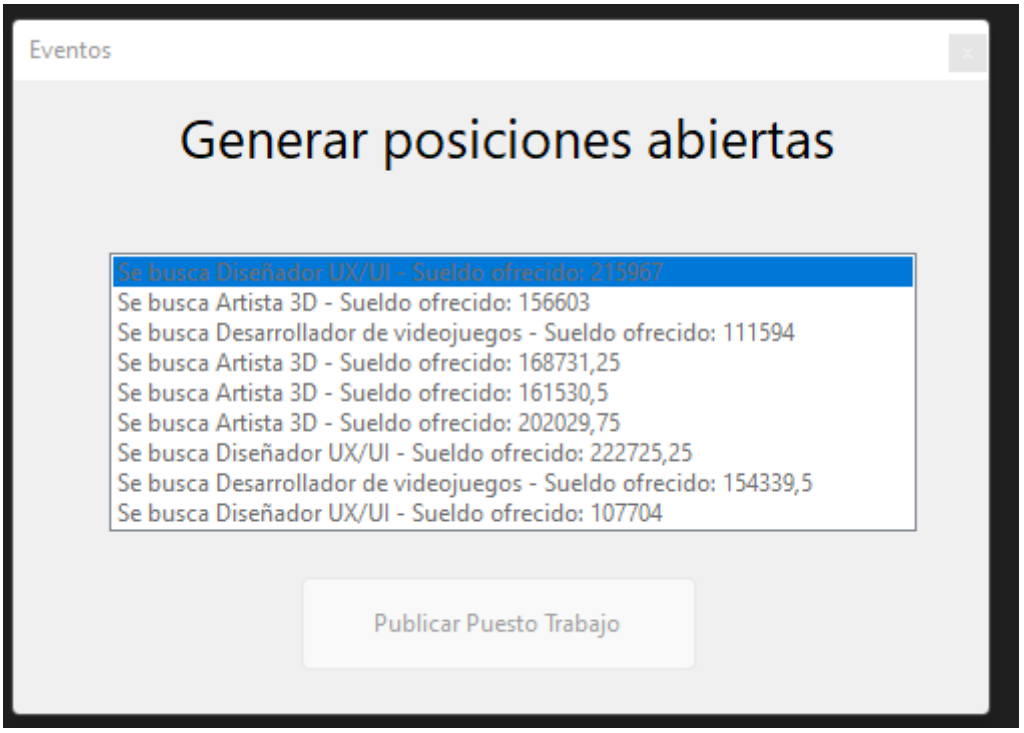
Eventos

Se presionará el botón “Publicar Puesto Trabajo” hasta que el cupo de postulaciones de la empresa esté lleno.
Al llenarse, se deshabilitarán los componentes y se mostrará por pantalla el siguiente mensaje en un messagebox:

“Todas las postulaciones han sido realizadas con éxito.
Agradecemos su participacion.
Los esperamos la proxima“.

*Tip: La propiedad **GetUnPuesto** de la clase estática **GeneradorDeDatos** devuelve un Puesto aleatorio.*





Test Unitarios

Se piden **dos** de los siguientes casos:

- Testear que al insertar un Empleado con valores incorrectos, se arroja correctamente la excepción que corresponde.

- Testear que un Empleado se insertó correctamente. (Tip: El método ExecuteNonQuery() devuelve el número de rows insertadas, por lo tanto si es mayor a 0, quiere decir que se insertó correctamente).

- Testear que los métodos **GetUnEmpleado** y **GetUnPuesto** devuelven datos válidos (En el mismo unit testing).

Script Base de datos:

1) Crear una base de datos llamada **ExamenPrimerFecha2022**

2) Correr el siguiente script

```
USE [ExamenPrimerFecha2022]
GO
/****** Object:  Table [dbo].[Empleados]    Script Date: 1/8/2022 19:44:49 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Empleados](
	[Dni] [int] NOT NULL,
	[Nombre] [nvarchar](50) NOT NULL,
	[Posicion] [nvarchar](50) NOT NULL,
	[Honorario] [int] NOT NULL
) ON [PRIMARY]
GO
```