**WebMarketingV5.0**

1. **Introducción**

Esta es una pequeña introducción a la aplicación de correo masivo su funcionalidad principal es él envió de correo masivo, esta aplicación tiene una mala estructuración en su arquitectura, esta es la forma como trabaja la aplicación a nivel de base de datos, actualmente la aplicación tiene la siguiente estructura de base de datos, se explicara en general la forma como realiza un envió la aplicación:

Las tablas maestras para él envió de un boletín son:

- **app\_Boletines**: en esta tabla se configuran los parámetros Generales del envió, un boletín solo puede tener un envío de correo masivo.

- **ListadosUsuarios**: estos son los usuario a los cuales se les realizara él envió de correo , estos listado puede ser cargados a esta tabla por la aplicación de diferentes fuentes como son, **Vistas, Importado Excel, Manual, Importado CSV, Formulario Dinámico, Procedimiento Almacenado, Búsqueda Usuarios**, más adelante se explicara cada una de estas funcionalidades con mayor detalle.

- **app\_BoletinesListados**: En esta tabla se relacionan los listados de usuarios a los que se le enviara el boletín configurado.

- **app\_Boletines\_Estados**: En esta tabla se encuentran los diferentes estados del boletín.

- **SmtpServers**: Estos son los diferentes smtp por los cuales se despacharan los correos del envío.

- **Usuarios**: Esta es la tabla en la cual se copiaran los usuarios a los que se le enviaran los correos del envío, esta es como la cola de despacho del envío, en esta tabla según la configuración realizada al boletín se copian el número de correo para enviar el correo uno a uno de los listados configurados para el envío del boletín.

- **app\_Boletines\_Emisiones**: Un boletín es programado para que se ejecute a una hora y fecha determinada, cuando se llega esta hora, y el boletín está aprobado, se realiza el registro de la emisión en esta tabla, un boletín solo tiene una emisión por envío, en esta se registra el inicio y el fin de la emisión del boletín.

- **app\_Boletines\_Emisiones\_Suscriptores**: en esta tabla se realiza el registro de los usuario a los cuales se les ha realizado el envío o no, esta registra de nuevo todos los datos del usuario, al pasar los usuarios a esta tabla se eliminan de la tabla **Usuarios (Tabla cola del envío)** registrando diferentes datos dependiendo de la funcionalidad programada para el boletín.

Esta son las tablas Generales sin entrar en detalle de las múltiples funcionalidades y configuraciones que se pueden realizar a un envío este es un diagrama General del flujo realizado en el envío por tablas:

**app\_Boletines**

**app\_BoletinesListados**

**Al cumplir el criterio de ejecución del boletín por fecha/hora se realiza el registro de la emisión en la tabla app\_Boletines\_Emisiones**

**Se copian de la tabla listadosUsuarios, los usuarios basados en la relación con todos sus datos una vez se inicia la emisión del boletín por un top determinado en los parámetros del boletín.**

**Usuarios**

**Se realice en el envío exitoso o no del correo al usuario, se elimina el registro de la tabla de usuarios y se registra en la tabla app\_Boletines\_Emisiones\_Suscriptores**

Esta forma de envío masivo tiene un problema Gigante ya que repite los datos de la tabla de usuarios a la tabla de envío completa, esta es una de oportunidad de mejora la cual deben de replantear esta estructura principal de envío de correo masivo, ya que incluso se ha ido adaptando a los requerimientos pedidos pero esta forma de trabajo tiene un gran problema debido al crecimiento de datos en la tabla final de la emisión, cada envío es como si creara de nuevo todos los usuarios de los listados seleccionados por envío en la tabla final, el mayor reto que tendrán no será las diferentes funcionalidades que realiza la aplicación, si no el trabajar con grandes cantidades de datos, ya que así se replantee la base de datos en su arquitectura, deben tener en cuenta que tendrán grandes cantidades de datos por envío, esto lo deben de tener en cuenta sobre todo a la hora de realizar validaciones y mostrar informes y resultados de los envíos desde el planteamiento de los query, hasta el cargar los datos por medio de json y Ajax(**Deben tener en cuenta en los módulos donde se mostraran grandes cantidades de registros paginar estos registros desde base de datos**: Hay un límite en el tamaño del objeto json? SOLO DEPENDE DEL METODO QUE USES PARA ENVIAR LOS DATOS, SI USAS GET EL LIMIETE ES **2048** caracteres SI USAS POST EL LIMITE NO EXISTE YA QUE ES ES UN TYPO MULTI TAMAÑO), deben tener en cuenta la paginación de base de datos a la hora de mostrar datos y realizar informes de esto, ya que la aplicación se verá enfrentada a grandes cantidades de datos, esto lo deben tener muy en cuenta y es lo principal, las funcionalidades serán explicadas cada una pero deben tener muy en cuenta que la aplicación trabajara con grandes cantidades de registros, esta aplicación a tocado incluso para la generación de informes, crear tareas por hilos de ejecución las cuales por debajo registran y actualizan constantemente los informes de cantidades y cálculos de envíos ya que a la hora de realizar los cálculos, porcentajes, gráficos, etc. contra múltiples envíos esto tumbara la aplicación o su base de datos, tiene un gran reto , la prueba final que le deben hacer a esta aplicación es que cuando este cargada con millones de registros, todos sus módulos y funcionalidades programadas ejecuten correctamente, siempre deben tener un log el cual registre cada actividad y proceso ejecutado en el envío esto es más que necesario.

1. **Funcionalidades SMTP**

Se explicara las diferentes funcionalidades realizadas y relacionadas con el envío de correo masivo.

* **Envió de Correo**:

La funcionalidad principal que es enviar un correo, la función de envío de correo masivo tiene diferentes funcionalidades internas (Métodos), las cuales pueden separadas en funciones y métodos como oportunidad de mejora, estas son las funcionalidades presentadas en esta función principal de envío:

\***Chilkat**

**MailMan mailMan = new MailMan();**

**mailMan.UnlockComponent("ESTRATMAILQ\_6bcNk7U06R3K");**

Este componente se usa diferentes funcionalidades, la empresa adquirió el SMTP y el POP3, en el envío se utiliza el SMTP del componente la documentación del componente y su uso se encuentra en

<https://www.chilkatsoft.com/>

En esta página esta la documentación para .net, este tiene diferentes funcionalidades cada una de ellas requiere licencia individual a no ser de que se adquiera la licencia full del componente lo cual no fue necesario.

La función de envío recibe diferentes parámetros uno de ellos es el html del correo con este se realiza diferentes funcionalidades internas las cuales las puede separar en funciones de utilidades como oportunidad de mejora y reutilización de código, estas funciones son las siguientes:

* **Manejo de imágenes Incrustadas en el cuerpo del correo**:

HtmlDocument doc = new HtmlDocument();

doc.LoadHtml(BoletinHTML);

var nodes = doc.DocumentNode.SelectNodes("//img");

if (nodes != null)

foreach (var node in nodes)

{

if (node.Attributes.Contains("src"))

{

string src = node.GetAttributeValue("src", "");

//System.Web.VirtualPathUtility.GetDirectory

src = GetRelativeURL(src);

bool directorioVitual = Convert.ToBoolean(ConfigurationManager.AppSettings["DirectorioVirtual"]);

if (directorioVitual)

{

int[] Index = IndexOfAll(src, "/");

string remove = src.Substring(Index[0], Index[1]);

src = src.Replace(remove, "..");

}

string app = Global.ApplicationPath.Substring(0, Global.ApplicationPath.Length - 1);

string srcFileOnDisk = app + src.TrimStart(".".ToCharArray()).Replace("/", "\\").Replace(@"\\", "\\");

string cid = email.AddRelatedFile(srcFileOnDisk);

node.Attributes["src"].Value = "cid:" + cid;

}

}

BoletinHTML = doc.DocumentNode.OuterHtml;

Esta funcionalidad permite insertar una imagen en el cuerpo del correo, convirtiendo las rutas relativas de la imagen en rutas absolutas con la función **GetRelativeURL,** reemplazando este en cada tag **Img** de las imágenes en su ruta en el atributo **Src** del elemento html.

* **TrackedLinks**:

Esta es una funcionalidad la cual se explicara desde su inicio hasta que llega a esta parte de la función de envío de correo, el track link registra los clic en cada link en el cuerpo del boletín si es configurado, este registra el correo del usuario, el boletín al que pertenece y fecha/hora del clic, también realiza un contador de los clic dados en el link, el proceso con el seguimiento de *l*ink es el siguiente:

Todo boletín tiene un cuerpo html en el cual irán todas las etiquetas de los elementos html tipo link que tenga este, para realizar la búsqueda de esta etiquetas se usa un componente llama **HtmlAgilityPack** esta línea de código realiza la búsqueda de esta

**HtmlNodeCollection nodes = doc.DocumentNode.SelectNodes("//a[@href]");**

Luego se recorren estas etiquetas y si no se encuentran en la tabla **app\_TrackedLinks** se crean si se ha configurado para que realice el boletín seguimientos a los enlaces

if (nodes != null)

foreach (HtmlNode img in nodes)

{

string href = img.GetAttributeValue("href", "");

string desc = href.Replace("http://", " ");

DBDataHelper dbDHTraked = new DBDataHelper(@"SELECT \* FROM app\_TrackedLinks WHERE BoletinID = @BoletinID AND LinkDesc = @LinkDesc");

dbDHTraked.AddParameter("@BoletinID", boletinID);

dbDHTraked.AddParameter("@LinkDesc", desc.Replace("&amp;", "&"));

DataTable DtdbDHTraked = dbDHTraked.ExecuteDataTable();

if (DtdbDHTraked.Rows.Count < 1)

{

DBDataHelper dbDHTrakedInsert = new DBDataHelper(@"INSERT app\_TrackedLinks (BoletinID,LinkDesc,LinkedURL,LinkClickCount) VALUES (@BoletinID,@LinkDesc,@LinkedURL,0)");

dbDHTrakedInsert.AddParameter("@BoletinID", boletinID);

dbDHTrakedInsert.AddParameter("@LinkedURL", href.Replace("&amp;", "&"));

dbDHTrakedInsert.AddParameter("@LinkDesc", desc.Replace("&amp;", "&"));

dbDHTrakedInsert.ExecuteNonQuery();

}

}

Esto enlaces creados se reutilizaran para el registro de los clic sobre cada enlace por usuario, una vez se realiza esto se toma este html en la función de envió SendEmail esta tiene la siguiente funcionalidad interna

if (Convert.ToBoolean(TrackedLinks))

{

HtmlDocument docLinked = new HtmlDocument();

docLinked.LoadHtml(BoletinHTML);

DBDataHelper dbDHTraked = new DBDataHelper(@"SELECT \* FROM app\_TrackedLinks WHERE BoletinID = @BoletinID ");

dbDHTraked.AddParameter("@BoletinID", BoletinID);

DataTable DtdbDHTraked = dbDHTraked.ExecuteDataTable();

foreach (DataRow rowTrackedLinks in DtdbDHTraked.Rows)

{

DataRow links = rowTrackedLinks;

var hrefs = docLinked.DocumentNode.SelectNodes("//a[@href]");

foreach (var htmlNode in hrefs)

{

string href = htmlNode.GetAttributeValue("href", "").Replace("&amp;", "&");

string notracklink = htmlNode.GetAttributeValue("class", "");

if (href.Equals(links["LinkedURL"]) && !notracklink.Equals("nolinktrack"))

{

string url = ConfigurationManager.AppSettings["SiteRoot"] + "redir.aspx" +

"?Email=" + to + "&Url=" + links["LinkedURL"] + "&LinkID=" + links["LinkID"];

htmlNode.Attributes["href"].Value = url;

}

}

}

BoletinHTML = docLinked.DocumentNode.OuterHtml;

Cuando se desea que el link no sea afectado por el tracklink se debe agregar la clase **'nolinktrack'** al enlace

if (href.Equals(links["LinkedURL"]) && !notracklink.Equals("nolinktrack"))

{

string url = ConfigurationManager.AppSettings["SiteRoot"] + "**redir.aspx**" +

"?Email=" + to + "&Url=" + links["LinkedURL"] + "&LinkID=" + links["LinkID"];

htmlNode.Attributes["href"].Value = url;

}

Esta funcionalidad remplaza la url de cada link para que cuando el usuario le dé clic antes de ir a su destino pase por una página interna de la aplicación llama redir.aspx, esta página antes de redirigir el usuario realiza el registro del clic capturando los datos del usuario por medio del link y parámetros get en la url

**ViewState["Email"] = Request.QueryString["Email"] ?? string.Empty;**

**ViewState["LinkID"] = Request.QueryString["LinkID"] ?? string.Empty;**

**if (!string.IsNullOrEmpty(ViewState["LinkID"].ToString()) && !string.IsNullOrEmpty(ViewState["Email"].ToString()))**

**{**

**DBDataHelper dbDHTrakedCount = new DBDataHelper(@"SELECT \* FROM app\_TrackedLinks WHERE LinkID = @LinkID");**

**dbDHTrakedCount.AddParameter("@LinkID", ViewState["LinkID"].ToString());**

**DataTable DtdbDHTrakedCount = dbDHTrakedCount.ExecuteDataTable();**

**if (DtdbDHTrakedCount.Rows.Count > 0)**

**{**

**DBDataHelper dbDHTrakedBoletin =**

**new DBDataHelper(@"SELECT \* FROM app\_Boletines WHERE ID = @ID");**

**dbDHTrakedBoletin.AddParameter("@ID", DtdbDHTrakedCount.Rows[0]["BoletinID"]);**

**DataTable DtdbDHTrakedBoletin = dbDHTrakedBoletin.ExecuteDataTable();**

**if (DtdbDHTrakedBoletin.Rows.Count > 0)**

**{**

**int trakedlink = (!string.IsNullOrEmpty(DtdbDHTrakedBoletin.Rows[0]["LinkClickCount"].ToString())) ? Convert.ToInt32(DtdbDHTrakedBoletin.Rows[0]["LinkClickCount"]) : 0;**

**DBDataHelper dbDHTrakedBoletinCount =**

**new DBDataHelper(@"UPDATE app\_Boletines SET LinkClickCount = @LinkClickCount WHERE ID = @ID");**

**dbDHTrakedBoletinCount.AddParameter("@ID", DtdbDHTrakedCount.Rows[0]["BoletinID"]);**

**dbDHTrakedBoletinCount.AddParameter("@LinkClickCount", trakedlink + 1);**

**dbDHTrakedBoletinCount.ExecuteNonQuery();**

**//**

**DBDataHelper dbDHTraked =**

**new DBDataHelper(**

**@"UPDATE app\_TrackedLinks SET LinkClickCount = @LinkClickCount WHERE LinkID = @LinkID");**

**dbDHTraked.AddParameter("@LinkClickCount",**

**Convert.ToInt32(DtdbDHTrakedCount.Rows[0]["LinkClickCount"]) + 1);**

**dbDHTraked.AddParameter("@LinkID", ViewState["LinkID"].ToString());**

**dbDHTraked.ExecuteNonQuery();**

**//**

**DBDataHelper dbDHTrakedStat =**

**new DBDataHelper(@"SELECT \* FROM app\_TrackedLinksStat WHERE LinkID = @LinkID AND Email = @Email");**

**dbDHTrakedStat.AddParameter("@LinkID", ViewState["LinkID"].ToString());**

**dbDHTrakedStat.AddParameter("@Email", ViewState["Email"].ToString());**

**DataTable DtdbDHTrakedStat = dbDHTrakedStat.ExecuteDataTable();**

**if (DtdbDHTrakedStat.Rows.Count < 1)**

**{**

**DBDataHelper dbDHTrakedStatInsert =**

**new DBDataHelper(**

**@"INSERT app\_TrackedLinksStat (LinkID,Email,DateTimeLink,LinkClickCountDet) VALUES (@LinkID,@Email,@DateTime,1)");**

**dbDHTrakedStatInsert.AddParameter("@LinkID", ViewState["LinkID"].ToString());**

**dbDHTrakedStatInsert.AddParameter("@Email", ViewState["Email"].ToString());**

**dbDHTrakedStatInsert.AddParameter("@DateTime",**

**DateTime.Now.AddHours(Convert.ToInt32(Global.ConfiguracionHora\_Global)));**

**dbDHTrakedStatInsert.ExecuteDataTable();**

**}**

**else**

**{**

**DBDataHelper dbDHTrakedStatClick =**

**new DBDataHelper(**

**@"UPDATE app\_TrackedLinksStat SET LinkClickCountDet = @LinkClickCountDet WHERE LinkID = @LinkID AND Email = @Email");**

**dbDHTrakedStatClick.AddParameter("@LinkID", ViewState["LinkID"].ToString());**

**dbDHTrakedStatClick.AddParameter("@Email", ViewState["Email"].ToString());**

**dbDHTrakedStatClick.AddParameter("@LinkClickCountDet",**

**Convert.ToInt32(DtdbDHTrakedStat.Rows[0]["LinkClickCountDet"]) +**

**1);**

**dbDHTrakedStatClick.ExecuteNonQuery();**

**}**

**}**

**Response.Redirect(DtdbDHTrakedCount.Rows[0]["LinkedURL"].ToString());**

**-Adjuntos**

Esta versión de la aplicación solo maneja adjuntos por boletín en general para todo el usuario en general del envió

string path = Global.ApplicationPath + "App\_Upload\\";

DirectoryInfo AttachMentds = new DirectoryInfo(path);

if (AttachMentds.Exists)

{

FileInfo[] files = AttachMentds.GetFiles(BoletinID + "\_" + "\*.\*", SearchOption.AllDirectories);

if (files.Length > 0)

{

foreach (FileInfo file in files)

{

email.AddFileAttachment(file.FullName);

}

}

}

* **Chilkat Key Smtp**

El chikcat en la función de envió se utiliza una propiedad del componente en su funcionalidad de smtp con el cual en el encabezado del cuerpo del mensaje, se insertan llaves las cuales contienen datos tipo string

**email.AddHeaderField("X-EmisionxSuscriptorID", emisionxSuscriptorID.ToString());**

En este caso se envía el Id de la emisión marcando así el correo para procesos como son la lectura del correo, rebotes de boletines, confirmación de email, entre otras funcionalidades que se explicaran mas adelante.

* **Tags personalizados**

Esta es una utilidad del envió simple en la cual en un primer paso se pueden crear múltiples tags para la aplicación, los cuales serán reemplazados por el texto que se desee reemplazar en el cuerpo del boletín, la aplicación tiene algunos tags predefinidos estáticos los cuales son:

|  |  |
| --- | --- |
| **[@correoconfirmacionsimple@]** | **Inserta un link para confirmación del usuario al correo.** |
| **[@correoconfirmacion@]** | **Inserta un link para confirmación del correo.** |
| **[@redessociales@]** | **Inserta un link para compartir el boletín en facebbok y twitter.** |
| **[@cancel@]** | **Inserta un link para cancelar la suscrición al boletín.** |
| **[@view@]** | **Inserta un link para visualizar el boletín en una página fuera del correo.** |
| **[@Cedula@]** | **Cedula del suscriptor.** |
| **[@Nombre@]** | **Nombre del suscriptor.** |
| **[@Email@]** | **Email del Suscriptor.** |
| **[@Cp1@] ... [@Cp10@]** | **Campos Personalizados del 1 al 10. [@Cp1@] => campo personalizado 1,**  **[@Cp2] => campo personalizado 2, etc.** |
| **[@...@]** | **Campos personalizados 1 a 10 si ...**  **es igual que la descripción de campo personalizado.**  **De lo contrario,**  **el texto se colocará en el correo electrónico sin ninguna modificación** |

También se puede crear tags personalizados por ejemplo

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | |  | **Texto** | **Alias** |  | |  | Señor Asociado(a) | [@TratamientoAsociado@] |  | |  | Estimado Usuario(a) | [@TratamientoUsuario@] |  | |  | Respetado(a) Doctor(a) | [@Doctor(a)@] |  | |

A nivel de código lo que se realiza es un simple reemplazar por medi de un diccionario de datos

private string GetFormattedHTML(DataRow row, string html, string emisionxSuscriptorID)

{

Dictionary<string, string> diccionarioVariables = new Dictionary<string, string>();

diccionarioVariables.Add("[@Cedula@]", (row["Cedula"] != null) ? row["Cedula"].ToString() : string.Empty);

diccionarioVariables.Add("[@Email@]", row["Email"].ToString());

diccionarioVariables.Add("[@Nombre@]", (row["Nombre"] != null) ? row["Nombre"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp1@]", (row["Cp1"] != null) ? row["Cp1"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp2@]", (row["Cp2"] != null) ? row["Cp2"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp3@]", (row["Cp3"] != null) ? row["Cp3"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp4@]", (row["Cp4"] != null) ? row["Cp4"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp5@]", (row["Cp5"] != null) ? row["Cp5"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp6@]", (row["Cp6"] != null) ? row["Cp6"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp7@]", (row["Cp7"] != null) ? row["Cp7"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp8@]", (row["Cp8"] != null) ? row["Cp8"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp9@]", (row["Cp9"] != null) ? row["Cp9"].ToString() : string.Empty);

diccionarioVariables.Add("[@Cp10@]", (row["Cp10"] != null) ? row["Cp10"].ToString() : string.Empty);

if (html.Contains("[@redessociales@]"))

{

diccionarioVariables.Add("[@redessociales@]", string.Format(@"<table border='0' cellspacing='0' cellpadding='6' align='center'><td><p>Comparta este bolet&iacute;n en: </font></p></td>

<td><a class='nolinktrack' href='{0}' target='\_blank'><img src='{2}' alt='facebook' title='facebook'

alt='Facebook' title='Facebook' width='30' height='30' border='0' /></a></td>

<td><a class='nolinktrack' href='{1}' target='\_blank'><img src='{3}' alt='Twitter' title='Twitter' width='30' height='30' border='0' /></a></td></table>", "http://www.facebook.com/sharer.php?u=" + ConfigurationManager.AppSettings["SiteRoot"] + "ViewNewsLestter.aspx" + "?BoletinID=" + BoletinID, "http://twitter.com/home?status=" + Boletin.Titulo + " - " + ConfigurationManager.AppSettings["SiteRoot"] + "ViewNewsLestter.aspx" + "?BoletinID=" + BoletinID, ConfigurationManager.AppSettings["SiteRoot"] + "Images/facebook.jpg", ConfigurationManager.AppSettings["SiteRoot"] + "Images/twitter.jpg"));

}

if (html.Contains("[@correoconfirmacion@]"))

{

StringCrypto.CryptoResult tokenEmail = StringCrypto.Encrypt(row["Email"].ToString());

StringCrypto.CryptoResult tokenEmisionID = StringCrypto.Encrypt(EmisionID.ToString());

string url = ConfigurationManager.AppSettings["SiteRoot"] + "ConfirmEmail.aspx?Em=" + tokenEmail.Value + "&Id=" + tokenEmisionID.Value;

diccionarioVariables.Add("[@correoconfirmacion@]", string.Format(

"<a href='{0}' class='nolinktrack' >" + Global.Confirmacion\_Global + "</a>.", url));

}

if (html.Contains("[@correoconfirmacionsimple@]"))

{

StringCrypto.CryptoResult tokenSuscriptorID = StringCrypto.Encrypt(emisionxSuscriptorID);

diccionarioVariables.Add("[@correoconfirmacionsimple@]", string.Format("<a href='{0}' class='nolinktrack'>" + Boletin.MessageConfirm + "</a>.", ConfigurationManager.AppSettings["SiteRoot"] + "ConfirmSimple.aspx" + "?EmisionxSuscriptorID=" + tokenSuscriptorID.Value));

}

DBDataHelper dbDHFields = new DBDataHelper(@"SELECT Value,KeyName FROM app\_Tags");

DataTable dt = dbDHFields.ExecuteDataTable();

foreach (DataRow diccionarioRow in dt.Rows)

{

diccionarioVariables.Add("[@" + diccionarioRow["KeyName"] + "@]", diccionarioRow["Value"].ToString());

}

html = html.Replace("&amp;", "&");

return diccionarioVariables.Aggregate(html, (current, variable) => current.Replace(variable.Key, variable.Value));

}

* **Lectura de correo**

Para la lectura de un correo enviado, este se marca con una imagen oculta en el cuerpo del correo enviado

**BoletinHTML += string.Format("<img src='{0}' style='display:none;visibility:hidden;display:none !important;'/>", ConfigurationManager.AppSettings["SiteRoot"] + "OpenEmail.aspx" + "?EmisionxSuscriptorID=" +**

**emisionxSuscriptorID);**

Esta es insertada en el cuerpo del correo con un estilo oculto, esto ha tenido un problema, que este estilo en gmail, no lo toma, lo puede poner como una oportunidad de mejora para la futura aplicación, esta imagen lo que genera es una petición a una página interna de la aplicación llamada **OpenEmail.aspx,** todos estos parámetros pasados por get se pueden encriptar también como oportunidad de mejora a la seguridad de la aplicación.

* **BlackList**

Esta funcionalidad permite realizar la excepción de enviar un correo a un usuario que este registrado en la tabla **app\_BlackList** ,esta se puede cargar de forma individual o de forma masiva por medio de un archivo de Excel, una vez el correo se encuentra registrado en esta tabla en el envió si se encuentra el correo en esa tabla se realizara la excepción de envió por correo en lista negra.

**if (EmailBlackList(mail))**

**{**

**ewc.LogMessage(Constants.Email\_BlackList);**

**continue;**

**}**

* **Excepciones de Envió**

En un envió se pueden presentar multiples excepciones diferentes a las de un error de envio como son:

* Email\_Empty = "El usuario no tiene e-mail.";
* Email\_Syntax\_Invalid = "Sintaxis del e-mail no es válida.";
* Email\_Duplicate = "Email duplicado.";
* Server\_Limit = "Servidores de envíos no disponibles.";
* Email\_Cancel = "Suscripción Cancelada.";
* Email\_BlackList = "Email en lista negra.";

Las siguientes excepciones se configuran en la aplicación para que después de dos rebotes al mismo correo, este queda inhabilitado para envíos masivos futuros

* Email\_DisableHardBounce = "Correo Inhabilitado por rebotes fuertes.";
* Email\_DisableSoftBounce = "Correo Inhabilitado por rebotes suaves.";
* Email\_DeleteHardBounce = "Correo Eliminado por rebotes fuertes.";
* Email\_DeleteSoftBounce = "Correo Eliminado por rebotes suaves.";
* Email\_DisableExternalSoftBounce = "Correo Inhabilitado por rebotes suaves origen externo.";
* Email\_DisableExternalHardBounce = "Correo Inhabilitado por rebotes fuertes origen externo.";
* **CustomsFields(Campos Personalizados)**

Estos son simples campos los cuales una vez configurados en la aplicación serán reemplazados por el valor configurado en la tabla

**app\_CustomsFields**

**Ejemplo:**

**Descripción de campos Personalizados**

|  |  |
| --- | --- |
| Cp1 | Telefono |
| Cp2 | Celular |
| Cp3 | Dirección Residencia |
| Cp4 | Tiene Hijos |
| Cp5 | Cargo / Ocupación |
| Cp6 | Profesión / Ocupación |
| Cp7 | Estado Civil |
| Cp8 | Mesenger / Facebook |
| Cp9 | Nombres (Campo repetido) |
| Cp10 | Apellidos |

* **Cancelaciones**

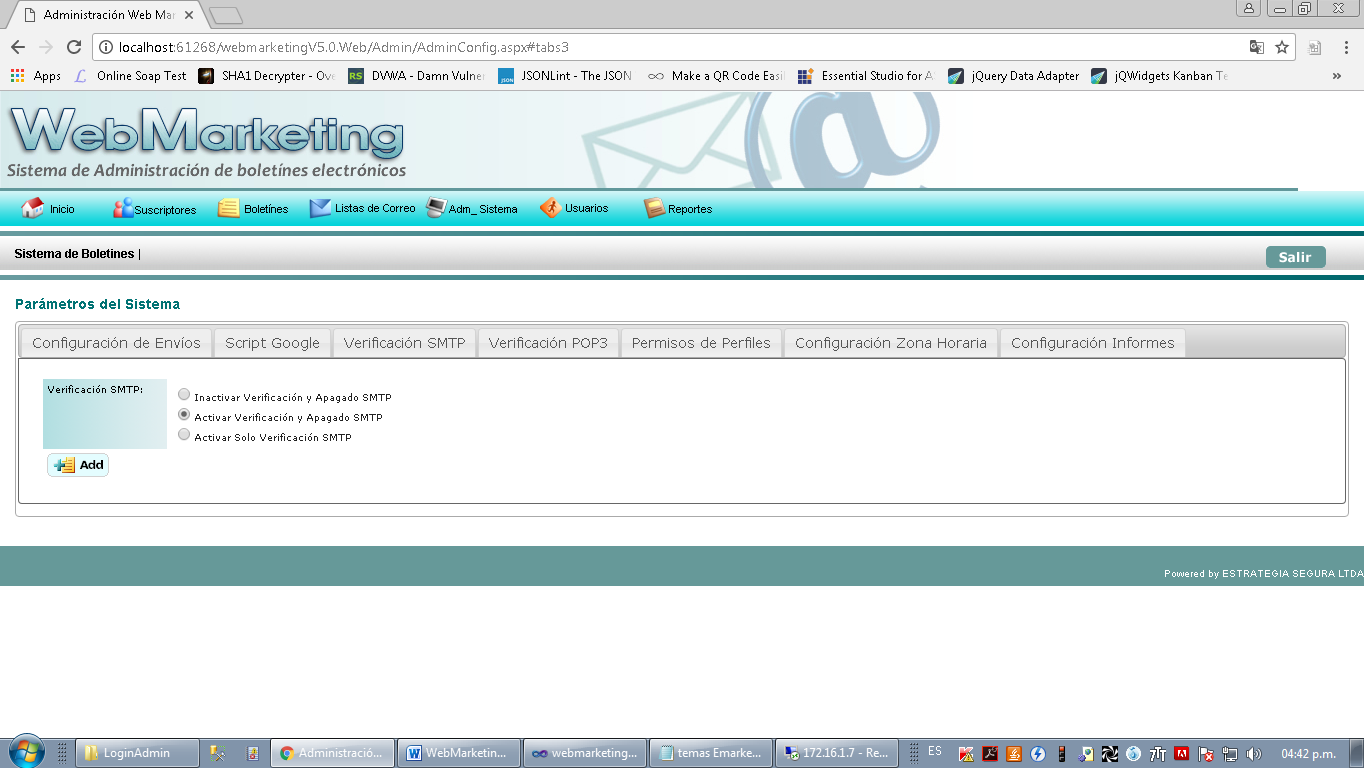
Las cancelaciones se realizan mediante un tag que cuando el usuario da clic desde su correo abierto en este link, lo redirige para una página interna de la aplicación llamada **unsubscribe.aspx**, este realiza un registro de la cancelación para el correo, si este fue cancelado, en un próximo envió este no le enviara correo registrándolo como una excepción de envió por cancelación.

* **Templates**

Los templates, son plantillas html las cuales puede ser reutilizadas en el cuerpo del boletín, se puede seleccionar en un lista de plantillas prediseñadas en la aplicación y pegar este código como una plantilla en el editor de html de la creación o edición del boletin.

* **Multiples Smtp y Pop3**

Para la aplicación se pueden crear múltiples smtp y múltiples pop3, con los múltiples smtp en un envió se distribuye el envió de correo entre los smtp que se encuentran activos un smtp se puede habilitar desde la parte de su configuración o por medio de un proceso interno el cual tiene la aplicación tanto para los smtp como para los pop3 en la administración de los parámetros de configuración



Esta configuración de verificación se puede aplicar tanto al smtp como a los pop3 a nivel de código la validación de pop3 y de smtp se realiza por medio del componente chilkat, esta validación se realiza automática por medio de hilo o tiene páginas para verificar el funcionamiento de cada smtp y de cada pop3.

A nivel de código la distribución de carga de envió y disponibilidad se realiza de la siguiente manera:

**private DataTable GetSmtpServers()**

**{**

**DBDataHelper dbDH1 = new DBDataHelper(@"SELECT ID, IP, Port, Name, Email, Password, Ssl, EnviosxDia, Tls**

**FROM SmtpServers WHERE Estado = 'True' ORDER BY ID");**

**return dbDH1.ExecuteDataTable();**

**}**

**private DataTable GetSmtpServerxDay()**

**{**

**DateTime date = Convert.ToDateTime(DateTime.Now.AddHours(Convert.ToInt32(Global.ConfiguracionHora\_Global)));**

**foreach (var Row in GetSmtpServers().AsEnumerable())**

**{**

**DBDataHelper dbDHServerDay = new DBDataHelper(@"SELECT \***

**FROM SmtpServersEnvioxDia WHERE SmtpServerId = @SmtpServerId AND convert(varchar, FechaEnvio , 11) = convert(varchar,@DateTime, 11)");**

**dbDHServerDay.AddParameter("@SmtpServerId", (int)Row["ID"]);**

**dbDHServerDay.AddParameter("@DateTime", Convert.ToDateTime(DateTime.Now.AddHours(Convert.ToInt32(Global.ConfiguracionHora\_Global))));**

**var Result = dbDHServerDay.ExecuteDataTable().Rows.Count;**

**if (Result >= 1) continue;**

**try**

**{**

**DBDataHelper dbDHInsert = new DBDataHelper(@"INSERT INTO SmtpServersEnvioxDia (SmtpServerId,FechaEnvio,LimiteEnviosxDia,EstadoEnviosxDia,TotalEnviosxDia) VALUES (@SmtpServerId,@DateTime,@LimiteEnviosxDia,@EstadoEnviosxDia,@TotalEnviosxDia)");**

**dbDHInsert.AddParameter("@SmtpServerId", (int)Row["ID"]);**

**//dbDHInsert.AddParameter("@FechaEnvio", Convert.ToDateTime(DateTime.Now).ToString("yyyy-MM-dd", ci.DateTimeFormat));**

**dbDHInsert.AddParameter("@LimiteEnviosxDia", (int)Row["EnviosxDia"]);**

**dbDHInsert.AddParameter("@EstadoEnviosxDia", true);**

**dbDHInsert.AddParameter("@TotalEnviosxDia", 0);**

**dbDHInsert.AddParameter("@DateTime", date);**

**dbDHInsert.ExecuteNonQuery();**

**}**

**catch (Exception)**

**{**

**continue;**

**}**

**}**

**DBDataHelper dbDHServers = new DBDataHelper(@"SELECT SmtpServers.EmailRemitente, SmtpServers.ID, SmtpServers.IP, SmtpServers.Port, SmtpServers.Name, SmtpServers.Email, SmtpServers.Password, SmtpServers.Ssl,SmtpServers.Tls, SmtpServers.EnviosxDia , SmtpServersEnvioxDia.EstadoEnviosxDia,SmtpServersEnvioxDia.TotalEnviosxDia**

**FROM SmtpServersEnvioxDia INNER JOIN SmtpServers ON SmtpServersEnvioxDia.SmtpServerId = SmtpServers.ID INNER JOIN app\_SmtpServersxBoletin ON SmtpServersEnvioxDia.SmtpServerId = app\_SmtpServersxBoletin.IDSmtp WHERE SmtpServersEnvioxDia.EstadoEnviosxDia = 'True' AND convert(varchar,FechaEnvio, 11) = convert(varchar,@DateTime, 11) AND app\_SmtpServersxBoletin.IDBoletin = @IDBoletin ORDER BY SmtpServers.ID");**

**dbDHServers.AddParameter("@IDBoletin", BoletinID);**

**dbDHServers.AddParameter("@DateTime", date);**

**return dbDHServers.ExecuteDataTable();**

**}**

Cada smtp configurado tiene un número máximo de correo a enviar por día y un total general, esta disponibilidad de un servidor se explicara en el siguiente tema.

* **Envíos por día/pausas**

En la creación de un Smtp se pone un límite de envíos por día, cuando la aplicación inicia una emisión en el día el smtp realiza un registro de cada smtp que se encuentra trabajando en la tabla **SmtpServersEnvioxDia,** este registra el número permitido por dia del smtp y un contador cada vez que el smtp envía en el día registrado independiente de la emisión, este contador se incrementa, al llegar el smtp al límite de envíos por día este cambiará su estado por pausado, cuando el día finalice este cambiara de nuevo el estado al smtp pausado y continuara con su funcionamiento realizando de nuevo el mismo proceso hasta entrar en pausa en el día.

**Nota**: Solo los smtp activos serán utilizados en los envíos, un smtp puede ser desactivado por el hilo de validación que se encarga de verificar que el smtp este con conexión al servidor de envió, lo puede desactivar si la configuración esta para deshabilitar el smtp en el administrador, también puede entrar en pausa por límite de envíos, cuando una emisión en progreso entra en pausa puede ser que todos los smtp llegaron al límite por envió del día o que no tienen conexión y están deshabilitados, en los dos casos la emisión en curso entrara en pausa por límite de envió.

private DataTable GetSmtpServers()

{

DBDataHelper dbDH1 = new DBDataHelper(@"SELECT ID, IP, Port, Name, Email, Password, Ssl, EnviosxDia, Tls

FROM SmtpServers WHERE Estado = 'True' ORDER BY ID");

return dbDH1.ExecuteDataTable();

}

private DataTable GetSmtpServerxDay()

{

DateTime date = Convert.ToDateTime(DateTime.Now.AddHours(Convert.ToInt32(Global.ConfiguracionHora\_Global)));

foreach (var Row in GetSmtpServers().AsEnumerable())

{

DBDataHelper dbDHServerDay = new DBDataHelper(@"SELECT \*

FROM SmtpServersEnvioxDia WHERE SmtpServerId = @SmtpServerId AND convert(varchar, FechaEnvio , 11) = convert(varchar,@DateTime, 11)");

dbDHServerDay.AddParameter("@SmtpServerId", (int)Row["ID"]);

dbDHServerDay.AddParameter("@DateTime", Convert.ToDateTime(DateTime.Now.AddHours(Convert.ToInt32(Global.ConfiguracionHora\_Global))));

var Result = dbDHServerDay.ExecuteDataTable().Rows.Count;

if (Result >= 1) continue;

try

{

DBDataHelper dbDHInsert = new DBDataHelper(@"INSERT INTO SmtpServersEnvioxDia (SmtpServerId,FechaEnvio,LimiteEnviosxDia,EstadoEnviosxDia,TotalEnviosxDia) VALUES (@SmtpServerId,@DateTime,@LimiteEnviosxDia,@EstadoEnviosxDia,@TotalEnviosxDia)");

dbDHInsert.AddParameter("@SmtpServerId", (int)Row["ID"]);

//dbDHInsert.AddParameter("@FechaEnvio", Convert.ToDateTime(DateTime.Now).ToString("yyyy-MM-dd", ci.DateTimeFormat));

dbDHInsert.AddParameter("@LimiteEnviosxDia", (int)Row["EnviosxDia"]);

dbDHInsert.AddParameter("@EstadoEnviosxDia", true);

dbDHInsert.AddParameter("@TotalEnviosxDia", 0);

dbDHInsert.AddParameter("@DateTime", date);

dbDHInsert.ExecuteNonQuery();

}

catch (Exception)

{

continue;

}

}

DBDataHelper dbDHServers = new DBDataHelper(@"SELECT SmtpServers.EmailRemitente, SmtpServers.ID, SmtpServers.IP, SmtpServers.Port, SmtpServers.Name, SmtpServers.Email, SmtpServers.Password, SmtpServers.Ssl,SmtpServers.Tls, SmtpServers.EnviosxDia , SmtpServersEnvioxDia.EstadoEnviosxDia,SmtpServersEnvioxDia.TotalEnviosxDia

FROM SmtpServersEnvioxDia INNER JOIN SmtpServers ON SmtpServersEnvioxDia.SmtpServerId = SmtpServers.ID INNER JOIN app\_SmtpServersxBoletin ON SmtpServersEnvioxDia.SmtpServerId = app\_SmtpServersxBoletin.IDSmtp WHERE SmtpServersEnvioxDia.EstadoEnviosxDia = 'True' AND convert(varchar,FechaEnvio, 11) = convert(varchar,@DateTime, 11) AND app\_SmtpServersxBoletin.IDBoletin = @IDBoletin ORDER BY SmtpServers.ID");

dbDHServers.AddParameter("@IDBoletin", BoletinID);

dbDHServers.AddParameter("@DateTime", date);

return dbDHServers.ExecuteDataTable();

}

Con estas funcionen se define si el smtp entra en pausa cuando todos los smtp llegan a su límite de envió por día, la emisión en curso entrara en pausa por límite de envíos, al cambiar el día se reinicia de nuevo el contador de los smtp y la emisión reinicia automáticamente su envió con los smtp activos.

1. **POP3**

La aplicación actualmente maneja para la lectura y validación de conexión del pop3 el componente chikat, esta funcionalidad del POP3 tiene una licencia adquirida por la empresa

**MailMan.UnlockComponent("ESTRATMAILQ\_6bcNk7U06R3K");**

La aplicación tiene dos formas de manejar estos rebotes, la primera es por medio de la configuración global para los rebotes con múltiples servidores POP3, esta recibe los rebotes generales de los smtp de envió, estos rebotes son luego filtrados por servidor por medio de una función que se encuentra en el archivo **Bounced.cs** esta función realiza en la lectura de las bandejas de rebotes de los pop3, y registra el rebote a cada servidor POP3 clasificando primero el rebote, puede haber diferentes tipos de rebotes, el componente clasifica estos de la siguiente forma

0 No se ha podido clasificar correo electrónico

1 Rebote Fuerte

2 Rebote Suave

3 Rebote General: No hay dirección de correo electrónico

4 Rebote General

5 Correo Bloqueado

6 Auto-Responder

7 Error Transitorio

8 Solicitud Suscripción

9 Cancelar Suscripción

10 Notificación de Virus

11 Sospecha de Rebote

Luego de ser clasificado el rebote se recupera el id de la emisión a la que pertenece el rebote realizando la lectura del encabezado del correo mediante el siguiente código

**EmisionxSuscriptorID = emailAttached.GetHeaderField("X-EmisionxSuscriptorID");**

**if (string.IsNullOrEmpty(EmisionxSuscriptorID))**

**{**

**string[] lines = email.Body.Split(new string[] { "\r\n", "\n" }, StringSplitOptions.None);**

**var result = Array.FindAll(lines, s => s.StartsWith("X-EmisionxSuscriptorID"));**

**if (result.Length > 0)**

**{**

**string key = result[0];**

**EmisionxSuscriptorID = Regex.Match(key, @"\d+", RegexOptions.RightToLeft).Value;**

**}**

**}**

Luego se crea el registro del rebote, esta verificación de las bandejas de los pop3 configurados se realizan siempre con los que se encuentran activos, estos también se puede deshabilitar por configuración de verificación de conexión desde el admin, esta lectura se realiza por medio de un hilo de Quartz.

La segunda forma se configura para cada boletín cuando se está creado o editando un boletín en la opción Email Rebotes en esta opción se configuran a quien le llegara el rebote si se presenta en ese envió de correo para la emisión en curso por medio del Componente chilkat SMTP en la función de envió de la siguiente forma

// Ajustar la dirección de rebote. Si el correo electrónico rebota, el MDN se enviará a la dirección de rebote.

if (!SmtpTls)

{

if (!string.IsNullOrEmpty(BounceEmail) && Util.VerifyEmailSyntax(BounceEmail))

{

email.BounceAddress = BounceEmail;

}

}

Este manejo de los rebotes solo aplican si el smtp no tiene la configuración de cifrado Tls.

1. **Seguridad Roles**

La seguridad general de la aplicación esta manejada por el integrador de seguridad de asp.net webform el cual es Membership, esta tiene una personación realizada para la aplicación por medio de registros en tablas por rol, los roles tienen diferentes niveles y permisos en cada uno de los módulos de la aplicación

* crear subscriptores
* eliminar subscriptores
* crear listas
* eliminar listas
* ver todas las listas
* ver todos los boletines
* crear y modificar boletines
* eliminar boletines
* Aprobar Boletines
* Eliminar historia Envíos
* Detener /pausar envíos
* Módulo de administración
* Crear /modificar Plantillas
* Eliminar Plantillas
* Ver las Plantillas
* Ver Reportes
* Ver Todos los Suscriptores

Estos son configurables por rol, los roles existentes en la aplicación son

- Administrador

- Publicador\_Nivel1

- Publicador\_Nivel2

- Aprobador

- Publicador\_y\_Aprobador\_Nivel1

- Publicador\_y\_Aprobador\_Nivel2

- Administrador\_Listas

1. **Quartz**

Este componente se utilizó en su primer versión **1.0,** versión completa

Assembly: ScheduleTimer, Version=1.0.0.0, Culture=neutral

Esta es una versión ya sin soporte, esta tiene infinidad de problemas para trabajar dependiendo del ambiente del servidor, es muy inestable pero se logró hacer con ella lo que se requirió, esta tiene un gran problema que no puede reciclar la cache, cargando la memoria temporal del server a medida de que trabaja, este es un componente para realizar Schedule, Jobs pero debido a su versión es muy limitado en la actualidad el componente actual de este mismo fabricante es mucho más completo con múltiples mejoras y soluciones a los problemas que se presentan en esta versión anterior, no se cambió debido a que el manejo de la primer versión es muy diferente a la del componente actual, estos han cambiado la forma como se realizaban las diferentes opciones de configuración y ejecución que presenta el componente para su manejo.

Uno de los problemas que presenta el componente es que si por algún motivo se cae el servicio del iis (reinicios, errores) los hilos de trabajo (Este componente se ejecuta en el pool del iis), este no es capaz de reinicia su trabajo solo, para solucionar esto se realizó una página llamada interna **RunJobs.aspx** esta página cuando se le realizado un llamado esta busca los hilos que se encontraban realizando trabajo y reinicia de nuevo el proceso, por código esta es el método que realiza este reinicio por medio de una interface realizada al componente

namespace App\_Code

{

public interface IRestarMailerJob

{

void ResumeSchedule();

}

public class RestarMailerJob : IRestarMailerJob

{

public int BoletinID { get; set; }

public int EmisionID { get; set; }

public void ResumeSchedule()

{

try

{

DBDataHelper dbDH = new DBDataHelper("SELECT ID AS EmisionID, BoletinID FROM app\_Boletines\_Emisiones WHERE EstadoID = 2 AND FechaHoraFin IS NULL AND FechaPausaLimite IS NULL;");

DataTable dt = dbDH.ExecuteDataTable();

foreach (DataRow row in dt.Rows)

{

BoletinID = (int)row["BoletinID"];

EmisionID = (int)row["EmisionID"];

BoletinEngine be = new BoletinEngine(BoletinID);

//Eliminar Jobs

be.DeleteMailerJob(BoletinID);

//Crear Jobs

be.SetMailerJob(BoletinID, EmisionID);

DBDataHelper dbDHUpdate = new DBDataHelper("UPDATE app\_Boletines\_Emisiones SET EstadoID = 2,FechaPausaLimite = NULL,Observaciones = NULL WHERE ID = @EmisionID");

dbDHUpdate.AddParameter("@EmisionID", EmisionID);

dbDHUpdate.ExecuteNonQuery();

}

}

catch (Exception ex)

{

Util.DumpMsg(string.Format("Error al reiniciar las tareas: ({0})", ex.Message));

}

}

}

}

El iis tiene un servicio de Windows por lo cual este cuando el servidor tiene algún reinicio este sube de nuevo solo, por lo que la aplicación queda en línea, para poder que el quartz reiniciara de nuevo su trabajo sin que el usuario de la aplicación le tocara llamar esta página de reinicio de hilo o su interface manual se creó un servicio de Windows (Windows servicie) el cual realiza el llamado a esta página interna de la aplicación de reinicio, si la aplicación se reiniciara por algún motivo y los hilos de trabajo se cae o fallan, el servicio de Windows que sube automáticamente realiza un llamado a la página interna de la aplicación RunJobs para que continúe con su trabajo, el servicio de Windows debe ser instalado y configurado en el servidor que trabajara, este puede hacer llamados a múltiples páginas de reinicio, independiente que este en el server o en uno diferente, realiza múltiples llamadas a las paginas configuradas en el servicio, reiniciando así el proceso de trabajo, con esto se logró estabilizar en los casos más críticos el hilo de trabajo, como se comenta el componente actual tiene múltiples soluciones ya a los bug que se presentaban anteriormente, ya que con los años todos los ambientes de servidor han cambiado en diferentes versiones de sistemas operativos.

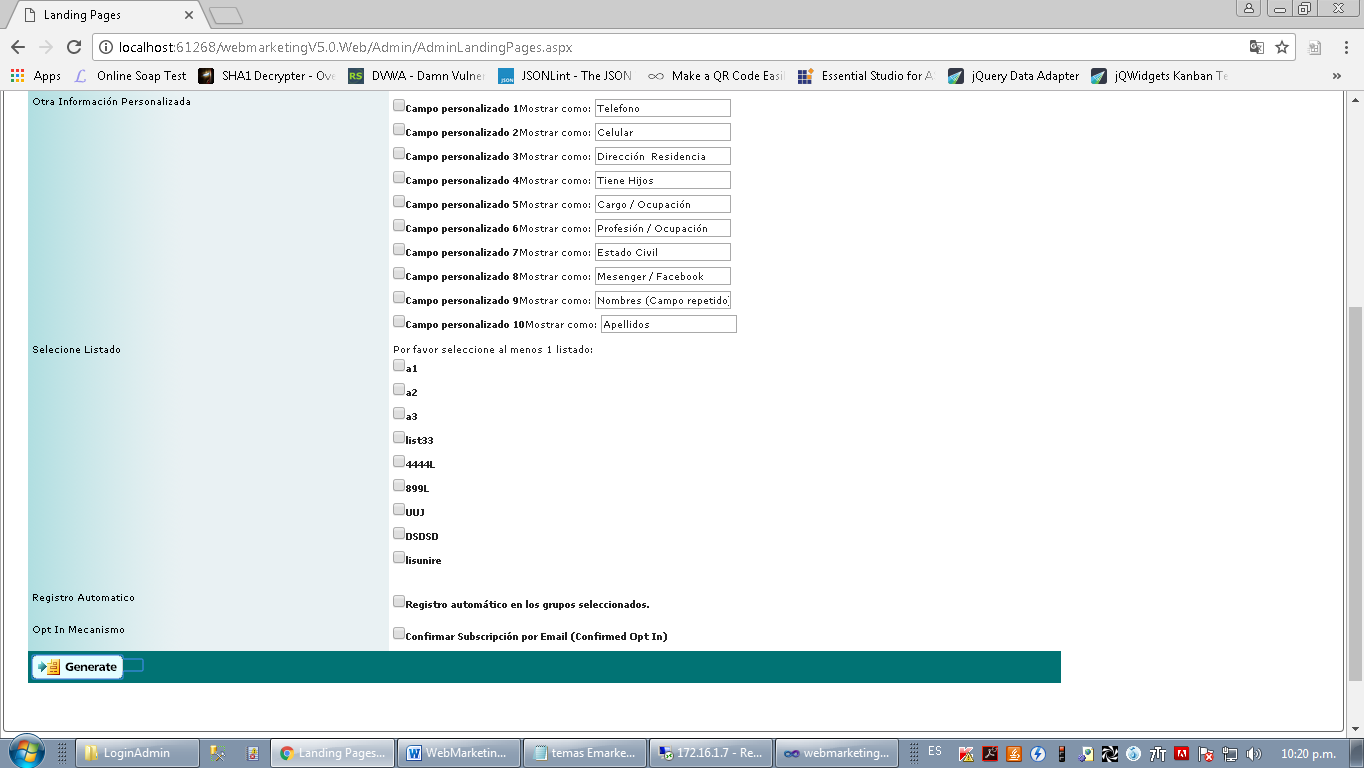
**Nota**: No se entrara en detalle en el manejo de esta versión, ya que la versión actual cambia totalmente el manejo de todas las utilidades del componente.

1. **LandingPages**

La finalidad de esta funcionalidad es los registros de usuarios mediante un formulario dinámico en uno o varios listados de usuarios de la aplicación, para el posterior envío de correo.

El manejo en la aplicación inicia en el siguiente orden

* El primer paso es crear el cuerpo del formulario en html en este se escogen que campos llevara el formulario de registro y a que listados se registrara el usuario por este formulario, se puede poner para que el usuario seleccione los grupos en el formulario de registro o para que la aplicación no muestre las opciones de seleccionar listados, si no que realice el registro a los listados seleccionados automáticamente, también se puede seleccionar la opción si el usuario al registrarse requiere que la aplicación envié un correo a la dirección de email proporcionada por el usuario, el usuario debe confirmar su email para que el registro sea completado en el listado o listados configurados para el registro.



La generación del html mediante c# , puede ser reutilizada para la aplicación futura, se puede extraer por completo, este genera el html del formulario marcando el formulario con una página de respuesta la cual recibirá la petición post del formulario generado, este dependiendo de las opciones seleccionadas oculta valores en el html generado para su posterior lectura esta petición se resive en una página llamada subAccForm.aspx.

Este es el ejemplo de un formulario Generado.

**<form method="POST" action="http://localhost:48714/webmarketingV5.0.Web/subAccForm.aspx?name=SubForm" target="ewSubResult" onsubmit="var sw = screen.availWidth;var sh = screen.availHeight;var width = 500;var t = 0;var l = (sw / 2) - (width / 2); window.open('', 'ewSubResult', 'width=' + width + ',height=' + (sh - 40).toString() + ',left=' + l + ',top=' + t + ',resizable=yes,scrollbars=yes')">**

**<div id='wrapper'>**

**<div id='page-wrapper' class='gray-bg'>**

**<div class='middle-box loginscreen animated fadeInDown'>**

**<div class='row'>**

**<div class='form-group'>**

**<div class='col-md-12'>**

**<label>**

**Dirección de Correo:<br>**

**</label>**

**<input type="text" class="form-control" name="tbxEmail" size="20" maxlength="80">**

**</div>**

**</div>**

**<div class='form-group'>**

**<div class='col-md-12'>**

**<label>**

**Nombre Completo:<br>**

**</label>**

**<input type="text" class="form-control" name="tbxNombres" size="20" maxlength="255">**

**</div>**

**</div>**

**<div style='display: none;' class='form-group'>**

**<div class='col-md-12'>**

**<label>**

**<br />**

**Por favor seleccione el listado al que desea suscribirse:<br />**

**</label>**

**<br />**

**<div class='form-group'>**

**<div class='col-md-12'>**

**<label class='checkbox-inline i-checks'>**

**<input name="cbxMailingList\_0" type="checkbox" checked='true' value="234">**

**pruebas1**

**</label>**

**</div>**

**</div>**

**<br />**

**</div>**

**</div>**

**<br />**

**<div class='form-group'>**

**<div class='col-md-12'>**

**<strong>**

**<br />**

**<input type="submit" value="Suscribirse" class='btn btn-primary block full-width m-b'>**

**</strong>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

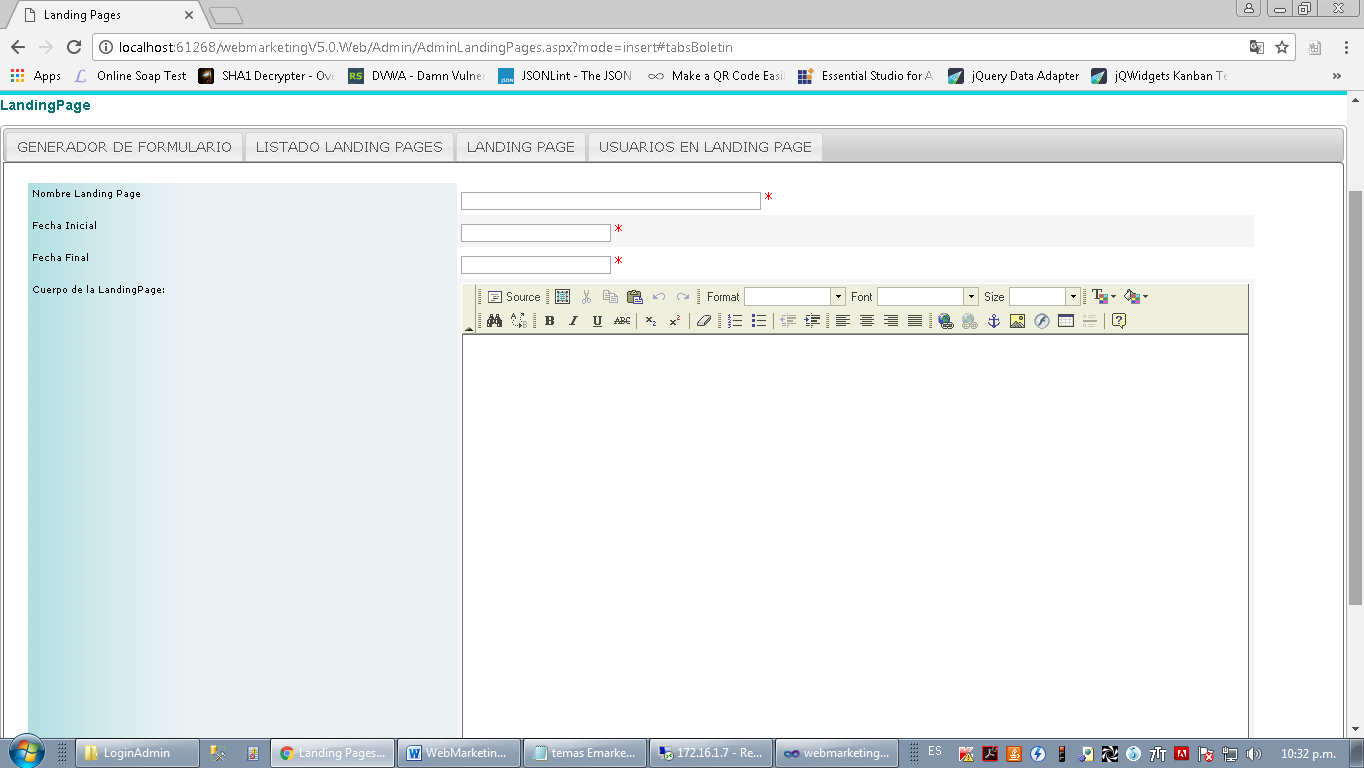
**</div>**

**<input type="hidden" name="hddFormCtrlNroChk" value="1">**

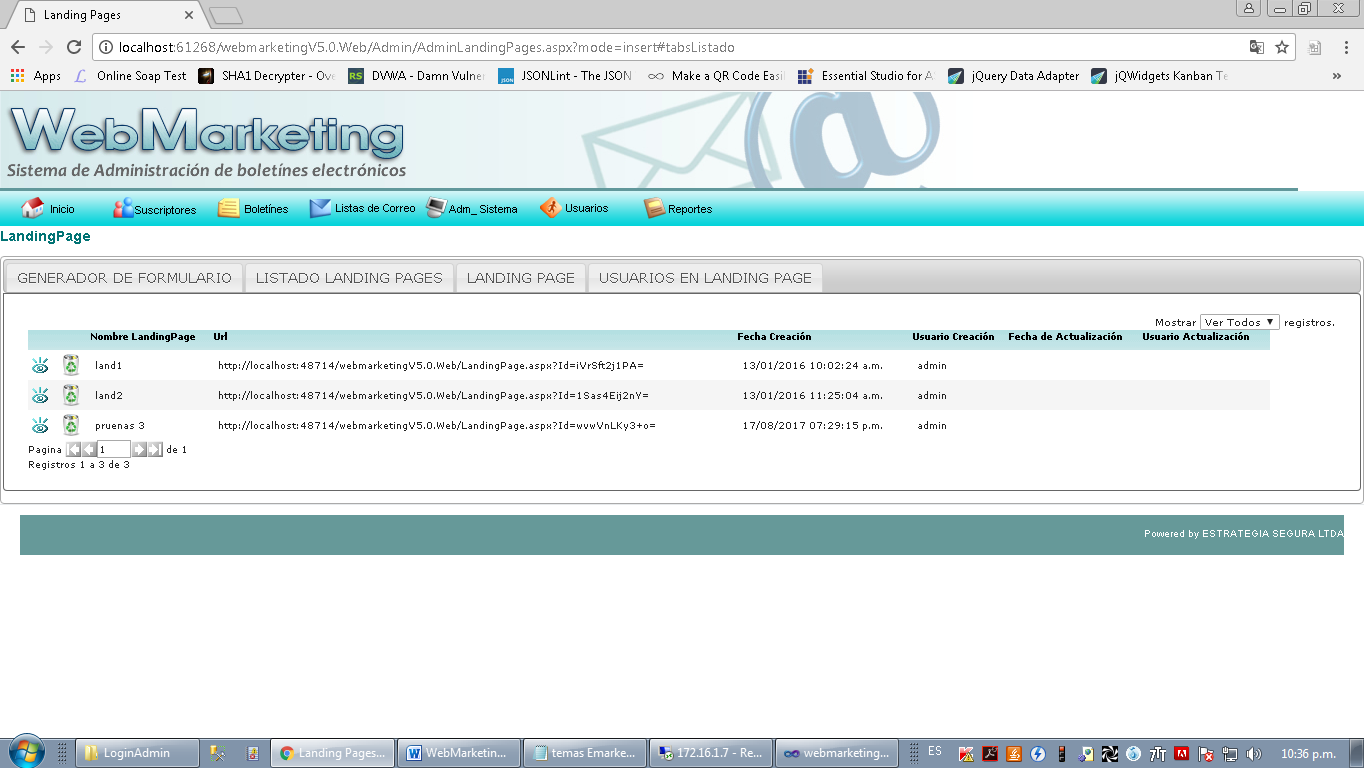
**<input type="hidden" name="hddFormCtrl" value="00000001\_1">**

**</form>**

Este formulario solo realiza el registro si tener en cuenta la información de la ladingpages, para esto se creo un módulo en el cul una vez se genera el formulario este se puede relacionar a una landingpages para tener un registro detallado de este proceso de registro



Una vez se pega el html generado y cargados los parámetros de la landingpages como son la fecha inicial y fecha final de la landingpages, con esto se tiene un control de caducidad del registro en el listado para este formulario, una vez se cumple el rango de tiempo no permitirá al usuario realizar el registro, incluso no llega al formulario de registro, es re direccionado a otro formulario informado que el formulario tenía un rango de fechas para el registro y este ya se cumplió.



Esta generación de una landingpages realiza la creación de una url con la cual se podrá exponer el formulario generado y así llevar un registro de la actividad con este formulario, esta página que expone el formulario de registro puede renderizar estilos de boostrap.

Esta funcionalidad la pueden pasar completa convirtiéndola incluso en una utilidad la generación de formularios dinámicos de registro como una oportunidad de mejora y reutilización de código.

1. **VerifyJob(Verificación de correos)**

Esta funcionalidad de la aplicación permite verificar por medio de diferentes tipos de pruebas el correo de uno o varios listados, existen diferentes tipos de verificación, dependiendo del tipo de verificación será más rápida o lenta la verificación de los correos, los tipos de verificación son

* **Syntaxis**
* **Domain MX record**
* **SMTP connection**
* **Send message attempt**

El código fue extraído de una aplicación de unos chinos que compro la empresa se desarmo la dll con .NET Reflector y se separó el código de verificación este funciona con una clase personalizada llamada **ewEmailVerify** , la clase es personalizada para la verificación de correo por medio de un componente llamado

**Quiksoft.EasyMail.Verify**

La información del ensamblado del componen es la siguiente

**Type: Quiksoft.EasyMail.Verify.EmailVerifier**

**Assembly: Verify.Net, Version=1.0.1.0, Culture=neutral, PublicKeyToken=986a9aa31056680b**

La clase que se personalizo para esta verificación es tiene el siguiente código este dependiendo de la validación utiliza una función del componente el cual se encarga de realizar la verificación, cada una de las verificaciones tiene su configuración con el componente

public ResultCode VerifyEmail(string strEmailAddress, string strSender, string strVerifyLevel, bool isLogReq, out string strLogMsg)

{

ResultCode success = ResultCode.Success;

VerifyLevel expression = VerifyLevel.Expression;

strLogMsg = string.Empty;

if (strVerifyLevel == "M" || strVerifyLevel.Equals("Domain MX record"))

{

expression = VerifyLevel.MXRecords;

}

else if (strVerifyLevel == "P" || strVerifyLevel.Equals("SMTP connection"))

{

expression = VerifyLevel.SmtpConnection;

}

else if (strVerifyLevel == "T" || strVerifyLevel.Equals("Send message attempt"))

{

expression = VerifyLevel.SmtpSendAttempt;

}

try

{

EmailVerifier verifier = new EmailVerifier();

MemoryStream stream = new MemoryStream();

verifier.LogStream = stream;

verifier.RegularExpression = ".\*";

foreach (string str in Share.strsDSNVerify)

{

stream.Seek(0L, SeekOrigin.Begin);

stream.SetLength(0L);

verifier.DnsServers.Clear();

if (!string.IsNullOrEmpty(str))

{

verifier.DnsServers.Add(str);

}

if (expression == VerifyLevel.SmtpSendAttempt)

{

verifier.MailFrom = strSender;

int num = strSender.LastIndexOf('@');

if (num > 0)

{

verifier.HeloDomain = strSender.Substring(num + 1);

}

}

VerifyLevel level2 = verifier.Verify(strEmailAddress, expression);

switch (level2)

{

case VerifyLevel.Success:

success = ResultCode.Success;

break;

case VerifyLevel.Expression:

success = ResultCode.ErrSyntax;

break;

case VerifyLevel.Lists:

success = ResultCode.ErrSystem;

break;

case VerifyLevel.MXRecords:

success = ResultCode.ErrMX;

break;

case VerifyLevel.SmtpConnection:

success = ResultCode.ErrSMTPConn;

break;

case VerifyLevel.SmtpSendAttempt:

success = ResultCode.ErrSend;

break;

default:

success = ResultCode.ErrSystem;

break;

}

if (isLogReq)

{

string str2 = Encoding.ASCII.GetString(stream.GetBuffer());

char[] chArray2 = new char[2];

chArray2[0] = ' ';

char[] trimChars = chArray2;

str2 = str2.Trim(trimChars);

if (level2 == VerifyLevel.MXRecords)

{

if (!str2.EndsWith("\r\n"))

{

str2 = str2 + "\r\n";

}

str2 = str2 + "DSN Server: ";

for (int i = 0; i < verifier.DnsServers.Count; i++)

{

str2 = str2 + verifier.DnsServers[i] + "; ";

}

str2 = str2 + "\r\n\r\n";

}

if (level2 != VerifyLevel.Success)

{

//Util.LogMsg("MSG0192I Email validation failed for [" + strEmailAddress + "]:\r\n" + str2);

}

strLogMsg = strLogMsg + str2;

}

if (level2 == VerifyLevel.Success)

{

return success;

}

}

}

catch (LicenseException exception)

{

Util.LogMsg("MSG0189W - License key error: " + exception.Message);

success = ResultCode.ErrSystem;

}

catch (DnsException exception2)

{

Util.LogMsg("MSG0190W - DNS error: " + exception2.Message);

success = ResultCode.ErrDNS;

}

catch (Exception exception3)

{

//Util.LogMsg("MSG0191W - Fail to verify email address [" + strEmailAddress + "]: " + exception3.Message);

success = ResultCode.ErrSystem;

}

return success;

}

Este componente tiene una licencia extraída de la aplicación de los chinos

**EmailPassword password = new EmailPassword();**

**License.Key = password.DecryptEmailPassword("F56E69CFDA7D9B98DD1F9B0D472B44C6F1361592F352EE46ED2E165CE6B03DC0") + password.DecryptEmailPassword("3B711BDDAE6FFF03D7B11F838F1D34A2CFA9E3D10D5A8CB7E1559BF7327DB3B1");**

**isKeySet = true;**

El componente se encuentra en su primer versión pero cumple con su funcionalidad final, hay componentes en la actualidad que realizan lo mismo pero ya tocaría mirar precios y licencia, así como su implementación atreves de la documentación del proveedor

La verificación de correo realiza una cola para verificar cada correo del listado, esta verificación se vuelve un poco compleja ya que los listados tienen diferentes fuentes como procedimientos almacenados, vistas, usuarios internos.

Se realiza mediante un hilo que se programa para que realice la verificación configurada por cada correo, es una operación la cual el usuario puede poner a trabajar y cerrar la página y esta notificará al terminar la verificación de los correo dando un informe final de la verificación adjunta en el correo de notificación, esta verificación tiene un gran problema debido al uso de los hilos del componente de tareas esta carga de tal manera la cache de la aplicación que se come casi por completo la memoria de un servidor ya que si se encuentra envían y realizando verificación los hilos del componente antiguo no soportan esto y presenta problemas, toca realizar estos procesos individualmente, si está enviando no realizar verificaciones de correo y viceversa.

1. **Listados**

* **Carga Masiva**

Esta carga se reprogramo, para distribuir la carga de trabajo entre la aplicación y la base de datos, el filtrado y validaciones del listado se realizan desde la base de datos y la carga se realiza desde la aplicación del listado limpio.

* **Remoción masiva de Subscriptores**

Se realiza mediante un archivo de Excel precargado, este elimina los usuarios en general de la aplicación.

* **Búsqueda Avanzada**

Se utiliza un filtro avanzado mediante el cual se buscan los usuarios según el criterio seleccionado en la búsqueda, luego el resultado de la búsqueda se puede adicionar a un grupo, o crear un nuevo grupo con esto.

1. **Log de registro(Errores y actividad)**

Se realiza el seguimiento de toda actividad de la aplicación registra en una tabla llamada DumpLogs, en esta se registra todo el trabajo de la aplicación, para los errores hay una tabla que clasifica los errores personalizando los mensajes de error llamada ewns\_tblMessage, y también se capturan por medio de controladores de error los cuales redirigen el error hasta el global y allí se registra el error de la siguiente forma

**void Application\_Error(object sender, EventArgs e)**

**{**

**Exception ex = Server.GetLastError().GetBaseException();**

**StackTrace trace = new StackTrace(ex, true);**

**if (trace.FrameCount == 0)**

**return;**

**StackFrame stackFrame = trace.GetFrame(0);**

**string className, fileName, functionName, message;**

**int line = 0;**

**// if for some reason, filename is being retrieved as null**

**if (stackFrame.GetFileName() == null)**

**{**

**className = ex.TargetSite.ReflectedType.Name;**

**fileName = ex.TargetSite.ReflectedType.Name;**

**functionName = ex.TargetSite.Name;**

**message = ex.Message;**

**}**

**else**

**{**

**// Collect data where exception occured**

**string[] splitFile = stackFrame.GetFileName().Split('\\');**

**className = splitFile[splitFile.Length - 1];**

**fileName = stackFrame.GetFileName();**

**functionName = stackFrame.GetMethod().Name;**

**message = ex.Message;**

**line = stackFrame.GetFileLineNumber();**

**}**

**#region**

**if (!functionName.Equals("CheckVirtualFileExists") && !message.Contains("Conversion failed when converting the nvarchar value") && !message.Contains("ChilkatLog: SendEmail"))**

**{**

**string error =**

**string.Format(**

**" className : {0} // fileName : {1} // functionName : {2} // message : {3} // line : {4} ", className, fileName, functionName, message, line);**

**Util.LogMsg(error);**

**}**

**Server.ClearError();**

**#endregion**

**}**

1. Informes

Para la generación de informes fue necesario realizar una tarea mediante un hilo de proceso que se encarga de realizar cálculos y conteos de las diferentes emisiones realizadas este hilo ejecuta unos procedimientos almacenados que registran los contadores en 3 tablas

* app\_InfoEmision
* app\_InfoSmtpSend
* app\_InfoSmtpSendMonth

Mediante estas tablas se realizan los informes de la aplicación mejorando el tiempo de ejecución y problemas anteriores para generar los reportes de la aplicación.

1. **Web Service Emarketing**

La aplicación tiene un webservice creado mediante el cual se exponen las siguientes operaciones

**CrearSuscriptor(string email, string listadoID, string cp1, string cp2, string cp3, string cp4, string cp5, string cp6, string cp7, string cp8, string cp9, string cp10, string nombre, string cedula, string tokenUser);**

**CrearGrupo(string nombreGrupo, string tokenUser);**

**ListarGrupos(string tokenUser);**

**ListarGruposEmail(string email, string tokenUser);**

**EliminarSuscriptores(string email, string tokenUser);**

**EliminarSuscriptor(string email, string listadoID, string tokenUser);**

**BuscarSuscriptor(string email, string listadoID, string tokenUser);**

**BuscarSuscriptoresGrupos(string email, string tokenUser);**

**BuscarBoletinesEnviadosSuscriptor(string email, string tokenUser);**

**BuscarBoletinesLeidosSuscriptor(string email, string tokenUser);**

**BuscarSuscriptoresCancelados(string listadoID, string tokenUser);**

**BuscarSuscriptorCancelados(string email, string tokenUser);**

**BuscarSuscriptoresBlackList(string tokenUser);**

**BuscarSuscriptorBlackList(string email, string tokenUser);**

**BuscarSuscriptorReboteFuerte(string email, string tokenUser);**

1. **Elementos base de datos**

El procedimiento almacenado de filtrado, se encarga de limpiar una lista de correos por medio de un sp este tiene una paginación ya que se presentan casos de cliente conectados a sp con gran cantidad de datos

SpPruebas

**ALTER PROCEDURE [dbo].[SpPruebas]**

**@Start int,**

**@End int,**

**@paging bit**

**/\*Cursor\*/**

**AS**

**IF OBJECT\_ID('tempdb..#NewServerTable') IS NULL**

**BEGIN**

**--no existe**

**CREATE TABLE #NewServerTable (CedulaNew varchar(50), Nombrenew varchar(200),Emailnew VARCHAR(200))**

**END**

**ELSE**

**BEGIN**

**--existe**

**DELETE from #NewServerTable**

**END**

**/\*Cursor\*/**

**DECLARE @Cedula varchar(50), @Nombre varchar(200),@Email varchar(200)**

**DECLARE cursorRepetidos CURSOR**

**FOR**

**Select Cedula, Nombre, Mail FROM dbo.TbUsuarios**

**OPEN cursorRepetidos -- open the cursor**

**FETCH NEXT FROM cursorRepetidos**

**INTO @Cedula, @Nombre, @Email**

**-- Fila 0**

**WHILE @@FETCH\_STATUS = 0**

**BEGIN**

**/\*Zona de trabajo\*/**

**IF NOT EXISTS (SELECT \* FROM #NewServerTable WHERE Emailnew = @Email)**

**BEGIN**

**INSERT INTO #NewServerTable (CedulaNew,Nombrenew,Emailnew) values (@Cedula,@Nombre,@Email)**

**END**

**/\*Fin Zona\*/**

**FETCH NEXT FROM cursorRepetidos**

**-- Filas siguientes**

**INTO @Cedula, @Nombre, @Email**

**END**

**CLOSE cursorRepetidos -- close the cursor**

**DEALLOCATE cursorRepetidos -- Deallocate the cursor**

**/\*Fin Cursor\*/**

**BEGIN**

**/\*Paginación listado\*/**

**IF(@paging = 1)**

**BEGIN**

**WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY Email) AS RowNumber from (Select CedulaNew as Cedula,Nombrenew as Nombre,Emailnew as Email , ' ' as Cp1,' ' as Cp2, ' ' as Cp3, ' ' as Cp4,' ' as Cp5,' ' as Cp6,' ' as Cp7,' ' as Cp8,' ' as Cp9,' ' as Cp10 from #NewServerTable) a ) SELECT \* FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End**

**END**

**ELSE**

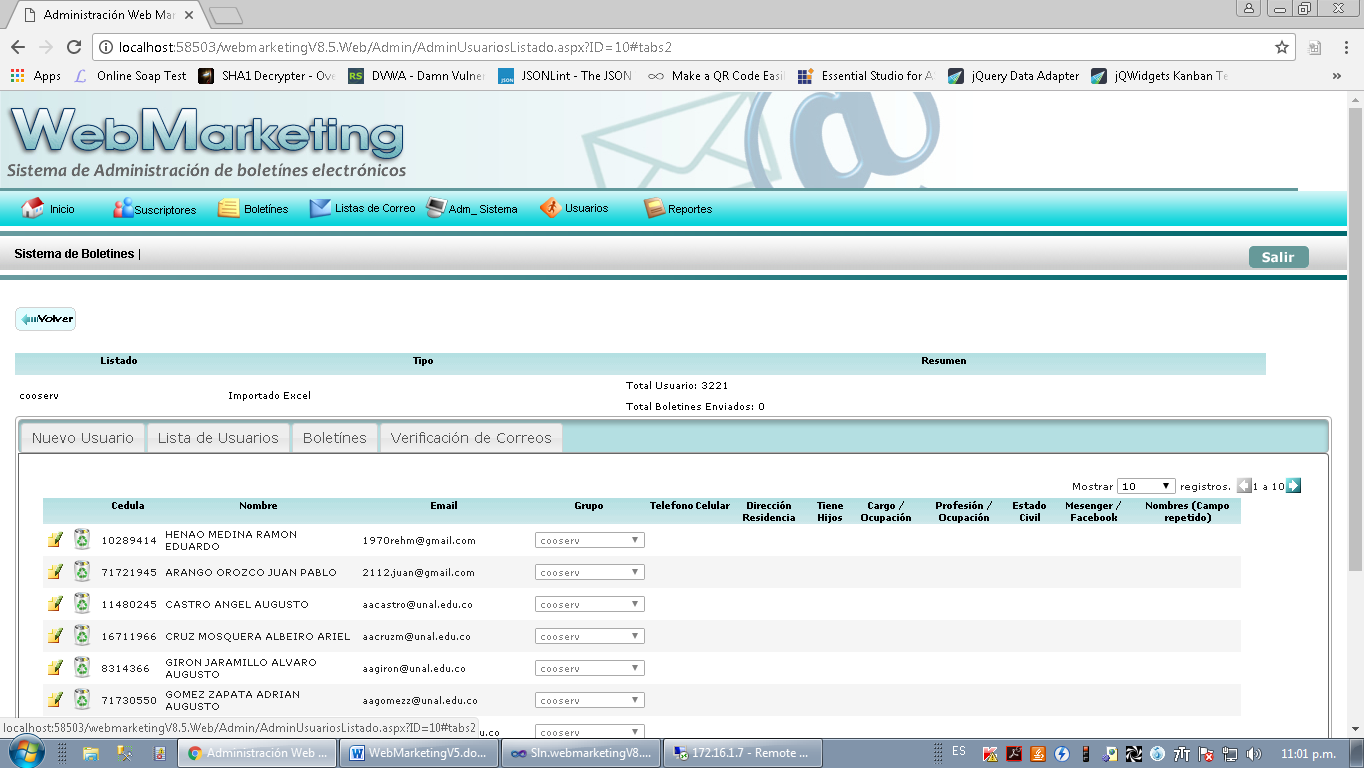
**BEGIN**

**Select CedulaNew as Cedula,Nombrenew as Nombre,Emailnew as Email , ' ' as Cp1,' ' as Cp2, ' ' as Cp3, ' ' as Cp4,' ' as Cp5,' ' as Cp6,' ' as Cp7,' ' as Cp8,' ' as Cp9,' ' as Cp10 from #NewServerTable**

**END**

**END**

Esta paginación se ejecuta en los listados al ver su detalle en el módulo del webmarketing



Esta paginación se realiza por medio de c#

**protected void PagerBinding(object sender, EventArgs e)**

**{**

**int pageIndex = GridView1.PageIndex;**

**int pageSize = GridView1.PageSize;**

**bool allowPaging = GridView1.AllowPaging;**

**int pageCount = GridView1.PageCount;**

**int count = GridView1.Rows.Count;**

**this.txtPager.Text = Convert.ToString((int)(pageIndex + 1));**

**if (allowPaging)**

**{**

**if (pageIndex == 0)**

**{**

**this.btnPagerFirst.ImageUrl = "~/images/firstdisab.gif";**

**this.btnPagerFirst.Enabled = false;**

**this.btnPagerPrev.ImageUrl = "~/images/prevdisab.gif";**

**this.btnPagerPrev.Enabled = false;**

**}**

**else**

**{**

**this.btnPagerFirst.ImageUrl = "~/images/first.gif";**

**this.btnPagerFirst.Enabled = true;**

**this.btnPagerPrev.ImageUrl = "~/images/prev.gif";**

**this.btnPagerPrev.Enabled = true;**

**}**

**if (pageIndex == (pageCount - 1))**

**{**

**this.btnPagerNext.ImageUrl = "~/images/nextdisab.gif";**

**this.btnPagerNext.Enabled = false;**

**this.btnPagerLast.ImageUrl = "~/images/lastdisab.gif";**

**this.btnPagerLast.Enabled = false;**

**}**

**else**

**{**

**this.btnPagerNext.ImageUrl = "~/images/next.gif";**

**this.btnPagerNext.Enabled = true;**

**this.btnPagerLast.ImageUrl = "~/images/last.gif";**

**this.btnPagerLast.Enabled = true;**

**}**

**this.txtPager.Text = Convert.ToString((int)(pageIndex + 1));**

**}**

**else**

**{**

**this.btnPagerFirst.ImageUrl = "~/images/firstdisab.gif";**

**this.btnPagerFirst.Enabled = false;**

**this.btnPagerPrev.ImageUrl = "~/images/prevdisab.gif";**

**this.btnPagerPrev.Enabled = false;**

**this.btnPagerNext.ImageUrl = "~/images/nextdisab.gif";**

**this.btnPagerNext.Enabled = false;**

**this.btnPagerLast.ImageUrl = "~/images/lastdisab.gif";**

**this.btnPagerLast.Enabled = false;**

**this.txtPager.Text = "1";**

**}**

**this.lblTotalRecs.Text = Convert.ToString(count);**

**if (count > 0)**

**{**

**this.lblStartRec.Text = Convert.ToString((int)((pageIndex \* pageSize) + 1));**

**}**

**else**

**{**

**this.lblStartRec.Text = "0";**

**}**

**this.lblStopRec.Text = Convert.ToString((int)((pageIndex \* pageSize) + count));**

**this.lblPageCount.Text = Convert.ToString(pageCount);**

**}**

**protected void btnPagerFirst\_Click(object sender, ImageClickEventArgs e)**

**{**

**GridView1.PageIndex = 0;**

**GridView1.DataBind();**

**}**

**protected void btnPagerLast\_Click(object sender, ImageClickEventArgs e)**

**{**

**int pageCount = GridView1.PageCount;**

**GridView1.PageIndex = pageCount - 1;**

**GridView1.DataBind();**

**}**

**protected void btnPagerNext\_Click(object sender, ImageClickEventArgs e)**

**{**

**GridView1.PageIndex++;**

**GridView1.DataBind();**

**}**

**protected void btnPagerPrev\_Click(object sender, ImageClickEventArgs e)**

**{**

**GridView1.PageIndex--;**

**GridView1.DataBind();**

**}**

Con este código se controla los botones de opciones de vistas en los listados, este mismo tiempo de paginación se usa en los otros tipo de listados debido a la gran cantidad de usuarios que registraban ciertos clientes en esta aplicación.

A su vez se implementó esta paginación de diferentes módulos debido a la necesidad de optimizar la carga de la página, la paginación se realiza desde cada una de los elementos de fuentes de datos de las tablas de lista de asp.net **webform**

**<asp:SqlDataSource ID="SDS\_SUCCESS" runat="server" ConnectionString="<%$ ConnectionStrings:ApplicationServices %>" SelectCommand="WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY SuscriptorID) AS RowNumber from (SELECT app\_Boletines\_Emisiones\_Suscriptores.SuscriptorID,app\_Boletines\_Emisiones\_Suscriptores.SuscriptorNombre,Listados.NombreLista AS Grupo,'' as IdSendEmail,'' as IdBounce, convert(varchar,app\_Boletines\_Emisiones\_Suscriptores.FechaHoraEnviado,120) AS FechaHoraEnviado,app\_Boletines\_Emisiones\_Suscriptores.Email,app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId, SmtpServers.IP,app\_Boletines\_Emisiones\_Suscriptores.Observaciones, SmtpServers.Email AS Cuenta FROM app\_Boletines\_Emisiones\_Suscriptores WITH (TABLOCK ) LEFT OUTER JOIN SmtpServers ON app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId = SmtpServers.ID LEFT OUTER JOIN Listados ON app\_Boletines\_Emisiones\_Suscriptores.SuscriptorListadoID = Listados.Id WHERE (app\_Boletines\_Emisiones\_Suscriptores.EmisionID = @EmisionID) AND (app\_Boletines\_Emisiones\_Suscriptores.Observaciones IS NULL) AND (SmtpServers.IP IS NOT NULL)) a ) SELECT \* FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End">**

**<SelectParameters>**

**<asp:QueryStringParameter Name="EmisionID" QueryStringField="ID" />**

**<asp:ControlParameter Name="Start" ControlID="lblStartRec" PropertyName="Text" />**

**<asp:ControlParameter Name="End" ControlID="lblStopRec" PropertyName="Text" />**

**</SelectParameters>**

**</asp:SqlDataSource>**

**<asp:SqlDataSource ID="SDS\_FAILED" runat="server" ConnectionString="<%$ ConnectionStrings:ApplicationServices %>" SelectCommand=";WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY SuscriptorID) AS RowNumber from (SELECT app\_Boletines\_Emisiones\_Suscriptores.SuscriptorID,app\_Boletines\_Emisiones\_Suscriptores.SuscriptorNombre,Listados.NombreLista AS Grupo,'' as IdSendEmail,'' as IdBounce,convert(varchar,app\_Boletines\_Emisiones\_Suscriptores.FechaHoraEnviado,120) AS FechaHoraEnviado, app\_Boletines\_Emisiones\_Suscriptores.Email, app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId, SmtpServers.IP, app\_Boletines\_Emisiones\_Suscriptores.Observaciones, SmtpServers.Email AS Cuenta FROM app\_Boletines\_Emisiones\_Suscriptores WITH (TABLOCK ) LEFT OUTER JOIN SmtpServers ON app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId = SmtpServers.ID LEFT OUTER JOIN Listados ON app\_Boletines\_Emisiones\_Suscriptores.SuscriptorListadoID = Listados.Id WHERE (app\_Boletines\_Emisiones\_Suscriptores.EmisionID = @EmisionID) AND (app\_Boletines\_Emisiones\_Suscriptores.Observaciones IS NOT NULL AND (app\_Boletines\_Emisiones\_Suscriptores.Observaciones != 'Suscripción Cancelada.'))) a ) SELECT \*,(SELECT MAX(RowNumber) FROM objPaging) AS 'TotalRows' FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End">**

**<SelectParameters>**

**<asp:QueryStringParameter Name="EmisionID" QueryStringField="ID" />**

**<asp:ControlParameter Name="Start" ControlID="lblStartRec" PropertyName="Text" />**

**<asp:ControlParameter Name="End" ControlID="lblStopRec" PropertyName="Text" />**

**</SelectParameters>**

**</asp:SqlDataSource>**

**<asp:SqlDataSource ID="SDS\_POP3" runat="server" ConnectionString="<%$ ConnectionStrings:ApplicationServices %>" SelectCommand="WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY SuscriptorID) AS RowNumber from ( SELECT app\_Boletines\_Emisiones\_Suscriptores.SuscriptorID,app\_Boletines\_Emisiones\_Suscriptores.SuscriptorNombre,Listados.NombreLista AS Grupo,'' as IdSendEmail, app\_BouncePOP3.IdBounce ,convert(varchar,app\_Boletines\_Emisiones\_Suscriptores.FechaHoraEnviado,120) AS FechaHoraEnviado,app\_Boletines\_Emisiones\_Suscriptores.Email,(app\_TypeBounce.TextBounce) as Observaciones, app\_Boletines\_Emisiones\_Suscriptores.Email AS Cuenta, app\_BouncePOP3.DateBounce,SmtpServers.IP FROM app\_BouncePOP3 WITH (TABLOCK ) INNER JOIN app\_Boletines\_Emisiones\_Suscriptores ON app\_BouncePOP3.EmisionxSuscriptorID = app\_Boletines\_Emisiones\_Suscriptores.EmisionxSuscriptorID Inner join app\_TypeBounce on app\_BouncePOP3.TypeBounce = app\_TypeBounce.IdTypeBounce LEFT OUTER JOIN SmtpServers ON app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId = SmtpServers.ID LEFT OUTER JOIN Listados ON app\_Boletines\_Emisiones\_Suscriptores.SuscriptorListadoID = Listados.Id WHERE app\_Boletines\_Emisiones\_Suscriptores.EmisionID = @EmisionID) a ) SELECT \*,(SELECT MAX(RowNumber) FROM objPaging) AS 'TotalRows' FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End ">**

**<SelectParameters>**

**<asp:QueryStringParameter Name="EmisionID" QueryStringField="ID" />**

**<asp:ControlParameter Name="Start" ControlID="lblStartRec" PropertyName="Text" />**

**<asp:ControlParameter Name="End" ControlID="lblStopRec" PropertyName="Text" />**

**</SelectParameters>**

**</asp:SqlDataSource>**

**<asp:SqlDataSource ID="SDS\_OPEN" runat="server" ConnectionString="<%$ ConnectionStrings:ApplicationServices %>" SelectCommand="WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY SuscriptorID) AS RowNumber from (SELECT app\_Boletines\_Emisiones\_Suscriptores.SuscriptorID,app\_Boletines\_Emisiones\_Suscriptores.SuscriptorNombre,Listados.NombreLista AS Grupo,'' as IdBounce, 'Open' as IdSendEmail ,('Fecha Leido: '+ convert(varchar,app\_Boletines\_Emisiones\_Suscriptores.FechaHoraEnviado,120) + ' - Ip Leido: ' + app\_Boletines\_Emisiones\_Suscriptores.IpOpen) AS FechaHoraEnviado ,app\_Boletines\_Emisiones\_Suscriptores.Email,('Fecha Leido '+ CONVERT(VARCHAR(50),DateOpenEmail,100)) as Observaciones, app\_Boletines\_Emisiones\_Suscriptores.Email AS Cuenta, SmtpServers.IP FROM app\_Boletines\_Emisiones\_Suscriptores WITH (TABLOCK ) LEFT OUTER JOIN SmtpServers ON app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId = SmtpServers.ID LEFT OUTER JOIN Listados ON app\_Boletines\_Emisiones\_Suscriptores.SuscriptorListadoID = Listados.Id WHERE app\_Boletines\_Emisiones\_Suscriptores.EmisionID = @EmisionID and app\_Boletines\_Emisiones\_Suscriptores.IsOpen = 'True') a ) SELECT \*,(SELECT MAX(RowNumber) FROM objPaging) AS 'TotalRows' FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End ">**

**<SelectParameters>**

**<asp:QueryStringParameter Name="EmisionID" QueryStringField="ID" />**

**<asp:ControlParameter Name="Start" ControlID="lblStartRec" PropertyName="Text" />**

**<asp:ControlParameter Name="End" ControlID="lblStopRec" PropertyName="Text" />**

**</SelectParameters>**

**</asp:SqlDataSource>**

**<asp:SqlDataSource ID="SDS\_CANCEL" runat="server" ConnectionString="<%$ ConnectionStrings:ApplicationServices %>" SelectCommand="WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY SuscriptorID) AS RowNumber from (SELECT app\_Boletines\_Emisiones\_Suscriptores.SuscriptorID,app\_Boletines\_Emisiones\_Suscriptores.SuscriptorNombre,Listados.NombreLista AS Grupo,'' as IdSendEmail,'' as IdBounce,convert(varchar,app\_Boletines\_Emisiones\_Suscriptores.FechaHoraEnviado,120) AS FechaHoraEnviado, app\_Boletines\_Emisiones\_Suscriptores.Email, app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId, SmtpServers.IP, app\_Boletines\_Emisiones\_Suscriptores.Observaciones, SmtpServers.Email AS Cuenta FROM app\_Boletines\_Emisiones\_Suscriptores WITH (TABLOCK ) LEFT OUTER JOIN SmtpServers ON app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId = SmtpServers.ID LEFT OUTER JOIN Listados ON app\_Boletines\_Emisiones\_Suscriptores.SuscriptorListadoID = Listados.Id WHERE (app\_Boletines\_Emisiones\_Suscriptores.EmisionID = @EmisionID) AND (app\_Boletines\_Emisiones\_Suscriptores.Observaciones = 'Suscripción Cancelada.')) a ) SELECT \*,(SELECT MAX(RowNumber) FROM objPaging) AS 'TotalRows' FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End">**

**<SelectParameters>**

**<asp:QueryStringParameter Name="EmisionID" QueryStringField="ID" />**

**<asp:ControlParameter Name="Start" ControlID="lblStartRec" PropertyName="Text" />**

**<asp:ControlParameter Name="End" ControlID="lblStopRec" PropertyName="Text" />**

**</SelectParameters>**

**</asp:SqlDataSource>**

**<asp:SqlDataSource ID="SDS\_CONFIRM" runat="server" ConnectionString="<%$ ConnectionStrings:ApplicationServices %>" SelectCommand="WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY SuscriptorID) AS RowNumber from (SELECT app\_Boletines\_Emisiones\_Suscriptores.SuscriptorID,app\_Boletines\_Emisiones\_Suscriptores.SuscriptorNombre,Listados.NombreLista AS Grupo,'' as IdSendEmail,'' as IdBounce,convert(varchar,app\_Boletines\_Emisiones\_Suscriptores.FechaConfirmacionEmail,120) AS FechaHoraEnviado, app\_Boletines\_Emisiones\_Suscriptores.Email, app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId, SmtpServers.IP, app\_Boletines\_Emisiones\_Suscriptores.Observaciones, SmtpServers.Email AS Cuenta FROM app\_Boletines\_Emisiones\_Suscriptores WITH (TABLOCK ) LEFT OUTER JOIN SmtpServers ON app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId = SmtpServers.ID LEFT OUTER JOIN Listados ON app\_Boletines\_Emisiones\_Suscriptores.SuscriptorListadoID = Listados.Id WHERE (app\_Boletines\_Emisiones\_Suscriptores.EmisionID = @EmisionID) AND (app\_Boletines\_Emisiones\_Suscriptores.ConfirmacionEmail = 'True')) a ) SELECT \*,(SELECT MAX(RowNumber) FROM objPaging) AS 'TotalRows' FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End">**

**<SelectParameters>**

**<asp:QueryStringParameter Name="EmisionID" QueryStringField="ID" />**

**<asp:ControlParameter Name="Start" ControlID="lblStartRec" PropertyName="Text" />**

**<asp:ControlParameter Name="End" ControlID="lblStopRec" PropertyName="Text" />**

**</SelectParameters>**

**</asp:SqlDataSource>**

**<asp:SqlDataSource ID="SDS\_CONFIRMSIMPLE" runat="server" ConnectionString="<%$ ConnectionStrings:ApplicationServices %>" SelectCommand="WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY SuscriptorID) AS RowNumber from (SELECT app\_Boletines\_Emisiones\_Suscriptores.SuscriptorID,app\_Boletines\_Emisiones\_Suscriptores.SuscriptorNombre,Listados.NombreLista AS Grupo,'' as IdSendEmail,'' as IdBounce,convert(varchar,app\_Boletines\_Emisiones\_Suscriptores.FechaHoraEnviado,120) AS FechaHoraEnviado, app\_Boletines\_Emisiones\_Suscriptores.Email, app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId, SmtpServers.IP,('Respuesta :' + (CASE WHEN app\_Boletines\_Emisiones\_Suscriptores.ConfirmacionSimple = 'True' THEN 'SI' ELSE 'NO' END) + ' / Fecha de Confirmación:' + app\_Boletines\_Emisiones\_Suscriptores.FechaConfirmacionSimple) AS Observaciones, SmtpServers.Email AS Cuenta FROM app\_Boletines\_Emisiones\_Suscriptores WITH (TABLOCK ) LEFT OUTER JOIN SmtpServers ON app\_Boletines\_Emisiones\_Suscriptores.SmtpServerId = SmtpServers.ID LEFT OUTER JOIN Listados ON app\_Boletines\_Emisiones\_Suscriptores.SuscriptorListadoID = Listados.Id WHERE (app\_Boletines\_Emisiones\_Suscriptores.EmisionID = @EmisionID) AND (app\_Boletines\_Emisiones\_Suscriptores.ConfirmacionSimple IS NOT NULL)) a ) SELECT \*,(SELECT MAX(RowNumber) FROM objPaging) AS 'TotalRows' FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End">**

**<SelectParameters>**

**<asp:QueryStringParameter Name="EmisionID" QueryStringField="ID" />**

**<asp:ControlParameter Name="Start" ControlID="lblStartRec" PropertyName="Text" />**

**<asp:ControlParameter Name="End" ControlID="lblStopRec" PropertyName="Text" />**

**</SelectParameters>**

**</asp:SqlDataSource>**

**<asp:SqlDataSource ID="SDS\_UNSUCCESS" runat="server" ConnectionString="<%$ ConnectionStrings:ApplicationServices %>" SelectCommand=";WITH objPaging AS ( SELECT \*,ROW\_NUMBER() OVER (ORDER BY SuscriptorID) AS RowNumber from (SELECT Usuarios.Cedula AS SuscriptorID,Usuarios.Nombre AS SuscriptorNombre,Listados.NombreLista AS Grupo,'' as IdSendEmail,'' as IdBounce,'' AS FechaHoraEnviado, Usuarios.Email, '' AS SmtpServerId, '' AS IP, 'En cola de Envio...' AS Observaciones,'' AS Cuenta FROM Usuarios LEFT OUTER JOIN Listados ON Usuarios.ListadoID = Listados.Id WHERE (Usuarios.BoletinID = (Select [BoletinID] FROM [dbo].[app\_Boletines\_Emisiones] WHERE ID = @EmisionID ))) a ) SELECT \*,(SELECT MAX(RowNumber) FROM objPaging) AS 'TotalRows' FROM objPaging WHERE RowNumber >= @Start AND RowNumber <= @End">**

**<SelectParameters>**

**<asp:QueryStringParameter Name="EmisionID" QueryStringField="ID" />**

**<asp:ControlParameter Name="Start" ControlID="lblStartRec" PropertyName="Text" />**

**<asp:ControlParameter Name="End" ControlID="lblStopRec" PropertyName="Text" />**

**</SelectParameters>**

**</asp:SqlDataSource>**

A diferencia de la paginación anterior esta se ejecuta por medio de elementos de **webform** para fuentes de datos,esta cambia un poco la manera con carga las fuentes de datos y las pagina mediante c# ya que se utilizan los elementos cargados en la pagina mediante webform

**protected void loadPaging()**

**{**

**string total = "0";**

**if (string.IsNullOrEmpty(Request.QueryString["success"])) return;**

**switch (Request.QueryString["success"])**

**{**

**case "True":**

**total = Menu1.Items[0].Text;**

**break;**

**case "False":**

**total = Menu1.Items[1].Text;**

**break;**

**case "POP3":**

**total = Menu1.Items[2].Text;**

**break;**

**case "OPEN":**

**total = Menu1.Items[3].Text;**

**break;**

**case "CANCEL":**

**total = Menu1.Items[4].Text;**

**break;**

**case "CONFIRM":**

**total = Menu1.Items[5].Text;**

**break;**

**case "CONFIRMSIMPLE":**

**total = Menu1.Items[6].Text;**

**break;**

**case "Unsuccess":**

**total = Menu1.Items[7].Text;**

**break;**

**}**

**lblStartRec.Text = "1";**

**lblStopRec.Text = DropDownList1.SelectedItem.Text;**

**GridView1.DataBind();**

**int stopTotal = Convert.ToInt32(total);**

**if (Convert.ToInt32(lblStopRec.Text) >= stopTotal)**

**{**

**lblStopRec.Text = Convert.ToString(stopTotal);**

**this.btnPagerNext.ImageUrl = "~/images/nextdisab.gif";**

**this.btnPagerNext.Enabled = false;**

**}**

**else**

**{**

**this.btnPagerNext.ImageUrl = "~/images/next.gif";**

**this.btnPagerNext.Enabled = true;**

**}**

**this.btnPagerPrev.ImageUrl = "~/images/prevdisab.gif";**

**this.btnPagerPrev.Enabled = false;**

**}**

**protected void btnPagerNext\_Click(object sender, ImageClickEventArgs e)**

**{**

**string total = "0";**

**if (string.IsNullOrEmpty(Request.QueryString["success"])) return;**

**switch (Request.QueryString["success"])**

**{**

**case "True":**

**total = Menu1.Items[0].Text;**

**break;**

**case "False":**

**total = Menu1.Items[1].Text;**

**break;**

**case "POP3":**

**total = Menu1.Items[2].Text;**

**break;**

**case "OPEN":**

**total = Menu1.Items[3].Text;**

**break;**

**case "CANCEL":**

**total = Menu1.Items[4].Text;**

**break;**

**case "CONFIRM":**

**total = Menu1.Items[5].Text;**

**break;**

**case "CONFIRMSIMPLE":**

**total = Menu1.Items[6].Text;**

**break;**

**case "Unsuccess":**

**total = Menu1.Items[7].Text;**

**break;**

**}**

**this.btnPagerPrev.ImageUrl = "~/images/prev.gif";**

**this.btnPagerPrev.Enabled = true;**

**int stopTotal = Convert.ToInt32(total);**

**string stop = lblStopRec.Text;**

**int next = Convert.ToInt32(DropDownList1.SelectedItem.Text) + Convert.ToInt32(stop);**

**lblStopRec.Text = Convert.ToString(next);**

**if (next >= stopTotal)**

**{**

**lblStopRec.Text = Convert.ToString(stopTotal);**

**this.btnPagerNext.ImageUrl = "~/images/nextdisab.gif";**

**this.btnPagerNext.Enabled = false;**

**}**

**lblStartRec.Text = stop;**

**GridView1.DataBind();**

**}**

**protected void btnPagerPrev\_Click(object sender, ImageClickEventArgs e)**

**{**

**this.btnPagerNext.ImageUrl = "~/images/next.gif";**

**this.btnPagerNext.Enabled = true;**

**string stop = lblStartRec.Text;**

**string start = Convert.ToString(Convert.ToInt32(lblStartRec.Text) - Convert.ToInt32(DropDownList1.SelectedItem.Text));**

**if (Convert.ToInt32(start) <= 1)**

**{**

**this.btnPagerPrev.ImageUrl = "~/images/prevdisab.gif";**

**this.btnPagerPrev.Enabled = false;**

**}**

**lblStartRec.Text = start;**

**lblStopRec.Text = stop;**

**GridView1.DataBind();**

**}**

Esta paginación dependiendo de la pestaña que se encuentra activa. Cargara la fuente de datos del elemento datasource de webform y realiza la paginación del resultado de la consulta, esta es parame trizada por los tres parámetros generales de paginación, atrás , adelante y la opción de traer registros paginados desde base de datos, asi evitar carga datos ocultos en la página, si no se realiza esta paginación, cada query a pesar de que muestre cierta cantidad a cargado la cantidad completa de la consulta, la paginación resuelve esto ejecutando la consulta a la base de datos por tandas o tops.

Esta paginación se le aplica actualmente en los clientes a vistas que presentaban problemas de datos duplicados, la consulta realiza un filtrado de esta vista, pasando por medio de los sp los cuales son modificados mediante PL/SQL y pasado a un filtro que se encarga luego de realizar la paginación desde base de datos.

1. **Funciones Base de Datos**

* Función de Tabla - Split():

Se utiliza una función encargada de realiza una matriz, este se utiliza en las consultas que traen los listados de los boletines en su creación y edición

**ALTER FUNCTION [dbo].[Split] (@sep char(1), @s nvarchar(4000))**

**RETURNS table**

**AS**

**RETURN (**

**WITH Pieces(pn, start, stop) AS (**

**SELECT 1, 1, CHARINDEX(@sep, @s)**

**UNION ALL**

**SELECT pn + 1, stop + 1, CHARINDEX(@sep, @s, stop + 1)**

**FROM Pieces**

**WHERE stop > 0**

**)**

**SELECT pn,**

**SUBSTRING(@s, start, CASE WHEN stop > 0 THEN stop-start ELSE 512 END) AS s**

**FROM Pieces**

**)**

* **Funciones Escalares**

Existen Varias funciones la cuales se utilizan en los diferentes módulos de consulta del webmarketing como son:

* fn\_ValidaEmail(Función de validación de correos)
* GetListadosForBoletin(Listados cargados en el módulo de boletines de la aplicación)
* GetListadosForEmail(Listados por correo)
* GetSMTPForBoletin(Los multiples SMTP que trabajaran con el boletín programado)