**Proyecto: Campaña de mailing**:

Realizado para: Dirección de Emisiones. GGTI, ARBA.

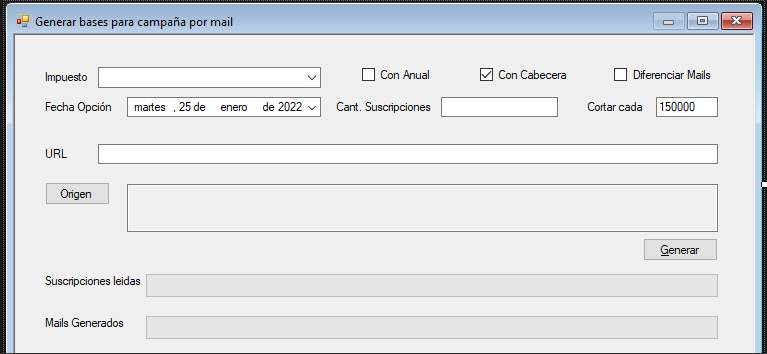
Programadores a cargo: Avendaño Oscar y Nicol Cruz.

Fecha de trabajo: Diciembre 2021 – Enero 2022.

Se realizó un trabajo de *refactoreo*, pasando un aplicativo, originalmente en .NET, (C#) a Lenguaje Java, en el entorno de Netbeans. La Aplicación trabaja con una base de datos en .txt, de la que toma datos y en base a los cuales forma, inyectando HTML, grupos de mails para envío masivo con un spam bot.

* El producto final principal son archivos .csv, con una cantidad indicada por el usuario en el GUI de mails (rows).
* Paralelamente el programa crea un .txt con un informe de cada .csv creado, y los mails que se formaron dentro.
* Por último, se crea también un .zip donde se guardan todos los .csv.

La aplicación (original) cuenta con las siguientes partes:



**Origen (bttn):**

Abre una ventana de selección: (por defecto si no se ha seleccionado un impuesto en el combobox “Impuesto”, este bttn abrirá una ventana de búsqueda en la dirección que se encuentre en “*this.Origen.InitialDirectory*”, en la clase “form2.designer.cs”.)

El archivo seleccionado vuelca toda su ruta y nombre en la variable: ***txtOrigen:***

*txtOrigen = this.Origen.FileName;*

**Combobox –Impuesto-:**

Si ya se ha elegido una de las opciones del Combobox **antes** de apretar el Bttn Origen, esta selección llama a un evento, un módulo del comboBox donde, acorde a la opción elegida forma un *String* con una ruta, ejemplo:

*String directorioOrigen = C:\\Users\\User\\Desktop\\BaseDatos\\*

|  |
| --- |
| ***directorioOrigen como variable está incrustada en el código: Por lo que la aplicación trabaja estrictamente con la ruta para la que fue programada:***@"\\arba.gov.ar\DE\GGTI\Gerencia de Produccion\Mantenimiento\Boleta Electronica\Origen\" |

*y agrega, según el impuesto seleccionado, dentro de su módulo y con un SWITCH-CASE, el impuesto con el que se va a trabajar:*

*directorioOrigen* ***+=*** *@Automotores”*

*quedando, por ejemplo.*

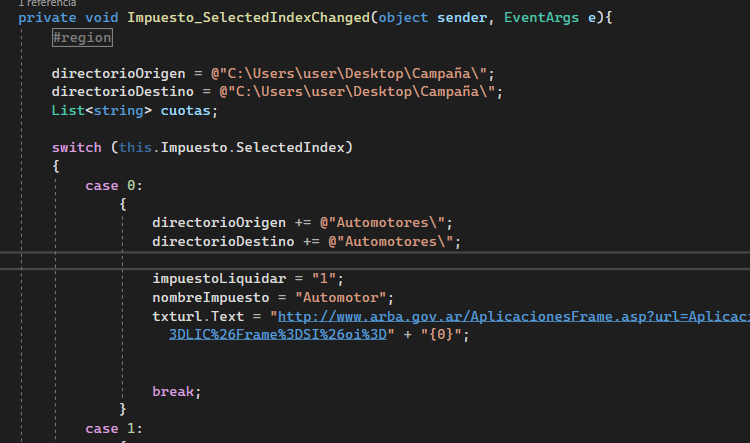
*directorioOrigen = C:\\Users\\User\\Desktop\\BaseDatos\\Automotores*

*Esta ruta se vuelca, dentro de este mismo evento en:*

*this.Origen.InitialDirectory = directorioOrigen;*

Lo cual marca la ruta a partir de la debe abrir el *OpenFileDialog();* al iniciar el evento de bottn Origen.

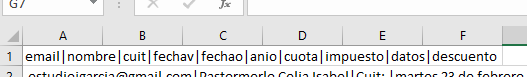
Los impuestos seleccionables pueden ser: Impuesto Automotor, Impuesto a las Embarcaciones, Impuesto Urbano Edificado, Impuesto Urbano Baldío, Impuesto Rural e Impuesto Complementario.



***Con Anual- Con Cabecera - Diferencial Mails - Fecha Opción:***

En el caso de estas tres opciones, lo que se busca es detallar en mayor grado el resultado final del proceso total que arroja la aplicación: Con Anual , agrega el monto anual que extrae de la base de datos. Con Cabecera: agrega una línea al principio de la redacción del .CSV que describe la información que aparecerá debajo:

"email|nombre|cuit|fechav|fechao|anio|cuota|impuesto|datos|descuento"



Fecha Opción: utiliza el *DateTimerPicker* y agrega la fecha del día en que se ejecuta la aplicación al resultado final.

Cant. Suscriptores:

Se sugiere un número que represente la cantidad de líneas a leer dentro del .txt que se toma de base. En caso de no ser igual al número final de líneas leídas, se obtiene un *MessageBox.Show()* que alerta sobre este detalle, además de informar la cantidad en concreto que se llegó a obtener de líneas leídas.

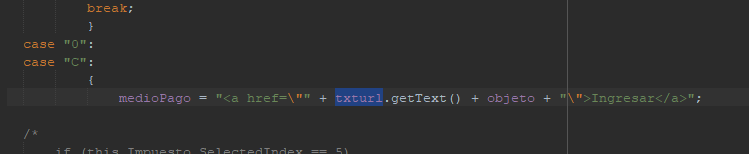
Cortar cada:

Esta opción funciona en conjunto con la anterior. Se establece con esta el número máximo de mails que deben cargarse en un CSV antes de comenzar a guardarlos en un CSV diferente.

La diferencia entre rows leídas en el archivo original y la cantidad de mails que se forman al final del proceso radica en la diferencia entre la cantidad de impuestos a pagar por cada correo o razón social diferente, siendo más de uno en muchos casos, por lo que comúnmente se vería un resultado donde es menor el número de mails al número de rows.

URL:

Toma la Url asociada al impuesto elegido, y la muestra por pantalla. Ofrece la posibilidad de llenar este espacio con una nueva dirección (siendo la que aparece en principio una dirección incrustada en código) que va a ser tomada entonces como la dirección que formará parte del resultado final. En tal caso de que la variable “debitoCredito” fuera “0” o “C”.



Barra: Subscripciones Leídas: toma el valor desde el cuadro de texto “cantidad de subscripciones” y por cada línea leída desde la base de datos, incorpora un mayor número al porcentaje mostrándolo por pantalla.

Barra: Mails Generados: Indica cuantos mail se han generado a partir de la lectura de la base de datos en la función *“Procesar();”*

Generar (Bttn):

funciona Tomando el valor desde "cantidad de subscripciones" en caso de ser nulo envía un mensaje por pantalla- Toma el texto de "Impuesto" y la ruta a través del Objeto 'Origen':

Ejemplo1:……………………………….\\arba.gov.ar\DE\GGTI\GerenciadeProduccion\Mantenimiento\Boleta Electronica\Origen\Edificado\20150519-3-CO.TXT )

Por medio de la sentencia “.Replace(this.Origen.InitialDirectory, "");” se vuelva sobre la variable “TxtDestino” el nombre del archivo seleccionado dando :

Ejemplo2 : "\Edificado\20150519-3-CO.TXT"

Este será el nombre del archivo seleccionado con el que también se va a nombrar al archivo resultante CSV al final.

En este módulo se llama a la función “Procesar()”

*Función Procesar() :*

Se trata de la función principal del programa. Dentro de esta se realiza la lectura de la base de datos elegida, el llamado a los módulos (principales):

- LeerLinea(): Separan cada porción de la línea leída en esa base, que colocan cada dato en una variable diferente:

- ArmarDatosMail(): enbebe de contenido HTML a las distintas variables que surgen del método anterior, para sumarlos a la variable “datosObjeto” ej: *cuotaNumero*:

datosObjeto += string.Format("<td class='amarillo'>Cuota {0}</td>", *cuotaNumero*);

Se declara el String que cargará los nombres del archivo final, sumando Txtdestino y un contador que lo diferenciará del resto :

string nombreArchivoGenerado = string.Format( "{0}-Parte-{1}.csv", txtDestino, cantidadArchivosGenerados);

con este “nombreArchivoGenerado” se crea un objeto de tipo *StreamWriter* llamado *SW,* que se encargará de crear y escribir el contenido que le pase el programa a un CSV.

A continuación se crea el nombre, se instancia el objeto, y se da contenido de cabecera a un archivo de tipo .txt que llevará un informe de todos los CSV creados, y la cantidad de mails dentro de cada uno de ellos. En este se agrega la variable *nombreArchivoGenerado.*

Path = directorioOrigen + "informe.txt";

StreamWriter SWinforme = new StreamWriter(Path);

SWinforme.Write("Se generearon los siguientes archivos:");

SWinforme.WriteLine();

SWinforme.Write(string.Format("Archivo \*\* {0} \*\* ", *nombreArchivoGenerado*));

Se crea el objeto File, de tipo StreamReader:

StreamReader file = new StreamReader(txtOrigen);

El objeto "file", de tipo streamReader, toma "txtOrigen" como argumento.

En la lectura de cada línea, dentro de un loop While, se cotejan las variables de Objetosocial (nombre) y mail. Conforme se traten de dos nombres diferentes, así como de dos correos diferentes, el programa separa en grupos distintos:

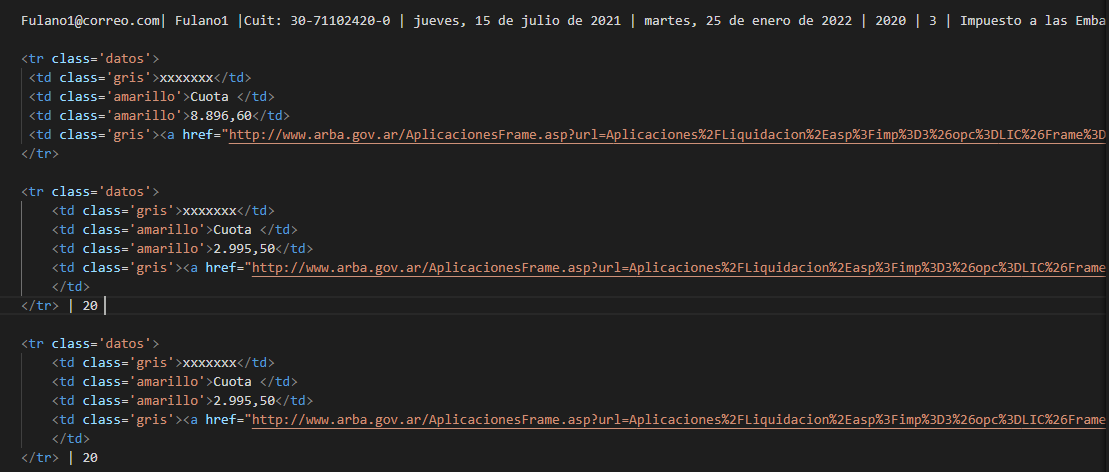
La siguiente Lectura formaría parte de un solo raw dentro del .csv:

Mail: [Fulano1@correo.com](mailto:Fulano1@correo.com) nombre: Fulano1.

Mail: [Fulano1@correo.com](mailto:Fulano1@correo.com) nombre: Fulano1.

Mail: [Fulano1@correo.com](mailto:Fulano1@correo.com) nombre: Fulano1.

Ejemplo:



La siguiente Lectura formaría parte de **dos** raw dentro del .csv:

Mail: Fulano2@correo.com nombre: Fulano1.

Mail: [Fulano1@correo.com](mailto:Fulano1@correo.com) nombre: Fulano1.

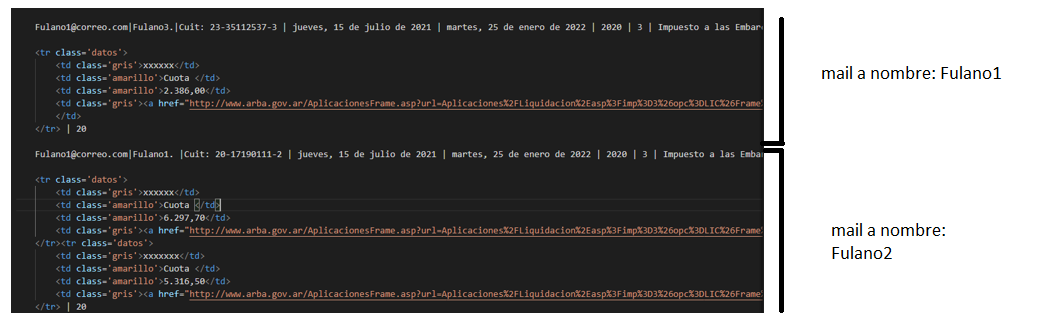
Mail: [Fulano1@correo.com](mailto:Fulano1@correo.com) nombre: Fulano1.

Asi también la siguiente Lectura formaría parte de **dos** raw dentro del .csv:

Mail: Fulano1@correo.com nombre: Fulano3.

Mail: [Fulano1@correo.com](mailto:Fulano1@correo.com) nombre: Fulano1.

Mail: [Fulano1@correo.com](mailto:Fulano1@correo.com) nombre: Fulano1.



El agrupamiento se realiza entre subscriptores que lleven ***objetosocial***y ***mail*** idénticos. En cuanto alguna de estas variables cambia, se acumulan los datos en una variable diferente y se cargan como un mail distinto a enviar dentro de la variable Mail Línea. Esta se completa con los datos:

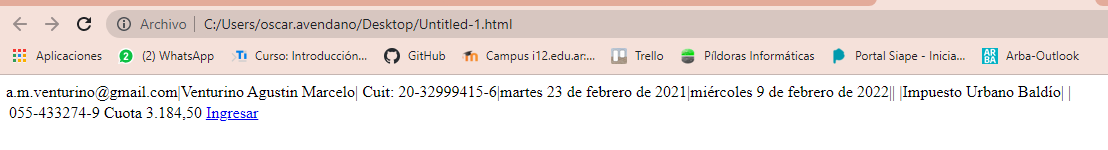
mailLinea += string.Format("{0}|Cuit: {1}|{2}|{3}|{4}|{5}|{6}|{7}|{8}", razonsocialAux, this.formatearCuit(cuitAux), fechaVencimiento, fechaOpcion, anio, cuota, impuesto, datosTodosObjetos, porcentaje);

Habiendo agregando el mail al principio dentro de la propia variable, y los datos embebidos en HTMl del resultado de “datosTodosObjetos” (variable que toma su valor de datosObjeto) esta variable puede verse en el resultado final asi:

a.m.venturino@gmail.com|Venturino Agustin Marcelo|Cuit: 20-32999415-6|martes 23 de febrero de 2021|miércoles 9 de febrero de 2022|||Impuesto Urbano Baldío|<tr class='datos'><td class='gris'>055-433274-9</td><tdclass='amarillo'>Cuota</td><td class='amarillo'>3.184,50</td><tdclass='gris'><ahref="http://www.arba.gov.ar/AplicacionesFrame.asp?url=Aplicaciones%2FLiquidacion%2Easp%3Fimp%3D0%26opc%3DLIC%26Frame%3DSI%26oi%3D0554332749">Ingresar</a></td></tr>|

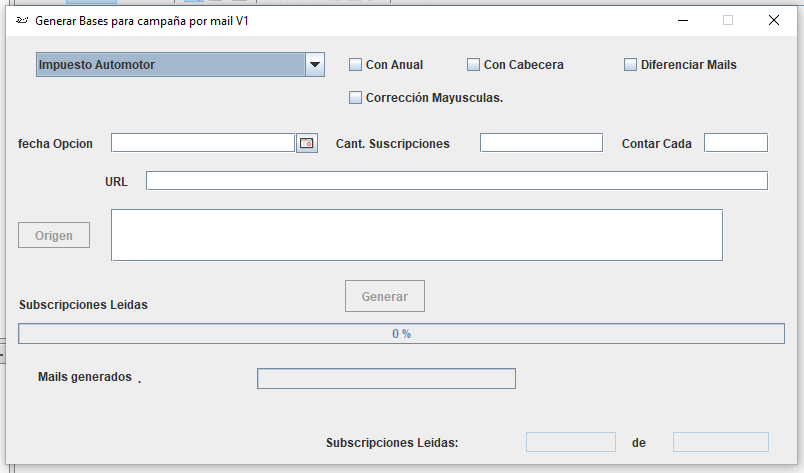
(En celeste el contenido HTML.)

Este tipo de código puede verse en un navegador del siguiente modo:



Al terminar, y tras informar si la cantidad de subscripciones leídas y las propuestas al principio son o no las mismas, el programa consulta si se desea guardar todo lo generado hasta ese momento (los CSV y el informe txt) en un zip. Llamando a la función *InformarArchivosGenerados().*

Programa en Java:



En términos generales las diferentes partes del GUI ( [*graphical user interface*](https://www.google.com/search?rlz=1C1CHBD_esAR981AR981&sxsrf=APq-WBuxYUV7x_wPWCXw0EBk4hC2HCLYYQ:1644413839956&q=graphical+user+interface&spell=1&sa=X&ved=2ahUKEwib9az23vL1AhXjILkGHQcjAgYQkeECKAB6BAgJEDc)) buscaron ser las mismas que la aplicación original, respetando nombres y ubicación a los fines de hacerlo accesible para los operadores ya familiarizados con aquella. Junto con esto la estructura del código interno se respetó en forma y ejecución de la lógica. Reusándose en muchos casos la misma progresión de elementos bajo nombres idénticos de variables y funciones. Todo esto también en búsqueda de hacer su código accesible para quienes ya hubieran operado con el código original en C#.

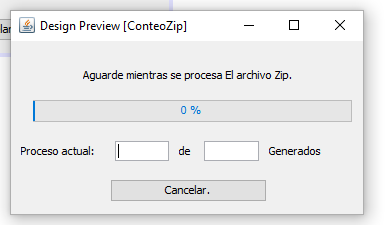
Dentro de algunas de las modificaciones que debieron hacerse para realizar el refactoreo del programa de mailing a java se encuentran:

° Se anexó un contador de líneas de lectura en la base del GUI con intención de llevar un conteo más prolijo del proceso de lectura y armado de los mails.

° El total de la lectura que lleva adelante la barra de “Subscripciones leídas” se define al principio del programa con una lectura que es anterior al trabajo principal del mismo. Esto con intenciones de ofrecer un resultado más exacto a la hora de indicar el porcentaje de proceso realizado, y ofrecer una mejor determinación del tiempo de espera.

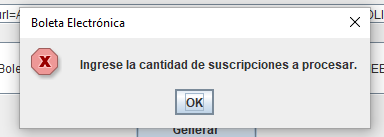
° La corrección de Mayúsculas se puso como una opción en un checkbox, dado en algunos casos la base de datos cuenta con nombres con tipos de caracteres que pueden llegar a generar inconvenientes en la conversión, y esto permite decidir desde el lugar del usuario si se desea intentar esta conversión o no.

° Se agregó una pantalla de carga para el ultimo módulo: *InformarArchivosGenerados().*



Lleva la cuenta de los archivos leídos por este módulo, ofreciendo en pantalla una barra de progreso que puede ser utilizado como indicador de tiempo para el usuario, previendo así cuanto puede llegar a tomarle convertir todo el trabajo a zip.

° El GUI está planeado de forma prohibitiva, indicando pasos a seguir de manera tal que no se puede acceder a los procesos internos del programa a menos que se hayan cargado todos los datos necesarios antes.



° En este mismo orden, una vez iniciado el proceso dejan de ser accesibles los botones de Origen y Generar.

° las direcciones de Origen y Destino fueron modificadas con respecto al programa original. Esto debido en parte a que .NET opera con una carpeta de guardado por defecto. En el caso de Java, ambas direcciones deben ajustarse en HardCode, acordando donde estarán las carpetas con nombres de los diferentes impuestos que deben buscarse. Así como ubicando una sección con carpetas de igual nombre donde se envíen los resultados del procesamiento.