**Documentación reto Servicios Web**

* **Planteamiento del desafío**

Para el desafío se planteo hacer una arquitectura SOAP en la cual hubiera un servidor que brindara un servicio y un cliente que lo consumiera. El servidor debe contar con datos ya fueran quemados, en una base de datos o usar una API ya hecha como servicio y de esta consumir los datos, como mínimo tienen que haber 3000 registros en cualquiera de las formas descritas anteriormente. El cliente puede ser una pagina web una app o una aplicación de Windows forms, este debe tener la funcionalidad de presentar los datos dentro de una tabla y exportar los datos a xls.

* **Desarrollo del desafío**

A continuación, se mostrará las herramientas utilizadas para poder desarrollar el proyecto de manera exitosa y los pasos que se llevaron a cabo.

* **Herramientas con las que se desarrolló el desafío**

1. **IDE**

Como entorno de desarrollo se utilizó visual studio community.

1. **Plantilla de proyecto**

- Para la parte del servidor se realizó con la plantilla “aplicación de servicios WCF” (Windows Communication Foundation), la cual es un marco para la creación de aplicaciones orientadas a servicios.

- Para la parte del cliente se hizo uso de la plantilla Aplicación de Windows forms (.NET Framework), ósea una aplicación de escritorio de Windows.

1. **Lenguaje de programación**

El desarrollo del proyecto se realizó con c #.

1. **Sistema gestor de bases de datos**

Se tomo como sistema gestor MYSQL con XAMPP.

1. **Exportación a xlsx**

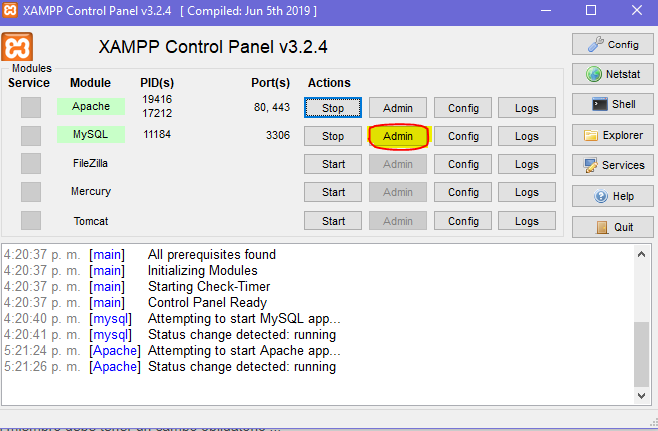
Para la exportación a Excel se utilizo SpreadsheetLight el cual es una librería de .NET Framework con la cual se pueden crear archivos Open XML sin necesidad de una licencia de office.

* **Pasos realizados para el desarrollo del proyecto**

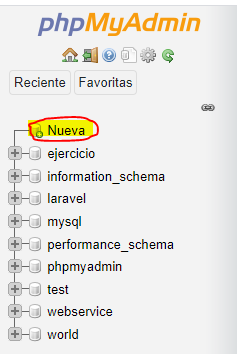
Para realizar el proyecto primero se debe contar con un servidor o servicio el cual se pueda consumir, para esto se realizaron los siguientes pasos:

1. **MYSQL -XAMPP**
2. **Creación de la base de datos**

Se abre el XAMPP y se enciende el Apache y el MYSQL, cuando prenda se da en el botón de admin que se encuentra en amarillo, este botón lleva a la pagina de phpmyadmin desde donde se creara la base de datos.

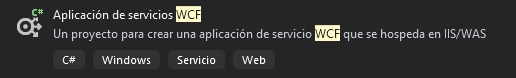


Cuando estemos en la pagina se da en nueva

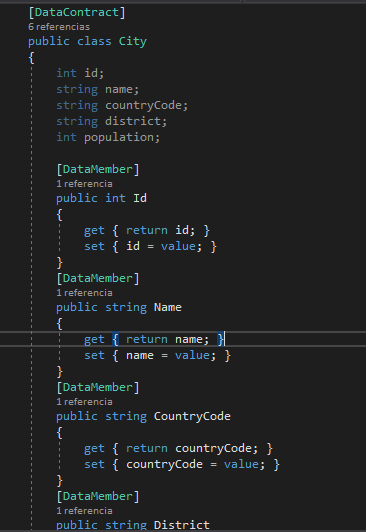


Se le da el nombre y el cotejamiento que se desea, luego de esto se crean las tablas y se le cargan sus respectivos registros.

1. **Servidor**
   1. **Creación del proyecto con la plantilla aplicación de servicios WCF**

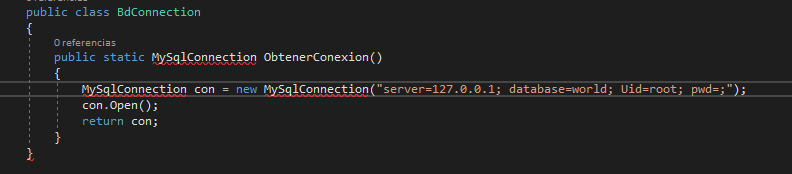


* 1. **Creación de la clase City, su contrato y sus respectivos atributos, los cuales tienen la estructura del modelo de la base de datos para recibir los registros que se encuentran ahí almacenados.**



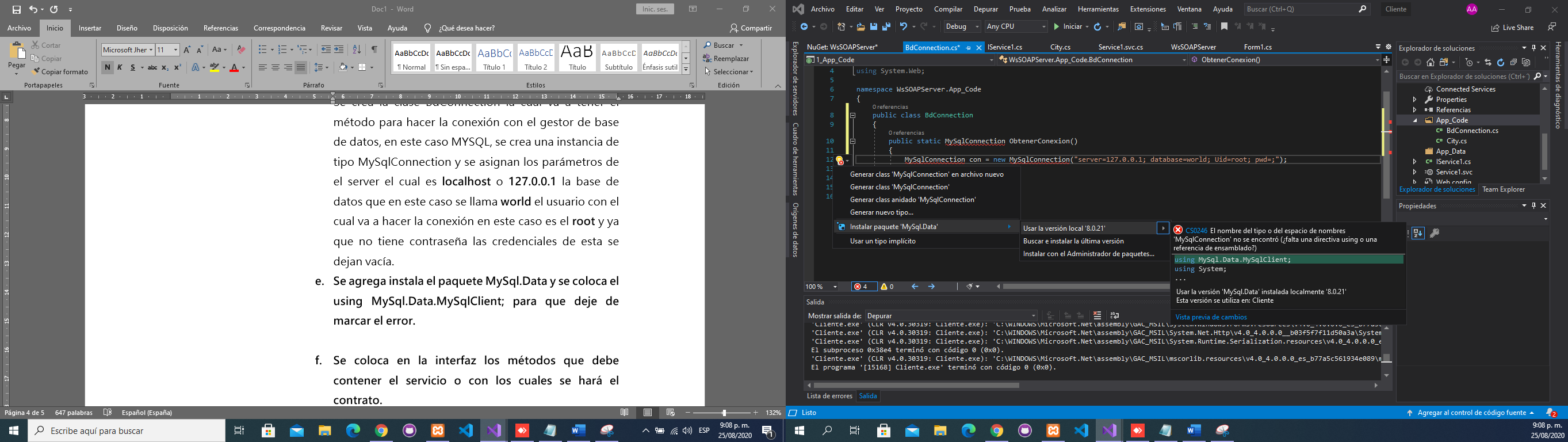
En esta clase los atributos se encuentran declarados y encapsulados para evitar que se pueda acceder a ellos directamente desde otras clases, además que cuentan con los data annotation los cuales sirven para decir que forman parte del contrato que se esta creando en el servidor.

* 1. **Creación de la conexión a la base de datos**

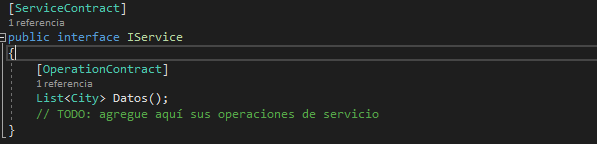


Se crea la clase BdConnection la cual va a tener el método para hacer la conexión con el gestor de base de datos, en este caso MYSQL, se crea una instancia de tipo MySqlConnection y se asignan los parámetros de el server el cual es **localhost** o **127.0.0.1** la base de datos que en este caso se llama **world** el usuario con el cual va a hacer la conexión en este caso es el **root** y ya que no tiene contraseña las credenciales de esta se dejan vacía.

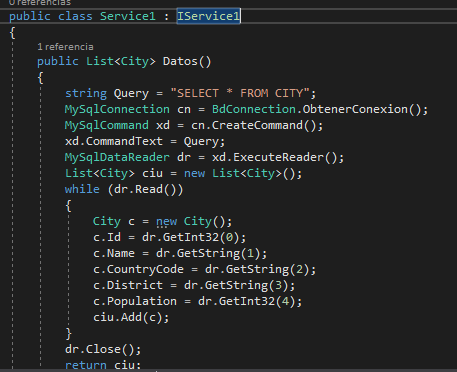
* 1. **Se instala el paquete MySql.Data y se coloca el using MySql.Data.MySqlClient; para que deje de marcar el error por falta del ensamblado.**



* 1. **Se crea la nueva operación del servicio o contrato especificando que es las cosas que debería tener para luego hacer uso de este (se eliminan las que están de ejemplo si así se quiere o se reemplazan)**

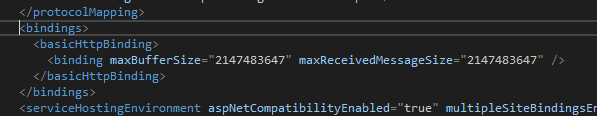


* 1. **Se crea dentro de la clase del servicio, el método que acabamos de estipular el cual devuelve una lista de datos de tipo “City”.**

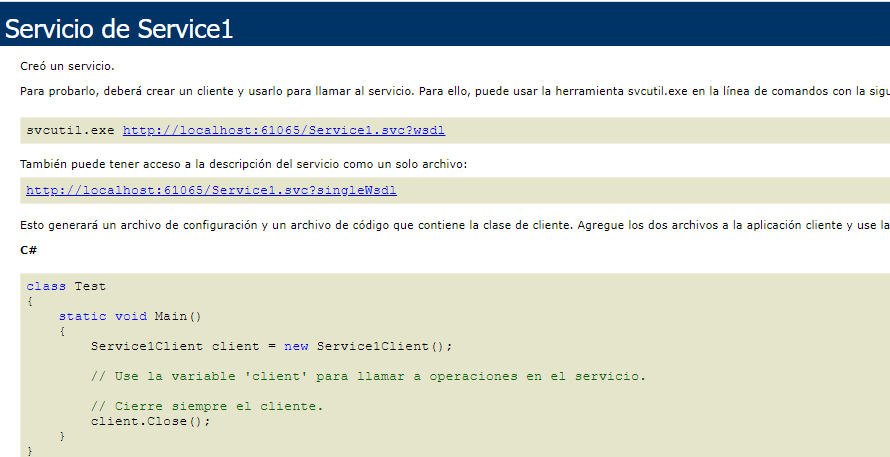


En este método preparamos la consulta que se hará a la base de datos, se crea la conexión haciendo uso del método que se creo anteriormente, se crea un objeto de tipo MySqlCommand el cual es el encargado de ejecutar las consultas, luego creamos un objeto de tipo MySqlDataReader el cual tiene los datos que devuelve la consulta, hacemos un ciclo para guardar cada uno de los datos dentro de una lista, esto mientras hayan datos por “leer”, se cierra y se retorna la lista.

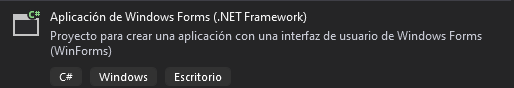
* 1. **Se añade en el Web.Config una configuración para que aumente la cantidad de datos que se pueden mandar y recibir por medio del “mensaje”.**



* 1. **Se corre el servicio para que se compilen todos los cambios hechos.**

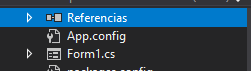
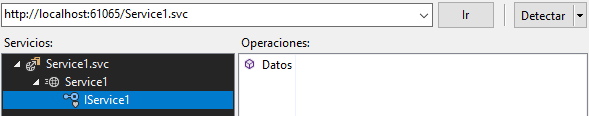


1. **Cliente**
   1. **Creación de la aplicación que va a consumir el servicio.**

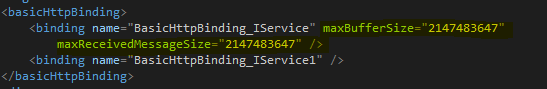


Se crea una aplicación de tipo Windows Forms

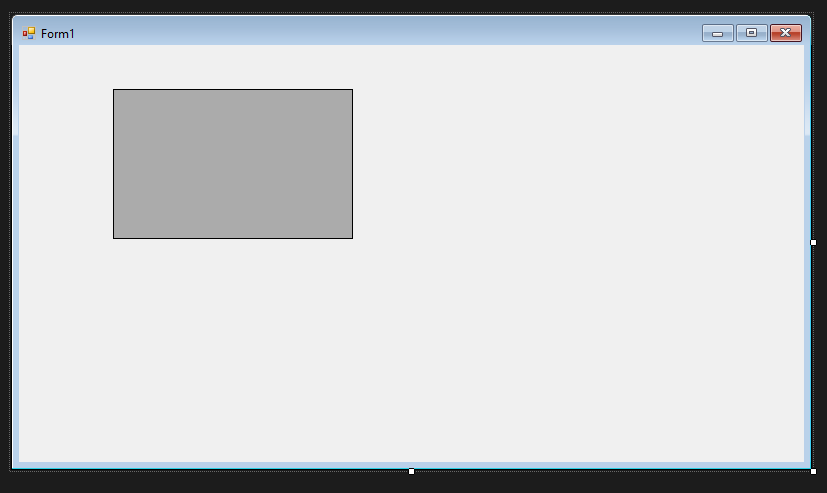
* 1. **Se agrega la referencia del servicio que se creó antes**

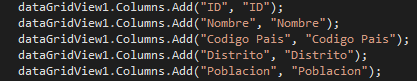
* 1. **Se agrega la configuración al App.Config para que pueda enviar más información al servidor de la “predeterminada”.**



* 1. **Se agrega un DataGridView**



* 1. **Desde el código se agregó el identificador y el nombre de las columnas**



* 1. **Se instancia una lista de objetos donde se van a almacenar los datos que se traigan desde el servidor**

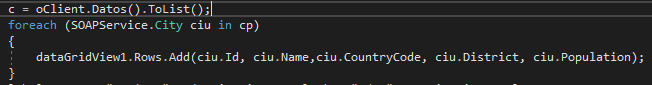


* 1. **Se instancia un objeto el cual hace la comunicación entre el cliente y el servidor a través de una topología de tipo proxy**



El proxy llega a ser como un intermediario el cual no sabe algo a menos que lo “pregunte”. Siguiendo la idea de que el cliente o la aplicación de Windows Forms le entrega los datos y el es quien ve que hacer con estos o por el contrario si le llega una información es quien se encarga y sabe cómo entregarla o qué hacer con ella. El Client define como la aplicación debe entregar los datos antes de que esta funcione.

* 1. **Se llenan las filas del DataGridView**

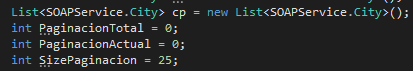


Como anteriormente “c” la definimos como una lista de la clase que trae el contrato del servicio, para que lo traiga como una lista y no como un array se le agrega a la llamada del método el ToList(); esto para más adelante ayudarnos a hacer la paginación de manera más sencilla.

* 1. **Se agregan los botones para la paginación**



* 1. **Se crea la lógica para cuando inicie la aplicación**

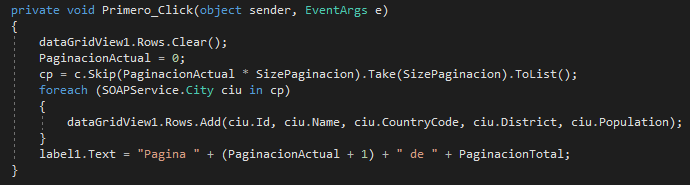


Se declaran de manera global los atributos que necesitamos para la lógica de paginar como lo es la paginación total que hay, en cual página se encuentra actualmente y cuantos registros queremos que se muestren, así como una lista de copia que utilizaremos.



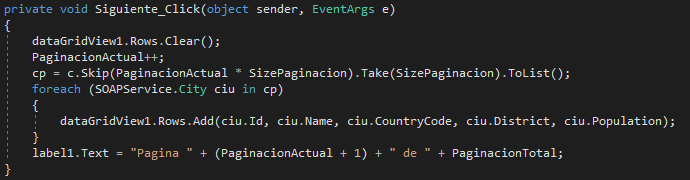
Dentro del constructor de la clase del formulario se le va a asignar el valor a la paginación total y se va a traer los datos que realmente se van a mostrar, esto lo hacemos tratando a la lista con LINQ con Skip el cual hace que se salte la cantidad que se coloque ahí y Take que es para decirle cuantos traiga, de una forma más es como un traiga tantos registros desde la posición x.

* 1. **Se hace la lógica para el botón de “Primera” el cual va a llevar a la primera página**



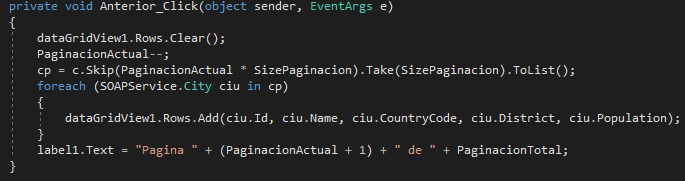
En este método limpiamos el DataGridView colocamos la paginación actual en 0 y se vuelve a calcular que registros es los que debe traer con los datos actuales, los cuales en este caso siempre llevara a la página 1.

* 1. **Se hace la lógica para el botón de “Siguiente” el cual lleva a la página siguiente de donde se encuentre la persona usando la aplicación**



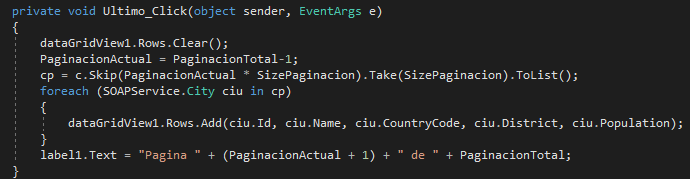
Se limpia el DataGridView se le suma uno a la paginación actual de donde se encuentre la persona y se vuelve a traer los registros con la nueva información.

* 1. **Se hace la lógica para el botón de “Anterior” el cual lleva a la página anterior de donde se encuentre la persona usando la aplicación**



Funciona de la misma manera de lo expuesto anteriormente con la diferencia de que en vez de sumarle uno le disminuimos uno a la cantidad actual.

* 1. **Se hace la lógica para el botón de “Ultima” el cual va a llevar a la primera página**

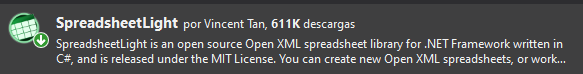


En este lo que se hace es que se iguala la paginación actual con el total de páginas que deberían haber – 1 ya que la paginación empieza desde 0 internamente entonces necesitamos restarle uno para que se muestre realmente la ultima pagina con registros de lo contrario saldría una página vacía.

* 1. **Se agrega el botón de exportar a Excel**



* 1. **Se implementa la librería SpreadsheetLight desde el administrador de paquetes NuGet**



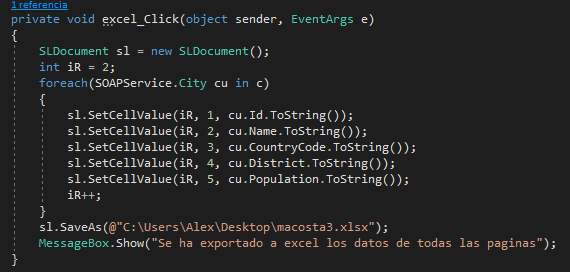
* 1. **Se añade la referencia del ensamblado de DocumentFormat.OpenXml**



* 1. **Se implementa el método para exportar a Excel cuando se de click en el botón “exportar a Excel”**

En esta aplicación yo use dos botones para que el usuario pueda elegir si exportar todos los datos a Excel o solo los datos de la pagina donde se encuentra actualmente situado.

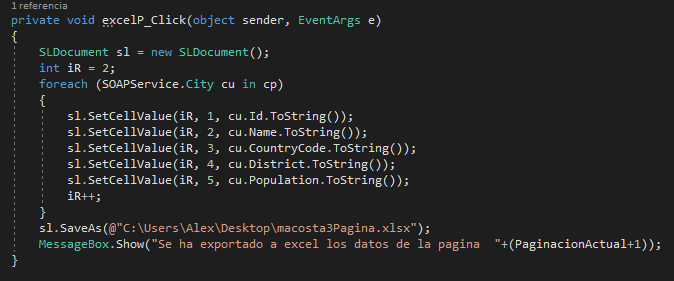
**- Exportar a todos los datos**



A través de la librería que importamos podemos hacer uso de estos métodos para organizar como queremos que se encuentren los datos dentro del documento Excel que exportaremos, como lo las posiciones en x, y.

Al mismo tiempo que se pasa donde debe estar se pasa lo que debe estar ahí ósea el dato.

**- Exportar los datos de la página actual**



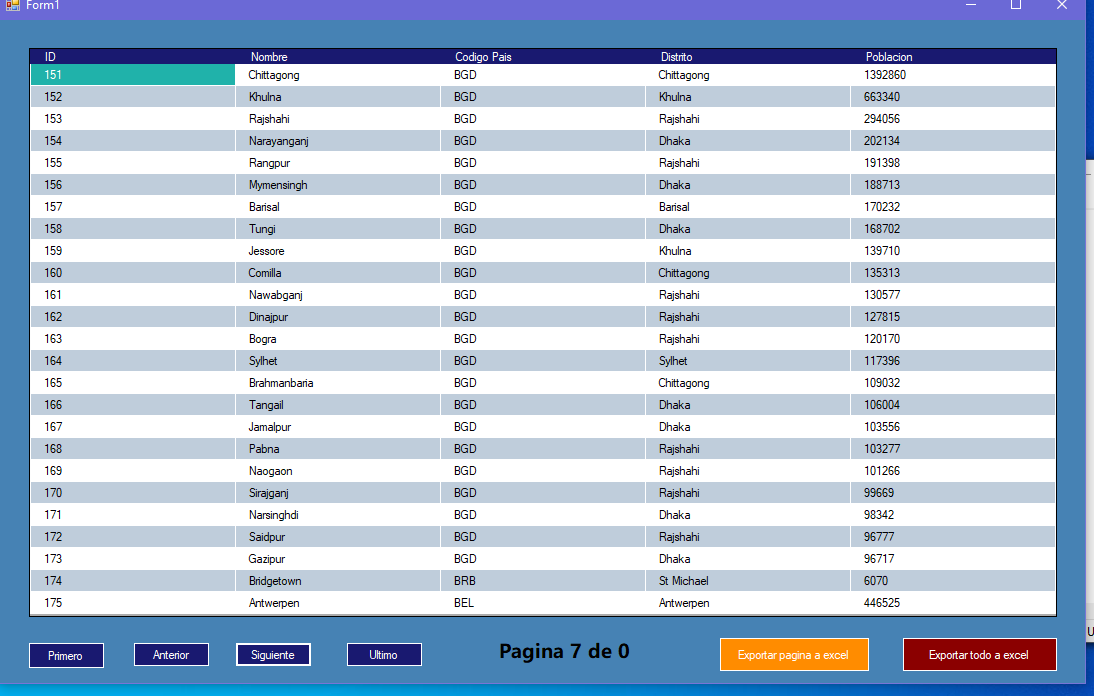
En este lo que hago es hacer uso de la lista de copia para que solo recorra esa y exportar únicamente estos datos.

* 1. **Arreglar la apariencia del formulario**

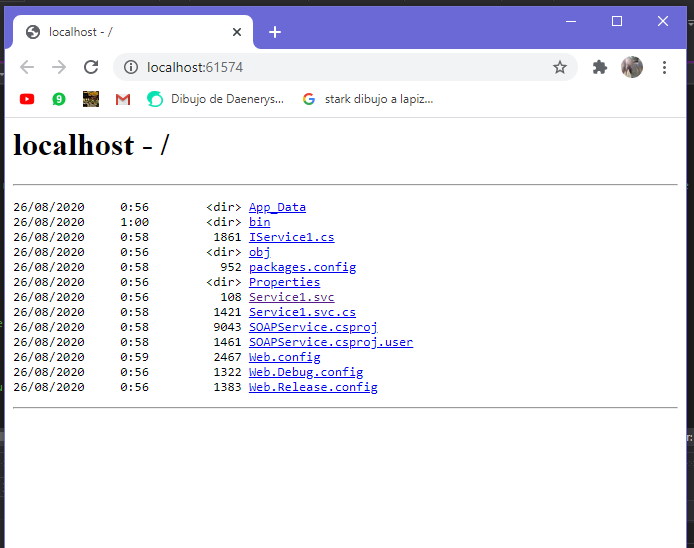


1. **Evidencias de funcionamiento**

**- Cliente**



**- Servidor**



**- Archivos**

