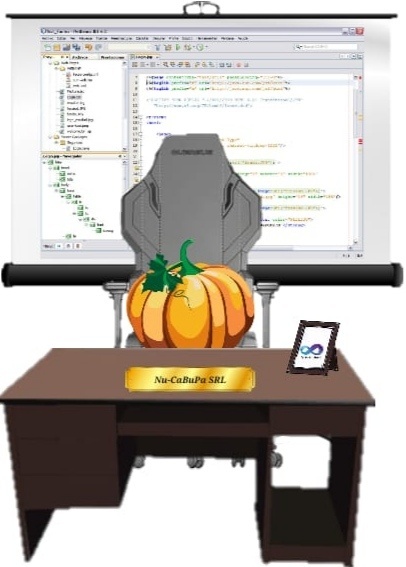
**Software de administración para la empresa**

**JVR Producciones**



Nu-CaBuPa. Software dedicado

Integrantes:

* Milagros Núñez
* Mateo Cabral
* ManuelBuslón
* Nahuel Pacheco

2020 Proyecto Integrador

Índice:

[Introducción: 5](#_Toc54946203)

[Nuestra empresa: 7](#_Toc54946204)

[Fundamentación 9](#_Toc54946205)

[Fundamentación: 10](#_Toc54946206)

[Objetivos 11](#_Toc54946207)

[Objetivos del proyecto: 12](#_Toc54946208)

[Misión: 12](#_Toc54946209)

[Visión: 12](#_Toc54946210)

[Objetivos como empresa: 13](#_Toc54946211)

[Analisis de los requerimientos 14](#_Toc54946212)

[Análisis de los requerimientos: 15](#_Toc54946213)

[Estudio de factibilidad 17](#_Toc54946214)

[Recurso de la empresa: 18](#_Toc54946215)

[Estudio de costo y precio de venta: 21](#_Toc54946216)

[Opciones de venta: 22](#_Toc54946217)

[Matriz Foda 23](#_Toc54946218)

[FODA 24](#_Toc54946219)

[Estrategias: 25](#_Toc54946220)

[Organización de la empresa 26](#_Toc54946221)

[Organización de la empresa 27](#_Toc54946222)

[Analisis de la entrevista 28](#_Toc54946223)

[Análisis de la entrevista: 29](#_Toc54946224)

[Base de Datos 30](#_Toc54946225)

[Base de datos 31](#_Toc54946226)

[MER: 31](#_Toc54946227)

[Pasaje a tablas 32](#_Toc54946228)

[Consultas de creación MySql 33](#_Toc54946229)

[Creación de usuario 36](#_Toc54946230)

[Código fuente 37](#_Toc54946231)

[Código: 38](#_Toc54946232)

[Modulos: 39](#_Toc54946233)

[Manual de usuario 125](#_Toc54946234)

[Presentación del manual: 126](#_Toc54946235)

[Manual: 127](#_Toc54946236)

[Medidas de Seguridad 132](#_Toc54946237)

[Medidas de seguridad 133](#_Toc54946238)

[Infraestructura de la empresa – Red de la empresa 135](#_Toc54946239)

[Configuración de red: 136](#_Toc54946240)

[Planos de la distribución: 136](#_Toc54946241)

[Conexión a la red: 136](#_Toc54946242)

[Configuración de router: 139](#_Toc54946243)

[Anexo 141](#_Toc54946244)

[S.R.L. 142](#_Toc54946245)

[Entrevista: 148](#_Toc54946246)

[Consultas de SQL 150](#_Toc54946247)

[Consultas de SQL 151](#_Toc54946248)

[Documento comercial 154](#_Toc54946249)

[Proyecto S.O 155](#_Toc54946250)

# 

Introducción

# Introducción:

El proyecto será desempeñado utilizando los conocimientos adquiridos en los años previos de estudio en el bachillerato de informática. También posee variedad de materias involucradas, las cuales son: Programación, Proyecto, Análisis y Desarrollo de Aplicaciones, Base de Datos, Formación Empresarial y Taller.

Nuestra empresa se definiría bajo el nombre “Nu-CaBuPa.SRL” y tendría su base en Uruguay, en el departamento de Salto. La misma se desarrollaría en el área tecnológica entorno a la producción de software dedicado desde fines administrativos hasta aplicaciones de entretenimiento.

El programa debe ser solo y únicamente para una empresa que necesite de este y pueda facilitarle tareas, las cuales serán indicadas por la empresa misma al igual que la organización del programa en caso de que el cliente desee especificarlo.

Nuestro software será diseñado para la empresa "JVR Producciones" y su propósito será enfocado en la administración y control de datos.

Nuestra empresa

**Nombre:** Nu-CaBuPa.SRL



**Nombre social:**3M&N

**Dirección:**25 Av. Feliciano Viera (Esquina Uruguay y Brasil)

**Teléfono:**4733 5987 - +598 980 235 001

**E-mail:**[NuCaBuPa@gmail.com](mailto:NuCaBuPa@gmail.com)

**Socios:** Manuel Buslón, Nahuel Pacheco, Milagros Nuñez, Mateo Cabral

# Nuestra empresa:

Nu-CaBuPa.SRL, es una empresa de tecnología informática (Desarrollo de Software, reparación, mantenimiento y venta de Hardware) fundada en marzo del año 2020, que tiene como objetivo la satisfacción de las necesidades de servicios tecnológicos informáticos de los clientes.La empresa cuenta con varios factores que le permiten el alcance del objetivo, uno de ellos es la utilización de tecnologías avanzadas, junto a los conocimientos de los técnicos abarca los entornos tecnológicos de: Base de datos (MySQL) y los lenguajes de programación de Java y .Net.

La forma jurídica que está compuesta la empresa Nu-CaBuPa, es de una SRL (Sociedad de Responsabilidad Ilimitada) debido a que los socios cumplen con los siguientes puntos:

* Los socios no deberían tener embargos inscriptos.
* La Administración estará a cargo de los socios en la forma que ellos mismos acuerden, por ejemplo en forma conjunta o indistinta, etc.
* El giro puede ser el estándar y particularmente se incluye como principal, la actividad más importante que se va a realizar.
* Las Sociedades se entregan inscriptas en DGI y BPS, debiéndose establecer para BPS, cual es el socio que aporta y que va a tener actividad.
* Los aportes a BPS se pueden realizar por uno solo de los socios, que deberá ser el socio que declare actividad.
* Los socios serán responsables de las deudas sociales, hasta el máximo del capital social, respondiendo con sus propios bienes por las obligaciones de la sociedad.
* Los socios permanecerán en la sociedad y solo por cesión de sus cuotas sociales podrán dejar de participar en la misma. Las cesiones de cutas también se inscriben en el registro nacional de comercio y se publican en Diario Oficial y particular.

## 

## Fundamentación

## Fundamentación:

A instancias de la necesidad por parte de las empresas de optimizar sus actividades han surgido proyectos de software de forma masiva. Cada uno adecuándose a las necesidades de un público específico o a un particular.

Este proyecto surge con los mismos fundamentos sobre los cuales emergen conjuntamente los demás, la posibilidad de volver más eficiente cualquier control e interacción entre los funcionarios y la gran cantidad de información a la que se enfrentan, en conjunto con las herramientas que estos deben utilizar continuamente.

Este software que será desarrollado, más que permitir una reducción en recursos y tiempo utilizado, lograrían tener una visión más amplia y objetiva de todas las actividades en las que se desarrolla la empresa. Se lo puede considerar directamente como una inversión realizada en el momento que se empieza el desarrollo, siguiendo esta visión, el software estará manteniendo una mejora continua en el tiempo sobre la utilización de recursos y gastos que gracias a este ya se volverían innecesarios.

En el sector de administración de recursos, lograría una automatización en la contabilidad de ganancias y un registro de las fuentes de ingreso en cada sector en la que se desarrolle. Permitiendo a dicho cliente un ahorro masivo en recursos tan valiosos como puede serlo el tiempo. Un ejemplo de este punto puede ser el acceso a variada cantidad de información almacenada ahora en servidores, pudiendo realizar consultas en un tiempo exponencialmente menor que el utilizado anteriormente. Una consecuencia directa de este ejemplo es la visibilidad de dicha información para el personal autorizado a acceder a ella, sin trabas externas a la empresa debido a las herramientas y la posibilidad de disminuir casi totalmente la perdida de información o destrucción de la misma por errores debido al método de almacenamiento empleado.

En conjunto al avance del tiempo y las tecnologías más actuales que se empleará a futuro el software contaría con la posibilidad de evolucionar y adaptarse al contexto, también contar con requisitos futuros en los que se deberá desempeñar. Dicha información almacenada y estructura en la codificación está regida por los paradigmas de la programación empleados en el mercado actual. Trayendo esto consigo una gran posibilidad de migración a diferentes software, remodelado de apartados en el programa y exportación de datos.

Como filosofía de este proyecto tenemos la utilización de una amplia gama de conocimientos, patrones de diseño y algoritmos estudiados y probados en entornos para el perfeccionamiento del programa en gran cantidad de casos que podrían ocurrir. Sumado a la correcta realización de pruebas y aplicación de diferentes herramientas en el desarrollo el software, contaría con el mejor desempeño y optimización adecuado a las necesidades del cliente. Todo esto permitiendo cambios futuros del mismo para obtener un mejor desempeño en las áreas que sean requeridas.

## 

## Objetivos

## Objetivos del proyecto:

Objetivos generales:

* Creación de un software administrativo, accesible para el usuario, que garantice una organización adecuada en la búsqueda y administración de datos.

Objetivos específicos:

* Diseñar una interfaz a medida y agradable para las necesidades del usuario.
* Crear un sistema de seguridad eficiente para la información ingresada.
* Conector adecuado entre la base de datos y el programa.
* Optimizar el tiempo de búsqueda de los datos solicitados.
* Software capacitado para la actualización de datos.
* Manejo adecuado de las publicidades emitidas por la radio “JVRProduciones”
* Software de bajos requerimientos.
* Utilización correcta y adecuada de los principios y bases de la ingeniería de software.

# Misión:

La empresa Nu-CaBuPa es una empresa que se dedica a la creación de software, mantenimientos de equipos, servicios técnicos y ventas de insumos informáticos, que brinda e implementa nuevos métodos y nuevas tecnologías para ofrecer soluciones al desarrollo en el área informática de empresas y clientes, abarcando todo el territorio uruguayo.

# Visión:

La empresa prende ser un referente internacional al momento que las personas y empresas busquen una solución informática. Para ello Nu-CaBuPa abordará las mejores soluciones y las formas de servicios que se ofrecen en la actualidad, brindando confianza y calidad al momento de elegirlos

# Objetivos como empresa:

General:

Como empresa deseamos alcanzar un nivel de referencia internacional en cuanto a tecnología se refiere. Solucionar los problemas que el mundo tecnológico presente y crear nuevos estándares. Mantenernos actualizados a los últimos procedimientos informáticos e indirectamente iniciando nuevos avances debido a la necesidad del cambio. Siendo reconocida no por la velocidad, ni cantidad, si no calidad y el impacto.

Específicos:

-Mejorar la eficiencia de los programas y tecnologías utilizadas con cada nuevo proyecto.

-Fomentar un área de trabajo saludable y participativa entre todo el personal.

-Expandirse a otros mercados en el sector de software.

-Diseñar un modelo empresarial rentable para el futuro.

-Tener una estrategia de marketing en caso de ser necesario.

-Atraer más socios capacitados que ayuden a la empresa y traigan su propia visión con ellos.

## Analisis de los requerimientos

## Análisis de los requerimientos:

Usuarios:

|  |  |
| --- | --- |
| Definición de requisitos de usuario: | Especificación de requisitos del sistema: |
| El consumidor debe poder tener un control sobre los distintos usuarios y sus permisos. | El usuario poseería campos para seleccionar los distintos permisos que se tengan además de poder controlar los nombres y contraseñas (si posee los permisos necesarios). |

Programa:

|  |  |
| --- | --- |
| El usuario podría administrar el precio de la cuota de mensual de cada programa. | Se podría elegir entra varias opciones de precios ya establecidos o agregar nuevos precios para otras categorías/programa. |
| El programa poseería fechas en las que se emiten y estas podrían ser controladas. | Se mostrarían en una lista las fechas ya establecidas y por medio de un botón se podría seleccionar en un calendario para agregar otra fecha.  Seleccionando una fecha de la lista se podría eliminar o modificar. |
| En el programa es posible que se muestren publicidades y se permitiría mantener el control sobre cuáles y hasta que fecha serían emitidas. | En una interfaz se podría seleccionar al programa que se desearía mostrar la publicidad y cuál de estas sería. |
| Se podría definir que un evento está relacionado con un programa. | Sería posible enlazarlo por medio de seleccionarlos de dos listas. |

Funcionarios:

|  |  |
| --- | --- |
| Se podría definir las funciones de cada funcionario. | A partir de seleccionar de un perfil del mismo dichas funciones (O agregarla si no se han creado con anterioridad). |
| El usuario puede establecer que funcionario y cumpliendo qué función tendría cada programa. | A base del funcionario y de la función se marca en que programa la ejercería. |

Publicidad:

|  |  |
| --- | --- |
| La publicidad tiene cuotas que se deben pagar y se registrarían las fechas. | Ingresando en el administrador de publicidades se pueden verificar las distintas cuotas a partir de una tabla, en la que se mostrar |
| Todas las publicidades pertenecen a alguna empresa y se debe poder tener registro de acual. | El administrador de empresa se mostraría las diferentes publicidades pertenecientes de dicha empresa en el orden en el cual se agregaron.  Se permitiría realizar una búsqueda por tema. |
| En la radio hay distintas tandas en las que pueden aparecer publicidades. | Permitiría administrar las tandas junto a las publicidades que aparecerían en la misma. Junto con un control de la fecha que empezarían a emitirse, junto a la fecha de finalización. |
| En los eventos se muestran publicidades y sería conveniente tener un registro de las mismas. | Mediante la unión a bese de la selección en listas (o buscando por temas) se relacionarían la publicidad con el evento. |

Videos:

|  |  |
| --- | --- |
| El usuario tendría el control de los videos y de las series, junto con la pertenencia de los mismos a las series. | Una vez agregado el video (el nombre y el tema) se podría seleccionar una serie existente a la cual pertenecería el mismo. |
| Sería posible que un evento tratara de un video de una serie y sería conveniente conocer los datos de ambos. | En una interfaz del control del evento habría una tabla con los videos relacionados permitiendo añadir, modificar y eliminar los mismos. |

# Estudio de factibilidad

# Recurso de la empresa:

Recursos humanos:

Desarrolladores:

* Manuel Buslón
* Nahuel Pacheco
* Milagros Nuñez
* Mateo Cabral

Recursos materiales:

Fundamentales:

1. PC’s
2. Routers
3. Impresoras
4. Laptops

Extras:

1. Escritorios
2. Sillas
3. Monitores
4. Estantes
5. Repuestos de teclado y mouse

Recursos financieros:

Para el inicio de la empresa y la recaudación de los recursos materiales, los socios aportarán una suma de $USD2494,66, la cual se dividirá entre los cuatro, aportando cada uno el monto de $USD623,665

Cada socio tomará la decisión de sacar un préstamo con la suma demandada para la recaudación.

|  |  |  |
| --- | --- | --- |
| **Descripción** | **Precio unitario** | **Precio total** |
| Router | $2.551 | $2.551 |
| Impresora | $5.102 | $5.102 |
| Laptop | $8.427 | $16.854 |
| PC | $21.091 | $42.182 |
| Monitor | $5.655 | $16.965 |
| Teclado | $510 | $3.060 |
| Mouse | $595 | $3.570 |
| Nic | $1.133 | $2.266 |
| Rosetas | $315 | $2.106 |
| Licencia Visual Studio 2019 | $1.199 | $4.476 |
| Licencia Windows 10 Pro | $1500 | $4.500 |
| **Total** | $106.432 |

Recursos tenoclógicos

|  |  |
| --- | --- |
| Router x1 | TP-Link Archer C60: Análisis de este router doble banda Wi-Fi AC1350 |
| Impresora x1 | IMPRESORA EPSON L4160 Multifunción |
| PC (Personalizadas) | Procesador Intel Core I3 7100  ● Placa Madre Gigabyte H110m-h  ● 4GB Ddr4 2133 Mhz Kingston  ● Fuente Atx P4 500w  ● Disco Duro Wd Blue 1TB Sata 7200 Rpm  ● Gabinete Negro Con Usb Y Audio Frontales  ● Ventilador 12 cm Corsair Air Af120 |
| Laptop x2 | Lenovo IdeaPad L340 (15) | 15" Laptop with Dolby Audio™ & TrueBlock Privacy  Shutter | Lenovo US |
| Monitores x3 | Monitor Curvo pantalla LED 24” serie F390 | SAMSUNG |
| Teclado x6 | Teclado Logitech K120 USB en Español | Macrotec |
| Mouse x6 | Logitech G203 Lightsync mouse gamer blanco - Thot Computación |
| NIC, 1 Puerto. X2 | Tarjeta De Interfaz De Red Pci-e Gigabit Ethernet Nic 10/100 - $ 1.133,00  en Mercado Libre |
| Rosetas modelo doble x 6 | ROSETA RED CAT5E RJ45 2 PUERTOS BLANCA CON SISTEMA DE BLOQUEO PRO – 2  puertos, UTP, Caja de montaje, Función de bloqueo, Blanca – 790826 – Icotech |
| Licencia Windows 10 Pro x4 | X1 año |
| Licencia Visual Studio 2019 | X1 año |

# Estudio de costo y precio de venta:

Presupuesto:

Por mes

|  |  |
| --- | --- |
| **Descripción** | **Importe** |
| Alquiler | $5.050 |
| UTE | $2.500 |
| ANTEL | $1.512 |
| OSE | $900 |
| Seguro | $4.000 |
| Salario | $128.640 |
| **Total** | $142.602 |
| **Total, en dólares** | $USD3328,56 |

|  |  |
| --- | --- |
| Remuneración mensual nominal | $ 40.000,00 |
| Remuneración mensual líquida | $ 32.160,00 |
| Seguro de Salud | 15 - Beneficiarios s/hijos y s/cony-conc |
|  |  |

Detalle de Aportes BPS

|  |  |  |  |
| --- | --- | --- | --- |
| Aporte | Aporte Personal | Aporte Patronal | Totales |
| Jubilatorio | $ 6.000,00 | $ 3.000,00 | $ 9.000,00 |
| Fondo de Garantía Créditos Laborales | $ 0,00 | $ 10,00 | $ 10,00 |
| Fonasa | $ 1.800,00 | $ 2.000,00 | $ 3.800,00 |
| F.R.L. | $ 40,00 | $ 40,00 | $ 80,00 |
|  | **$ 7.840,00** | **$ 5.050,00** | **$ 12.890,00** |

# Opciones de venta:

1- Software completo: $67.000 ($USD 1390,04)

2- Software completo + instalación: $68.00 ($USD 1413,60)

3- Software completo + instalación + manual: $68.250 ($USD 1425,38)

4- Software completo + instalación + servis por 9 meses: $69.000 ($USD 1425,38)

5- Servis del Software: $700 ($USD 16,49)

Sueldo hora/hombre= $40.000/30= $1.333/8=$166

Horas/mes= 80h

Sueldo por horas= 80hx$166= 13.280x5 (meses)=$66.400

# 

# Matriz Foda

# FODA

|  |  |  |
| --- | --- | --- |
|  | Interno | Externo |
| Ventaja: | Fortalezas:   * Conocimientos sobre desarrollo de aplicaciones y experiencia en el mismo. * Buena comunicación y organización interna. * Metas y objetivos claros. * Precios ajustados al producto y al trabajo requerido. * Clase y singularidad en los trabajos realizados. * Costo casi inexistente de producción de una vez finalizado el producto. | Oportunidad:   * Facilidad de crecimiento como empresa debido al rubro. * Necesidad por parte de las empresas de aplicaciones y servicios. * Mercado objetivo claro. * Facilidad para distribución del producto. |
| Desventajas: | Debilidades:   * No tener una fuerte relación previa con nuestros clientes. * Poca experiencia en el mercado. * Capital inicial inexistente. * Incapacidad de aceptar varios proyectos simultáneamente. * Poco conocimiento del entorno sobre la factibilidad y posibilidad del software. | Amenazas:   * Gran competencia en el rubro. * Crisis mundial. * Económica en estado de ahorro y en estado de inversiones. |

# Estrategias:

* Comenzar con proyectos pequeños, que ayuden a comprender y obtener experiencia en el mercado laboral
* Sacar préstamos para comenzar con la compra de los recursos materiales y tecnológicos necesarios.
* Luego de obtener los conocimientos básicos, pordemos capacitarnos para poder aceptar más proyectos, además de que en un futuro contratar a personas capacitado.
* Ser una empresa innovadora, estando al tanto de las nuevas tecnologías.
* Ser una empresa que sus precios sean llamativos para la demanda.

# Organización de la empresa

# Organización de la empresa



Directiva de la empresa:

* Manuel Buslón
* Nahuel Pacheco
* Milagros Nuñez
* Mateo Cabral

Divisiones:

Programación:

* Manuel Buslón
* Nahuel Pacheco

Mantenimiento:

* Nahuel Pacheco
* Milagros Nuñez
* Manuel Buslón

Admnistración:

* Nahuel Pacheco, Milagros Nuñez, Mateo Cabral

# 

## 

## Analisis de la entrevista

## Análisis de la entrevista:

La entrevista a nuestro cliente JVR Producciones fue realizada por medio de la plataforma zoom el día 06 del 04 por motivos de una emergencia sanitaria. Esta fue planteada con anticipación, planeando con el equipo nuestra presentación y una metodología, la cual nos serviría para la obtención de datos y que a raíz de eso generaríamos más preguntas para la recolección de todos los requerimientos deseados por el cliente para poder realizar el software a medida.

Las primeras preguntas que fueron realizadas dieron un enfoque a la obtención de información sobre la empresa, para la realización del programa nuestro equipo tiene que estar en conocimientos sobre las actividades que realiza nuestro cliente y como estuvo administrando los datos hasta el momento, para poder identificar los métodos que este tenía. Por medio de la respuesta del cliente pudimos establecer que sus métodos de administración son bastante ineficientes, ya que estos demandan una buena organización por parte del mismo, además de que este método utilizado puede generar retrasos, errores y posibles inconvenientes.

Al momento de la obtención de la información el equipo comenzó a realizar preguntas relacionadas con el interés del cliente que se basaban en la gestión del programa de tv y una administración de control con relación a la radio, para poder comenzar a sugerir ciertas propuestas para la realización del programa dando sugerencias para su creación final.

A medida que se iba obteniendo información, se comenzó a consultar sobre los requisitos del programa, realizando preguntas especificas que estuvieran conectadas con la información que el cliente nos brindaba. Preguntando cómo es que se hacían ciertas actividades al momento de la administración, como es que se realizaban esas actividades y haciendo un énfasis en el interés del cliente, dando sugerencias y algunas explicaciones al momento que este pedía algo en específico.

Para concluir con la recolección de información necesaria, se comenzó a preguntar sobre el entendimiento que el cliente tenía sobre el área, además de preguntar sobre quienes estarían utilizando el programa cuando esté finalizado, para poder saber cómo crear el manual de usuario, además de conocer cuáles eran los equipos que este manejaba, ya que el programa será utilizado tanto en la radio como en la casa del cliente.

Al finalizar se concordaron ciertas especificaciones para el diseño que el cliente optó por propias.

# 

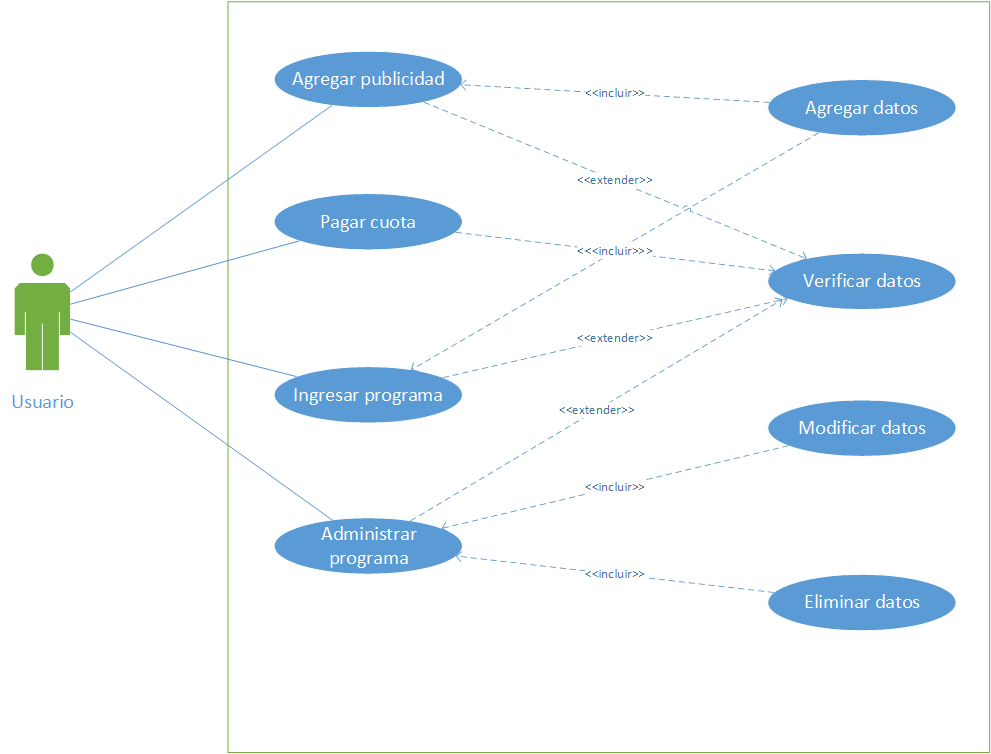
# Diagrama de flujo de datos

## 

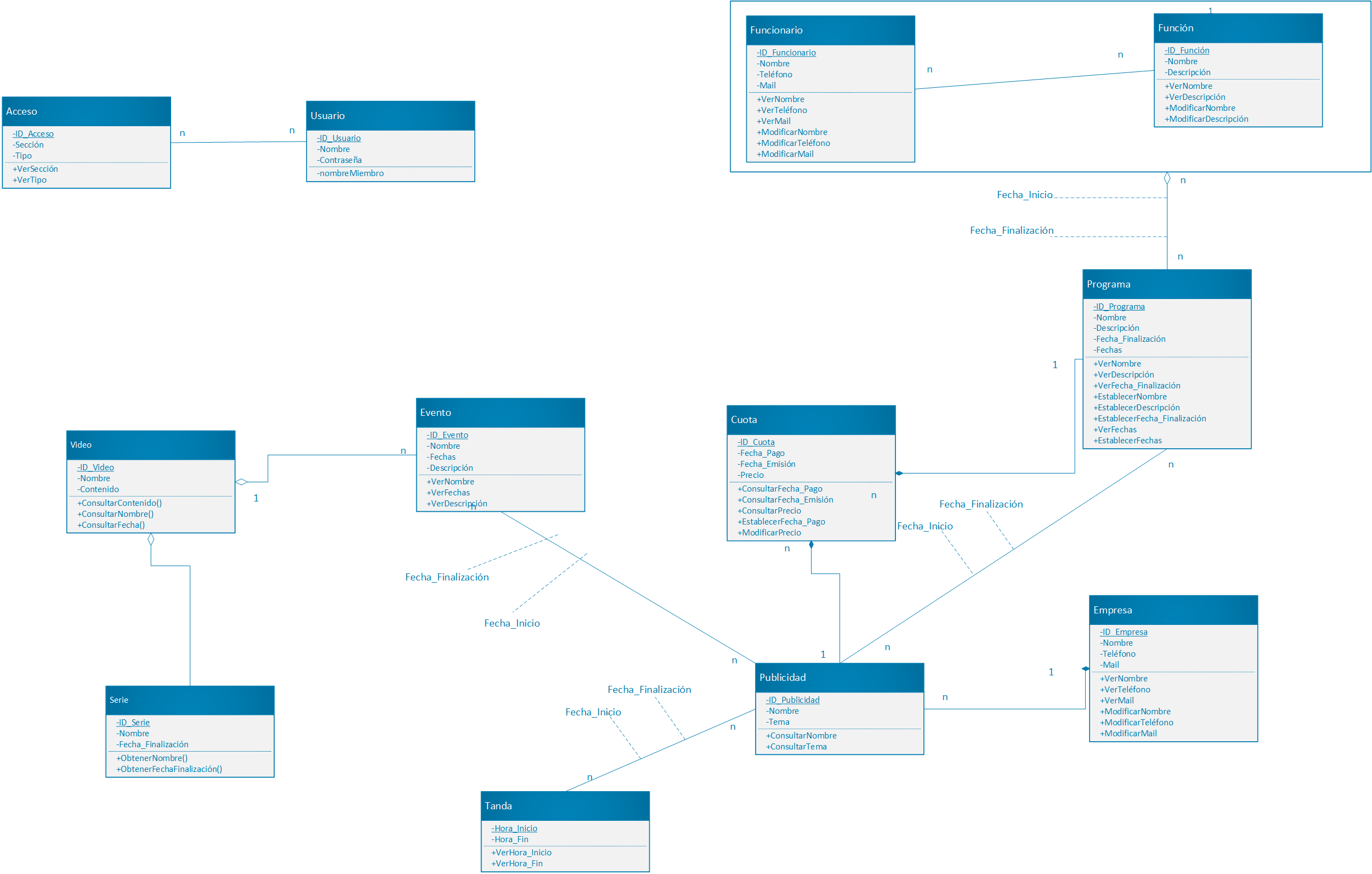
## 

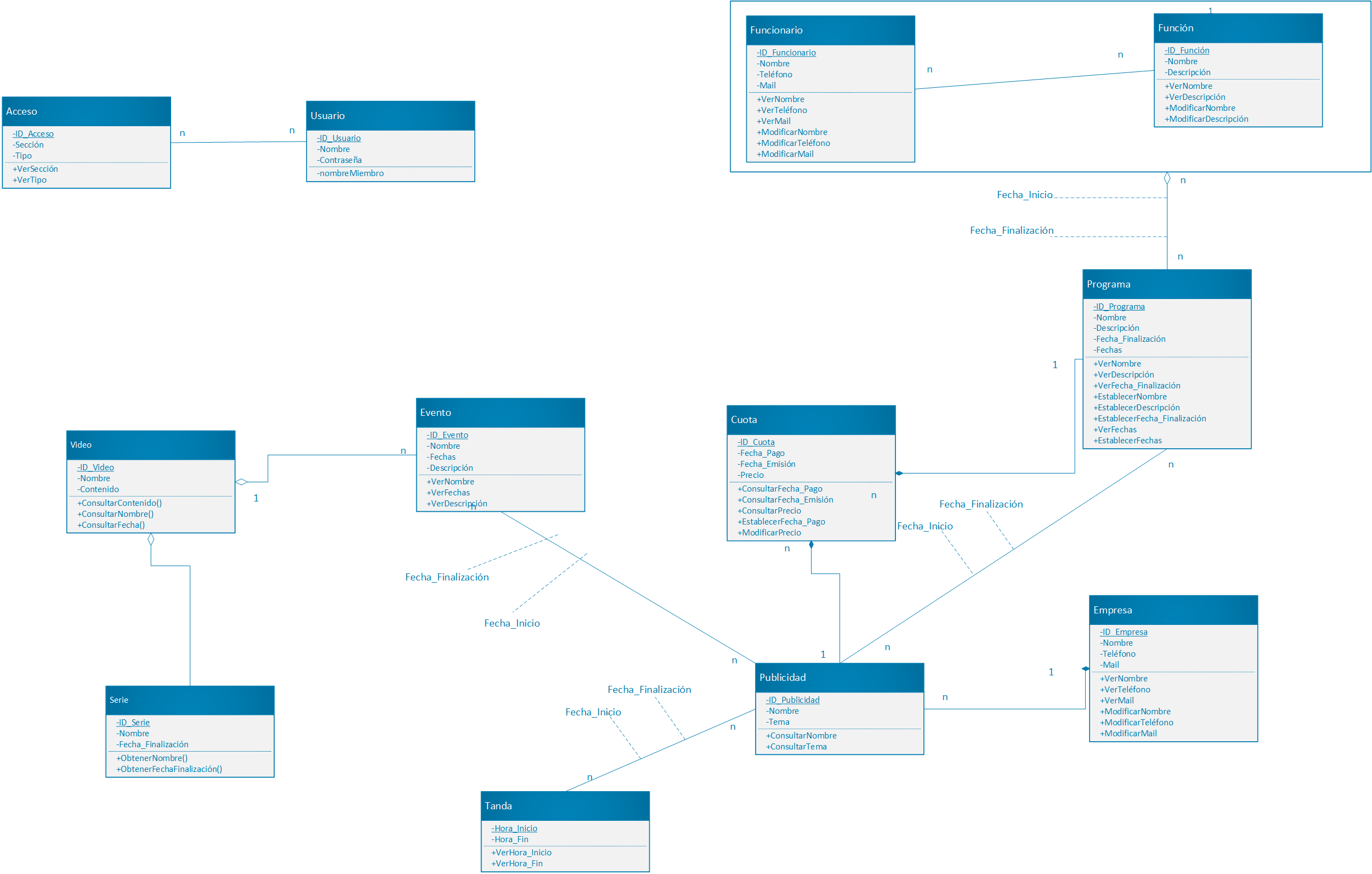
## 

# Casos de uso



# UML de clases





# Base de Datos

# Base de datos

## MER:

|  |
| --- |
| Pasaje a tablas |
| Programa: { ID\_Programa , Fecha\_Finalizacion, Nombre\_Programa, Descripción}  FechaPrograma: { Fecha , Hora\_Inicio , Hora\_Fin, ID\_Programa}  ProgramaCuota: { ID\_Programa\_Cuota, ID\_Programa , Fecha\_Pago, Fecha\_Emision, Precio}  Publicidad: { ID\_Publicidad ,Nombre, Tema, ID\_Empresa}  Usuarios: {ID\_usuario, Nombre, Contrasena}  Acceso: {id\_acceso, Seccion, Tipo}  UsTieneAcceso:{id\_usuario, id\_acceso}  PMuestraPubli: {ID\_Publicidad, ID\_Programa , Fecha\_Finalizacion, Fecha\_Inicio}  PublicidadCuota: {ID\_Cuota, Fecha\_Pago, Fecha\_Emision, Precio, ID\_Publicidad}  Empresa:{ID\_Empresa , Nombre, Telefono, Mail}  Tanda: { Hora\_Inicio, Hora\_Fin}  AparecePubli: { ID\_Publicidad , Hora\_Inicio , Fecha\_Inicio, Fecha\_Finalizacion}  Evento: { ID\_Evento , Nombre, Descripcion, ID\_Video }  FechaEvento: { ID\_Evento, Fecha }  EventoMuestraPubli: { ID\_Evento , ID\_Publicidad }  Funcionario: { ID\_Funcionario , Telefono, Nombre, Mail}  Funcion: { ID\_Funcion , Descripcion, Nombre}  TrabajaComo: { ID\_Funcionario, ID\_Funcion }  FunTrabaja: { ID\_Funcionario, ID\_Funcion, ID\_Programa , Fecha\_Inicio, Fecha\_Finalizacion}  Video: { ID\_Video , Contenido, Nombre, ID\_Serie, Fecha}  Serie: { ID\_Serie ,Fecha\_Finalizacion, Nombre} |

|  |
| --- |
| Consultas de creación MySql |
| CREATE TABLE `funcion` (  `ID\_Funcion` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `descripcion` varchar(128) NOT NULL,  Nombre varchar(16) not null,  PRIMARY KEY (ID\_Funcion)  );  CREATE TABLE `funcionario` (  `ID\_Funcionario` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `Telefono` varchar(16) DEFAULT NULL,  `nombre` varchar(48) NOT NULL,  `Mail` varchar(64) DEFAULT NULL,  PRIMARY KEY (`ID\_Funcionario`)  );  CREATE TABLE `acceso` (  `id\_acceso` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `seccion` varchar(8) NOT NULL,  `tipo` varchar(2) NOT NULL,  PRIMARY KEY (`id\_acceso`)  );  CREATE TABLE `serie` (  `ID\_Serie` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `fecha\_finalizacion` date DEFAULT NULL,  `nombre` varchar(48) NOT NULL,  PRIMARY KEY (`ID\_Serie`)  );  CREATE TABLE `video` (  `ID\_Video` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `contenido` varchar(128) NOT NULL,  `nombre` varchar(48) NOT NULL,  `ID\_Serie` int(6) unsigned DEFAULT NULL,  `fecha` date DEFAULT NULL,  PRIMARY KEY (`ID\_Video`),  Foreign KEY (ID\_Serie) references Serie(`ID\_Serie`)  );  CREATE TABLE `evento` (  `ID\_Evento` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `nombre` varchar(32) NOT NULL,  `descripcion` varchar(128) NOT NULL,  `ID\_Video` int(6) unsigned,  PRIMARY KEY (`ID\_Evento`),  Foreign KEY (ID\_Video)references Video(`ID\_Video`)  );  CREATE TABLE `empresa` (  `ID\_Empresa` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `Nombre` varchar(64) NOT NULL,  `Telefono` varchar(16) DEFAULT NULL,  `Mail` varchar(64) DEFAULT NULL,  PRIMARY KEY (`ID\_Empresa`)  );  CREATE TABLE `publicidad` (  `ID\_Publicidad` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `Nombre` varchar(32) NOT NULL,  `Tema` varchar(64) NOT NULL,  `ID\_Empresa` int(6) unsigned NOT NULL,  PRIMARY KEY (`ID\_Publicidad`),  Foreign KEY (ID\_Empresa) references Empresa(`ID\_Empresa`)  );  CREATE TABLE `usuarios` (  `id\_usuario` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `nombre` varchar(16) NOT NULL,  `contrasena` varbinary(256) NOT NULL,  PRIMARY KEY (`id\_usuario`)  );  CREATE TABLE `tanda` (  `Hora\_Inicio` time NOT NULL,  `Hora\_Fin` time NOT NULL,  PRIMARY KEY (`Hora\_Inicio`)  );  CREATE TABLE `programa` (  `ID\_Programa` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `Nombre\_Programa` varchar(48) NOT NULL,  `Descripcion` varchar(128) DEFAULT NULL,  `Fecha\_Finalizacion` datetime DEFAULT NULL,  PRIMARY KEY (`ID\_Programa`)  );  CREATE TABLE `aparecepubli` (  `ID\_Publicidad` int(6) unsigned NOT NULL,  `Hora\_Inicio` time NOT NULL,  `Fecha\_Inicio` date NOT NULL,  `Fecha\_Finalizacion` date NOT NULL,  PRIMARY KEY (`ID\_Publicidad`,`Hora\_Inicio`,`Fecha\_Inicio`),  Foreign KEY (Hora\_Inicio) references Tanda(Hora\_Inicio),  Foreign key (ID\_Publicidad) references Publicidad(ID\_Publicidad)  );  CREATE TABLE `eventomuestrapubli` (  `ID\_Evento` int(6) unsigned NOT NULL,  `ID\_Publicidad` int(6) unsigned NOT NULL,  `Fecha\_Inicio` date NOT NULL,  `Fecha\_Finalizacion` date NOT NULL,  PRIMARY KEY (`ID\_Evento`,`ID\_Publicidad`,`Fecha\_Inicio`),  Foreign KEY (ID\_Publicidad) references Publicidad(`ID\_Publicidad`),  Foreign KEY (ID\_Evento) references Evento(ID\_Evento)  );  CREATE TABLE `fechaevento` (  `ID\_Evento` int(6) unsigned NOT NULL,  `fecha` datetime NOT NULL,  PRIMARY KEY (`ID\_Evento`,`fecha`),  Foreign key (ID\_Evento) references Evento(ID\_Evento)  );  CREATE TABLE `fechaprograma` (  `Fecha` date NOT NULL,  `Hora\_Inicio` time NOT NULL,  `Hora\_Fin` time NOT NULL,  `ID\_Programa` int(6) unsigned DEFAULT NULL,  PRIMARY KEY (`Fecha`,`Hora\_Inicio`),  Foreign KEY (ID\_Programa) references Programa(`ID\_Programa`)  );  CREATE TABLE `trabajacomo` (  ID\_TrabajaComo int(6) unsigned not null auto\_increment,  `ID\_Funcionario` int(6) unsigned NOT NULL,  `ID\_Funcion` int(6) unsigned NOT NULL,  PRIMARY KEY (ID\_TrabajaComo),  Foreign KEY (ID\_Funcion) references Funcion(`ID\_Funcion`),  Foreign KEY (ID\_Funcionario) references Funcionario(ID\_Funcionario)  );  CREATE TABLE `funtrabaja` (  `ID\_TrabajaComo` int(6) unsigned NOT NULL,  `ID\_Programa` int(6) unsigned NOT NULL,  `fecha\_inicio` date NOT NULL,  `fecha\_finalizacion` date DEFAULT NULL,  PRIMARY KEY (`ID\_TrabajaComo`,`ID\_Programa`,`fecha\_inicio`),  Foreign KEY (ID\_TrabajaComo) references Funcion(`ID\_Funcion`),  Foreign KEY (ID\_Programa) references Programa(`ID\_Programa`)  );  CREATE TABLE `pmuestrapubli` (  `ID\_Publicidad` int(6) unsigned NOT NULL,  `ID\_Programa` int(6) unsigned NOT NULL,  `Fecha\_Inicio` date NOT NULL,  `Fecha\_Finalizacion` date NOT NULL,  PRIMARY KEY (`Fecha\_Inicio`,`ID\_Programa`,`ID\_Publicidad`),  Foreign KEY (ID\_Programa) references Programa(`ID\_Programa`),  Foreign KEY (ID\_Publicidad) references Publicidad(`ID\_Publicidad`)  );  CREATE TABLE `programacuota` (  `ID\_Programa\_Cuota` int(6) unsigned NOT NULL auto\_increment,  `ID\_Programa` int(6) unsigned NOT NULL,  `Fecha\_Pago` date DEFAULT NULL,  `Precio` decimal(7,2) unsigned NOT NULL,  `Fecha\_Emision` date NOT NULL,  PRIMARY KEY (`ID\_Programa\_Cuota`),  Foreign KEY (ID\_Programa) references Programa(`ID\_Programa`)  );  CREATE TABLE `publicidadcuota` (  `ID\_PublicidadCuota` int(6) unsigned NOT NULL AUTO\_INCREMENT,  `Fecha\_Pago` date DEFAULT NULL,  `Precio` decimal(7,2) unsigned NOT NULL,  `ID\_Publicidad` int(6) unsigned NOT NULL,  `Fecha\_Emision` date NOT NULL,  PRIMARY KEY (`ID\_PublicidadCuota`),  Foreign KEY (ID\_Publicidad) references Publicidad(`ID\_Publicidad`)  );  CREATE TABLE `ustieneacceso` (  `id\_usuario` int(6) unsigned NOT NULL,  `id\_acceso` int(6) unsigned NOT NULL,  PRIMARY KEY (`id\_usuario`,`id\_acceso`),  Foreign KEY (id\_acceso) references acceso(`id\_acceso`)  );  REPLACE `acceso` VALUES  (1,'Inicio','a'),  (2,'Configuracion','a'),  (3,'Programa','a'),  (4,'Serie','a'),  (5,'Video','a'),  (6,'Empresa','a'),  (7,'Publicidad','a'),  (8,'Evento','a'),  (9,'Tanda','a'),  (10,'Tanda','v'),  (11,'Configuracion','v'),  (12,'Programa','v'),  (13,'Serie','v'),  (14,'Video','v'),  (15,'Serie','v'),  (16,'Empresas','v'),  (17,'Publicidad','v'),  (18,'Eventos','v'); |

|  |
| --- |
| Creación de usuario |
| CREATE USER 'Admin'@'localhost' IDENTIFIED BY 'jvrp';  GRANT USAGE ON JVRPDATABASE.\* TO 'Admin'@'localhost';  GRANT ALL PRIVILEGES ON JVRPDATABASE.\*TO 'Admin'@'localhost';  FLUSH PRIVILEGES;  CREATE USER 'User'@'localhost' IDENTIFIED BY 'usuario';  GRANT USAGE ON JVRPDATABASE.\* TO 'User'@'localhost';  GRANT SELECT, INSERT, UPDATE, DELETEON JVRPDATABASE.\*TO 'User'@'localhost';  FLUSH PRIVILEGES;  CREATE USER 'Muestra'@'localhost' IDENTIFIED BY 'Default';  GRANT USAGE ON JVRPDATABASE.\*TO 'Muestra'@'localhost';  GRANT select ON JVRPDATABASE.\* TO 'Muestra'@'localhost';  FLUSH PRIVILEGES; |

## 

## Código fuente

## Código:

Ingreso genérico de administrador:

Generador\_DB.vb

Imports MySql.Data.MySqlClient

Public Class Generador\_DB

Dim dt As New DataTable

Dim nTabla As String = ""

Private Sub Generador\_DB\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

cbTablas.DataSource = ESQLSelect("show tables")

cbTablas.ValueMember = "Tables\_in\_jvrpdatabase"

nTabla = "acceso"

End Sub

Private Sub btnCTabla\_Click(sender As Object, e As EventArgs) Handles btnCTabla.Click

ModLog.Guardar(nTabla)

For i As Integer = dgvTabla.Columns.Count - 1 To 0 Step -1

dgvTabla.Columns.RemoveAt(i)

Next

dt = ESQLSelect("describe " + nTabla)

dgvTabla.DataSource = dt

For i As Integer = dgvTabla.Columns.Count - 1 To 1 Step -1

dgvTabla.Columns.RemoveAt(i)

Next

dgvTabla.Columns(0).ReadOnly = True

dgvTabla.Columns.Add("Datos", "Datos")

dgvTabla.Columns.Add("Mantener", "Mantener")

End Sub

Private Sub btnInsertar\_Click(sender As Object, e As EventArgs) Handles btnInsertar.Click

Dim Consulta As String() = GeneradorDeString()

ModConector.ISQL(nTabla, Consulta(0), Consulta(1))

Limpiador()

End Sub

Public Sub Limpiador()

For i As Integer = 0 To dgvTabla.Rows.Count - 1

If IsNothing(dgvTabla.Rows(i).Cells(2).Value) Then

dgvTabla.Rows(i).Cells(1).Value = ""

End If

Next

End Sub

Private Function GeneradorDeString() As String()

Dim Texto As String() = {"", ""}

For i As Integer = 0 To dgvTabla.Rows.Count - 1

If dt.Rows(i).Item(3).ToString() = "no" Or Not String.IsNullOrWhiteSpace(dgvTabla.Rows(i).Cells(1).Value) Then

Texto(0) += dgvTabla.Rows(i).Cells(0).Value.ToString() + ","

Texto(1) += "'" + dgvTabla.Rows(i).Cells(1).Value.ToString() + "',"

End If

Next

Texto(0) = Texto(0).Remove(Texto(0).Length - 1)

Texto(1) = Texto(1).Remove(Texto(1).Length - 1)

ModLog.Guardar(Texto(0) + Texto(1))

Return Texto

End Function

Private Sub Limpiar\_Click(sender As Object, e As EventArgs) Handles Limpiar.Click

Limpiador()

End Sub

Private Sub cbTablas\_SelectedValueChanged(sender As Object, e As EventArgs) Handles cbTablas.SelectedValueChanged

If nTabla <> "" Then

nTabla = cbTablas.SelectedValue

End If

End Sub

End Class

# Modulos:

ModCodificador.vb:

Imports System.Text

Imports System.Security.Cryptography

'Se utiliza para encriptar y desencriptar datos de tipo string

'Usa una key y un vector de incializacion, que llamaremos key maestra

'Por mas información de esta ultima se puede buscar en:

'https://es.wikipedia.org/wiki/Vector\_de\_inicializaci%C3%B3n

Module ModCodificador

Private KeyMaestra As String = "rpaSPvIvVLlrcmtzPU9/c67Gkj7yL1S5"

Private Key As String = "12345678"

Private IV() As Byte

Private EncryptionKey() As Byte

Private buffer() As Byte

Private des As TripleDESCryptoServiceProvider

Public Sub EKey(ByVal NKey As String)

If Not String.IsNullOrEmpty(NKey) Then

Key = NKey

End If

End Sub

Public Sub Actualizar()

IV = ASCIIEncoding.ASCII.GetBytes(Key) 'La clave debe ser de 8 caracteres

EncryptionKey = Convert.FromBase64String(KeyMaestra) 'No se puede alterar la cantidad de caracteres pero si la clave

des = New TripleDESCryptoServiceProvider

des.Key = EncryptionKey

des.IV = IV

End Sub

Public Sub EKeyMaestra(ByVal NKeyMaestra As String)

If Not String.IsNullOrEmpty(NKeyMaestra) Then

KeyMaestra = NKeyMaestra

End If

End Sub

Public Function GKey() As String

Return Key

End Function

Public Function GKeyMaestra() As String

Return KeyMaestra

End Function

Public Function Encriptar(ByVal Texto As String) As String

'convierte el texto a bytes utilizando el formato utf8

buffer = Encoding.UTF8.GetBytes(Texto)

'Crea un objeto de encriptacion, luego crea un valor hash con el buffer y utiliza todo su largo, comenzando en el byte 0

Return Convert.ToBase64String(des.CreateEncryptor().TransformFinalBlock(buffer, 0, buffer.Length()))

End Function

Public Function Desencriptar(ByVal Texto As String) As String

'convierte el texto encriptado a bytes

buffer = Convert.FromBase64String(Texto)

'desencripta el texto(ahora en array de bytes) y utiliza el formato utf8

Return Encoding.UTF8.GetString(des.CreateDecryptor().TransformFinalBlock(buffer, 0, buffer.Length()))

End Function

End Module

ModConector.vb:

Imports MySql.Data.MySqlClient

Imports System.Data

Imports MySql.Data

Imports System.Drawing.Text

Module ModConector

Private Debug As Boolean = True

Private conn As New MySqlConnection

Private connStr As String

Private Address, User, Database, Port, Pass As String

Private UsuarioID As Integer

Private Usuario, Password As String

Private objCmd As New MySqlCommand

Private ds As New DataSet

Private dt As New DataTable

#Region "GetDirection"

Public Function GDT(ByVal Nombre As String) As DataTable

Return ds.Tables(Nombre)

End Function

Public Function GUsuarioID() As Integer

Return UsuarioID

End Function

Public Sub BorrarUsuario()

Usuario = ""

Password = ""

UsuarioID = Nothing

End Sub

Public Function GDebug() As Boolean

Return Debug

End Function

Public Function GAddress() As String

Return Address

End Function

Public Function GUser() As String

Return User

End Function

Public Function GDatabase() As String

Return Database

End Function

Public Function GPort() As String

Return Port

End Function

Public Function GPass() As String

Return Pass

End Function

#End Region

#Region "Constructor"

Public Sub Crear(ByVal RAddress As String, ByVal RPort As String, ByVal RDatabase As String, ByVal RUser As String, ByVal RPass As String)

If Not String.IsNullOrEmpty(Address) Then

Address = RAddress

Else

Address = "localhost"

End If

If Not String.IsNullOrEmpty(Address) Then

User = RUser

Else

User = "root"

End If

If Not String.IsNullOrEmpty(Address) Then

Database = RDatabase

Else

Database = "JVRPDB"

End If

If Not String.IsNullOrEmpty(Address) Then

Port = RPort

Else

Port = "3306"

End If

If Not String.IsNullOrEmpty(Address) Then

Pass = RPass

Else

Pass = "root"

End If

End Sub

Public Sub EAddress(ByVal NAddress As String)

Address = NAddress

End Sub

Public Sub EPort(ByVal NPort As String)

Port = NPort

End Sub

Public Sub EUser(ByVal NUser As String)

User = NUser

End Sub

Public Sub EPass(ByVal NPass As String)

Pass = NPass

End Sub

Public Sub EDatabase(ByVal NDatabase As String)

Database = NDatabase

End Sub

#End Region

#Region "Conectar"

Public Sub Inicio()

Try

connStr = "DataSource=" + Address + "; Port=" + Port + "; Database=" + Database + "; Uid=" + User + "; Pwd=" + Pass + "; CharSet=utf8"

'connStr = "Server=" + Address + "; Database=" + Database + "; Uid=" + User + "; Pwd=" + Pass + "; CharSet=utf8mb4"

'connStr = "Server=" + Address + "; Database=" + Database + "; Uid=" + User + "; Pwd=root; CharSet=utf8mb4"

conn = New MySqlConnection(connStr)

ModLog.Guardar(connStr)

If Not conn.Ping Then

conn.Open()

End If

Catch ex As Exception

MessageBox.Show(ex.ToString())

End Try

End Sub

Public Sub desconectar()

conn.Close()

End Sub

Public Function RConexion() As MySqlConnection

Return conn

End Function

#End Region

#Region "Interpretar"

Public Function ESQL(ByVal sql As String) As Boolean

Dim conT = New MySqlConnection(connStr)

conT.Open()

ModLog.Guardar(sql)

Try

objCmd = New MySqlCommand(sql, conn)

objCmd.Prepare()

objCmd.ExecuteNonQuery()

Catch ex As Exception

MessageBox.Show(ex.ToString)

Return False

End Try

conT.Close()

Return True

End Function

Public Function ESQLSelect(ByVal sql As String) As DataTable

Dim dt As New DataTable

Dim sqladapter As MySqlDataAdapter

Dim conT = New MySqlConnection(connStr)

conT.OpenAsync()

Try

objCmd = New MySqlCommand(sql, conT)

objCmd.Prepare()

sqladapter = New MySqlDataAdapter(objCmd)

sqladapter.FillAsync(dt)

Catch ex As Exception

MessageBox.Show(ex.ToString)

Return Nothing

End Try

conT.CloseAsync()

Return dt

End Function

Public Function ESQLSelect(ByVal objCmd As MySqlCommand, ByVal guardar As Boolean) As DataTable

dt = New DataTable

Dim sqladapter As MySqlDataAdapter

Try

sqladapter = New MySqlDataAdapter(objCmd)

sqladapter.Fill(dt)

If guardar Then

If ds.Tables.Contains(dt.TableName) Then

ds.Tables.Remove(dt.TableName)

End If

ds.Tables.Add(dt)

End If

Catch ex As Exception

MessageBox.Show(ex.ToString)

Return Nothing

End Try

sqladapter.Dispose()

Return dt

End Function

#End Region

#Region "Comandos"

Public Sub BSQL(ByVal nTabla As String, ByVal Condition As String)

ESQL("DELETE FROM " + nTabla + " WHERE " + Condition)

ModLog.Guardar("DELETE FROM " + nTabla + " WHERE " + Condition)

End Sub

Public Sub ISQL(ByVal nTabla As String, ByVal Column As String, ByVal Data As String, Optional parentesis As Boolean = True)

If (parentesis) Then

ESQL("Insert IGNORE into " + nTabla + " ( " + Column + " ) values (" + Data + " )")

Else

ESQL("Insert IGNORE into " + nTabla + " ( " + Column + " ) values " + Data)

End If

End Sub

Public Sub MISQL(ByVal nTabla As String, ByVal Column As String, ByVal datos() As String, Optional comilla As Boolean = True)

Dim r As String = ""

For Each d In datos

r += String.Format("{0}),(", d)

Next

If Not r.Length < 3 Then

ISQL(nTabla, Column, r.Remove(r.Length - 3), comilla)

End If

End Sub

Public Function SSQL(ByVal nColumn As String, ByVal nTabla As String, ByVal Condition As String) As DataTable

If ds.Tables.Contains(nTabla) Then

dt = New DataTable

dt.Columns.Add(ds.Tables(nTabla).Columns(nColumn))

Return dt

Else

Return ESQLSelect("select " + nColumn + " FROM " + nTabla + " WHERE " + Condition)

End If

End Function

Public Function PSQL(ByVal nColumn As String, ByVal nTabla As String, ByVal Condition As String) As String

Return "select " + nColumn + " FROM " + nTabla + " WHERE " + Condition

End Function

Public Sub USQL(ByVal nTabla As String, ByVal Orden As String, ByVal Condition As String)

ESQL("update " + nTabla + " set " + Orden + " WHERE " + Condition)

End Sub

#End Region

#Region "Usuarios"

Public Function AUsuarios(ByVal actualizar As Boolean) As DataTable

If ds.Tables.Contains("usuarios") And Not actualizar Then

dt = New DataTable()

dt.Columns.Add(ds.Tables("usuarios").Columns("Nombre Usuarios"))

Return dt

Else

Return DevolverTabla(PSQL("id\_usuario as 'ID', nombre as 'Nombre Usuarios'", "usuarios", "True"))

End If

End Function

Public Function BUsuario(ByVal nombre As String, ByVal contraseña As String) As Boolean

Try

objCmd = New MySqlCommand(PSQL("id\_usuario", "usuarios as User", "nombre = @nombre AND contrasena = AES\_ENCRYPT(@contrasena,sha2(@key,256))"), conn)

objCmd.Parameters.Add("@nombre", MySqlDbType.VarChar).Value = nombre

objCmd.Parameters.Add("@contrasena", MySqlDbType.VarChar).Value = contraseña

objCmd.Parameters.Add("@key", MySqlDbType.VarChar).Value = ModCodificador.GKey

objCmd.Prepare()

Dim dt As DataTable = ESQLSelect(objCmd, False)

If Not IsNothing(dt) Then

If dt.Rows.Count = 0 Then

MessageBox.Show("Contraseña o Usuario Incorrecto")

Else

Usuario = nombre

Password = contraseña

UsuarioID = Integer.Parse(dt.Rows(0)("id\_usuario"))

ModPermisos.Esid(UsuarioID)

CargarPermiso()

Return True

End If

End If

Catch e As Exception

MessageBox.Show(e.ToString)

End Try

Return False

End Function

Public Sub IUsuario(ByVal nombre As String, ByVal contraseña As String)

ISQL("usuarios", "nombre , contrasena", "'" + nombre + "', AES\_ENCRYPT('" + contraseña + "',sha2('" + ModCodificador.GKey + "',256))")

End Sub

#End Region

#Region "Main"

Public Function DevolverTabla(ByVal Datos As String) As DataTable

Try

Dim dt As DataTable = ESQLSelect(Datos)

If Not IsNothing(dt) Then

If Not dt.Rows.Count = 0 Then

Return dt

End If

End If

Catch e As Exception

MessageBox.Show(e.ToString)

End Try

Return Nothing

End Function

#Region "Actualizar"

Public Function APrograma(fecha As Date) As DataTable

Return DevolverTabla(PSQL("p.id\_programa, time\_format(hora\_inicio, '%H:%i') as 'Inicio', time\_format(hora\_fin, '%H:%i') as 'Final', Nombre\_programa as 'Programa'", "fechaprograma f inner join programa p on f.id\_programa=p.id\_programa", "fecha = '" + Format(fecha, "yyyy-MM-dd") + "'"))

End Function

Public Function AFPrograma(Fecha As Date, idPrograma As Integer) As DataTable

Dim Columna As String = "fun.id\_funcionario, fun.Nombre, Telefono, Mail, f.Nombre as Función"

Dim Tablas As String = "(select \* from funtrabaja where id\_Programa = {0}) ft inner join trabajacomo tc on ft.id\_trabajacomo = tc.id\_trabajacomo inner join funcion f on f.id\_funcion = tc.id\_funcion inner join funcionario fun on fun.id\_funcionario = tc.id\_funcionario"

Tablas = String.Format(Tablas, idPrograma)

Dim Condicion As String = String.Format("fecha\_inicio<='{0}' and ifnull(fecha\_finalizacion,'{0}')>='{0}'", Format(Fecha, "yyyy-MM-dd"))

Return DevolverTabla(PSQL(Columna, Tablas, Condicion))

End Function

Public Function APublicidad(Fecha As Date, Hora As String) As DataTable

ModLog.Guardar("llegamos3")

ModLog.Guardar(PSQL("p.id\_publicidad, Nombre", "publicidad p inner join aparecepubli a on p.id\_publicidad=a.id\_publicidad", "a.fecha\_inicio <= '" + Format(Fecha, "yyyy-MM-dd") + "' and a.fecha\_finalizacion >= '" + Format(Fecha, "yyyy-MM-dd") + "' and a.hora\_inicio = '" + Hora + "'"))

Return DevolverTabla(PSQL("p.id\_publicidad, Nombre", "publicidad p inner join aparecepubli a on p.id\_publicidad=a.id\_publicidad", "a.fecha\_inicio <= '" + Format(Fecha, "yyyy-MM-dd") + "' and a.fecha\_finalizacion >= '" + Format(Fecha, "yyyy-MM-dd") + "' and a.hora\_inicio = '" + Hora + "'"))

End Function

Public Function APPublicidad(Fecha As Date, idPrograma As Integer) As DataTable

Return DevolverTabla(PSQL("pp.id\_publicidad, Nombre", "pmuestrapubli pp inner join publicidad ppp on pp.id\_publicidad = ppp.id\_publicidad", "pp.fecha\_inicio <= '" + Format(Fecha, "yyyy-MM-dd") + "' and pp.fecha\_finalizacion >= '" + Format(Fecha, "yyyy-MM-dd").ToString + "' and pp.id\_programa = '" + idPrograma.ToString + "'"))

ModLog.Guardar(PSQL("pp.id\_publicidad, Nombre", "pmuestrapubli pp inner join publicidad ppp on pp.id\_publicidad = ppp.id\_publicidad", "pp.fecha\_inicio <= '" + Format(Fecha, "yyyy-MM-dd") + "' and pp.fecha\_finalizacion >= '" + Format(Fecha, "yyyy-MM-dd").ToString + "' and pp.id\_programa = '" + idPrograma.ToString + "'"))

End Function

Public Function AEventos() As DataTable

Return DevolverTabla(PSQL("e.id\_Evento, DATE\_FORMAT(Fecha,'%d/%m/%Y') as Fecha, Nombre", "evento e inner join fechaevento f on f.id\_evento=e.id\_evento", "f.fecha >= now()"))

End Function

Public Function ATandas(Optional siguiente As Boolean = True) As DataTable

Dim condicion As String = "true"

If (siguiente) Then

condicion = "(hora\_inicio <= curtime() and hora\_fin >= curtime()) or hora\_inicio >= curtime()"

End If

Return DevolverTabla(PSQL("hora\_inicio as 'Inicio', hora\_fin as 'Final'", "tanda", condicion))

End Function

Public Function ADPrograma(idPrograma As Integer) As String

Dim dt As DataTable = DevolverTabla(PSQL("Descripcion", "programa", "id\_programa = '" + idPrograma.ToString + "'"))

Return dt.Rows(0)(0).ToString

End Function

#End Region

#End Region

End Module

ModInicializador.vb

' El modulo trabaja como union y acceso entre los formularios

Module ModInicializador

Public Cancelar As String = ""

Public frmPrin As frmPrincipal

Public Sub Configuracion()

Dim frmConfig As New frmConfiguracion

' Muestra el formulario de configuracion

frmConfig.ShowDialog()

End Sub

Public Sub Generadores()

Dim Gen As New Generador\_DB

' Muestra el formulario de configuracion

Gen.ShowDialog()

End Sub

Public Sub Principal()

frmPrin = New frmPrincipal

' Muestra el formulario de configuracion

frmPrin.Show()

End Sub

End Module

ModLog.vb

Imports System.IO

Module ModLog

':::Ruta donde crearemos nuestro archivo txt

Private ruta As String = "..\User\"

':::Nombre del archivo

Private archivo As String = "Log.txt"

Private ENUSU As Boolean = False

Public Sub Guardar(ByVal Acciones As String)

If Not ENUSU Then

ENUSU = True

Dim escribir As New StreamWriter(ruta & archivo, True)

escribir.WriteLine(Now.ToString + " " + My.Computer.Name + " ' " + Acciones + " ")

escribir.Close()

'ENUSU = False

End If

End Sub

End Module

ModPermisos.vb

Module ModPermisos

Public Permisos As DataTable = Nothing

Public PermisosT As DataTable = Nothing

Public TieneP As Boolean()

Public ActualizaP As Boolean()

Private EliminarP As String = ""

Private AgregarP As String = ""

Private USid As Integer = Nothing

#Region "usuario"

Public Sub Esid(ByVal EUsid As Integer)

USid = EUsid

End Sub

#End Region

#Region "Cargadores"

Public Sub CargaActualizacionP(ByVal che As CheckedListBox)

ReDim ActualizaP(TieneP.Length - 1)

For i As Integer = 0 To TieneP.Length - 1

ActualizaP(i) = False

Next

For i As Integer = 0 To che.CheckedItems.Count - 1

ActualizaP(che.CheckedIndices(i)) = True

Next

End Sub

Public Sub CompararPermisos(ByRef che As CheckedListBox)

For i As Integer = 0 To TieneP.Length - 1

If (TieneP(i) <> ActualizaP(i)) Then

If (TieneP(i) = False) Then

AgregarP += String.Format("('{0}','{1}'),", USid, PermisosT.Rows(i).Item(0).ToString)

Else

EliminarP += String.Format("'{0}',", PermisosT.Rows(i).Item(0).ToString)

End If

End If

Next

If (AgregarP <> "") Then

AgregarP = AgregarP.Remove(AgregarP.Length - 1)

ISQL("ustieneacceso", "id\_usuario, id\_acceso", AgregarP, False)

End If

If (EliminarP <> "") Then

EliminarP = EliminarP.Remove(EliminarP.Length - 1)

Dim con As String = String.Format("id\_acceso in ({0}) and id\_usuario = '{1}'", EliminarP, USid)

BSQL("ustieneacceso", con)

End If

CargarPermisosAll(USid)

CheckearPermisos()

EstablecerList(che)

AgregarP = ""

EliminarP = ""

End Sub

Public Sub CargarPermiso(ByVal UID As Integer)

Permisos = DevolverTabla(PSQL("a.\*", "acceso a inner join ustieneacceso u on a.id\_acceso = u.id\_acceso", "id\_usuario = " + UID.ToString + " order by id\_acceso"))

End Sub

Public Sub CargarPermiso()

Permisos = DevolverTabla(PSQL("a.\*", "acceso a inner join ustieneacceso u on a.id\_acceso = u.id\_acceso", "id\_usuario = " + USid.ToString + " order by id\_acceso"))

End Sub

Public Sub CargarPermisoT()

PermisosT = DevolverTabla(PSQL("\*", "acceso", "true order by id\_acceso"))

ReDim TieneP(PermisosT.Rows.Count - 1)

End Sub

Public Sub EstablecerAll(ByRef che As CheckedListBox)

If (TieneP.Length > 0) Then

Dim sec As Boolean

sec = Not che.CheckedIndices().Contains(0)

For i As Integer = 0 To TieneP.Length - 1

TieneP(i) = sec

Next

EstablecerList(che)

CheckearPermisos()

End If

End Sub

Public Sub CargarPermisosAll(ByVal UID As Integer)

USid = UID

CargarPermisoT()

CargarPermiso()

End Sub

Public Sub CheckearPermisos()

For i As Integer = 0 To TieneP.Length - 1

TieneP(i) = False

Next

If (Not IsNothing(Permisos)) Then

Dim e As Integer = 0

For a As Integer = 0 To PermisosT.Rows.Count - 1

For i As Integer = e To Permisos.Rows.Count - 1

If (PermisosT.Rows(a).Item(0) = Permisos.Rows(i).Item(0)) Then

TieneP(a) = True

e = i + 1

Exit For

End If

Next

Next

End If

End Sub

Public Sub EstablecerList(ByRef che As CheckedListBox)

che.Items.Clear()

If (Not IsNothing(PermisosT)) Then

For i As Integer = 0 To TieneP.Length - 1

che.Items.Add(String.Format("Sector: {0}. Acceso: {1}", PermisosT.Rows(i).Item(1), PermisosT.Rows(i).Item(2)), TieneP(i))

Next

End If

End Sub

#End Region

Public Function PoseePermiso(ByVal Seccion As String, Optional tipo As String = "") As Boolean

If Not IsNothing(Permisos) Then

For i As Integer = 0 To Permisos.Rows.Count - 1

If (Permisos.Rows(i).Item(1) = Seccion) Then

If (tipo <> "") Then

If (Permisos.Rows(i).Item(2) = tipo) Then

Return True

End If

Else

Return True

End If

End If

Next

End If

Return True 'FIXME: Cambiar a False

End Function

End Module

ModTablas.vb

Imports System.Text.RegularExpressions

Module ModTablas

Public dt\_programa As DataTable

Public dt\_BPrograma As DataTable

Public dt\_dprograma As DataTable

Public dt\_evento As DataTable

Public dt\_tandas As DataTable

Public dt\_tandasCon As DataTable

Public dt\_publi As DataTable

Public dt\_Ppubli As DataTable

Public dt\_BPubli As DataTable

Public dt\_Video As DataTable

Public dt\_Serie As DataTable

Public dt\_Empresa As DataTable

Public dt\_BFuncionario As DataTable

Public dt\_BFuncion As DataTable

Public dt\_BEvento As DataTable

Public Const VIDEO As Byte = 1

Public Const SERIE As Byte = 2

Public Const EMPRESA As Byte = 3

Public Const PROGRAMAS As Byte = 4

Public Const PUBLICIDAD As Byte = 5

Public Const FUNCIONARIO As Byte = 6

Public Const FECHAPROGRAMA As Byte = 7

Public Const PUBLICIDADPROGRAMA As Byte = 8

Public Const TANDASHORAS As Byte = 9

Public Const FUNTRABAJA As Byte = 10

Public Const CUOTA As Byte = 11

Public Const FUNCION As Byte = 12

Public Const CUOTAPUBLICIDAD As Byte = 13

Public Const EVENTO As Byte = 14

Public Const TRABAJACOMO As Byte = 15

Public Const PUBLICIDADEVENTO As Byte = 16

Public Function ValidarEmail(ByVal s As String) As Boolean

Return Regex.IsMatch(s, "^([0-9a-zA-Z]([-\.\w]\*[0-9a-zA-Z])\*@([0-9a-zA-Z][-\w]\*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$")

End Function

Public Sub CargarPubliT(ByRef dt As DataTable, ByRef dgv As DataGridView, ByVal id As String, ByVal hora As String, ByVal fecha1 As String, ByVal fecha2 As String)

dt = DevolverTabla(PSQL("fecha\_inicio as 'Fecha Inicio', fecha\_finalizacion as 'Fecha Finalizacion'", "aparecepubli", String.Format("id\_publicidad = {0} and hora\_inicio='{1}' and fecha\_inicio <='{3}' and fecha\_finalizacion>='{2}'", id, hora, fecha1, fecha2)))

ActualizarTablaC(dt, dgv, False)

End Sub

Public Sub CargarPubliP(ByRef dt As DataTable, ByRef dgvP As DataGridView, ByVal id As String, ByVal programaid As String, ByVal fecha1 As String, ByVal fecha2 As String)

dt = DevolverTabla(PSQL("fecha\_inicio as 'Fecha Inicio', fecha\_finalizacion as 'Fecha Finalizacion'", "pmuestrapubli", String.Format("id\_publicidad = {0} and id\_programa='{1}' and fecha\_inicio <='{3}' and fecha\_finalizacion>='{2}'", id, programaid, fecha1, fecha2)))

ActualizarTablaC(dt, dgvP, False)

End Sub

Public Sub CargarPubliE(ByRef dt As DataTable, ByRef dgvP As DataGridView, ByVal id As String, ByVal eventoid As String, ByVal fecha1 As String, ByVal fecha2 As String)

dt = DevolverTabla(PSQL("fecha\_inicio as 'Fecha Inicio', fecha\_finalizacion as 'Fecha Finalizacion'", "eventomuestrapubli", String.Format("id\_publicidad = {0} and id\_evento='{1}' and fecha\_inicio <='{3}' and fecha\_finalizacion>='{2}'", id, eventoid, fecha1, fecha2)))

ActualizarTablaC(dt, dgvP, False)

End Sub

Public Sub PubliDeFecha(ByRef dt As DataTable, ByRef dgvP As DataGridView, ByVal programaid As String, ByVal fecha1 As Date)

dt = DevolverTabla(PSQL("p.id\_publicidad, Nombre, fecha\_inicio as 'Fecha Inicio', fecha\_finalizacion as 'Fecha Finalizacion'", "pmuestrapubli pm inner join publicidad p on pm.id\_publicidad = p.id\_publicidad", String.Format("id\_programa='{0}' and fecha\_inicio <='{1}' and fecha\_finalizacion>='{1}'", programaid, Format(fecha1, "yyyy-MM-dd"))))

ModLog.Guardar(PSQL("p.id\_publicidad, Nombre, fecha\_inicio as 'Fecha Inicio', fecha\_finalizacion as 'Fecha Finalizacion'", "pmuestrapubli pm inner join publicidad p on pm.id\_publicidad = p.id\_publicidad", String.Format("id\_programa='{0}' and fecha\_inicio <='{1}' and fecha\_finalizacion>='{1}'", programaid, Format(fecha1, "yyyy-MM-dd"))))

ActualizarTablaC(dt, dgvP)

End Sub

Public Sub PubliDeFechaE(ByRef dt As DataTable, ByRef dgvP As DataGridView, ByVal id As String, ByVal fecha1 As Date)

dt = DevolverTabla(PSQL("p.id\_publicidad, Nombre, fecha\_inicio as 'Fecha Inicio', fecha\_finalizacion as 'Fecha Finalizacion'", "eventomuestrapubli pm inner join publicidad p on pm.id\_publicidad = p.id\_publicidad", String.Format("id\_evento='{0}' and year(fecha\_inicio) <= year('{1}') and year(fecha\_finalizacion) >= year('{1}')", id, Format(fecha1, "yyyy-MM-dd"))))

ActualizarTablaC(dt, dgvP)

End Sub

Public Function CargarID(ByRef Tabla As DataTable, ByRef Dgv As DataGridView) As Integer

If (Not IsNothing(Tabla)) And Dgv.SelectedRows.Count > 0 Then

Return Tabla.Rows(Dgv.SelectedRows(0).Index).Item(0).ToString

End If

Return -1

End Function

Public Function ceros(ByVal h As String) As String

Dim hor As String = If(h.Length = 1, "0" + h, h)

Return hor

End Function

Public Function MysqlHM(ByVal hora As DateTime) As String

Dim h As String = String.Format("{0}:{1}:{2}", ceros(Hour(hora).ToString), ceros(Minute(hora)), ceros(Second(hora)))

Return h

End Function

Public Function Dia(ByVal dias As Date) As String

Dim h As String = String.Format("{0}/{1}/{2}", ceros((dias).Day), ceros(Month(dias)), ceros(Year(dias)))

Return h

End Function

Public Function Mili(ByVal hora As DateTime, Optional aumento As Byte = 0) As Integer

Dim tiempo As Integer = 0

tiempo += hora.Second \* 1000

tiempo += hora.Minute + aumento \* 60000

tiempo += hora.Hour \* 360000

Return tiempo

End Function

Public Sub HORAIF(ByRef dat As DateTimePicker)

dat.MaxDate = Now.Date.AddDays(1)

dat.MinDate = Now.Date

End Sub

Public Sub NumerTime(ByVal dt As DataTable, ByRef time As Timer, ByVal numcol As Byte)

Dim tiempo As Integer

If Not IsNothing(dt) Then

If (dt.Rows.Count > 0) Then

For i As Integer = 0 To dt.Rows.Count - 1

tiempo = Mili(Convert.ToDateTime(Dia(Now.Date) + " " + MysqlHM(dt.Rows(i).Item(numcol).ToString)), 1) - Mili(Now)

If tiempo <= 0 Then

time.Stop()

time.Enabled = False

Continue For

End If

time.Enabled = True

time.Interval = tiempo

time.Start()

Exit Sub

Next

End If

time.Stop()

time.Enabled = False

Else

time.Stop()

time.Enabled = False

End If

End Sub

Public Function CargarID(ByRef Tabla As DataTable, ByRef Dgv As DataGridView, ByVal NumCol() As Byte) As String()

If (Not IsNothing(Tabla)) And Dgv.SelectedRows.Count > 0 Then

Dim res(NumCol.Length - 1) As String

Dim j As Byte = 0

For Each i As Byte In NumCol

res(j) = Tabla.Rows(Dgv.SelectedRows(0).Index).Item(i).ToString

j += 1

Next

Return res

End If

Dim err(0) As String

Return err

End Function

Public Function CargarTodo(ByRef Tabla As DataTable, ByVal NumCol As Byte) As String()

If (Not IsNothing(Tabla)) Then

Dim res(Tabla.Rows.Count) As String

For i As Byte = 0 To Tabla.Rows.Count - 1

res(i) = Tabla.Rows(i).Item(NumCol).ToString

Next

Return res

End If

Dim err(0) As String

Return err

End Function

Public Sub ClickCheck(ByRef Dgv As DataGridView, ByVal columna As Integer)

If Dgv.Rows.Count > 0 Then

If columna = Dgv.Columns.Count - 1 And Dgv.SelectedRows.Count > 0 Then

Dgv.SelectedRows(0).Cells(Dgv.Columns.Count - 1).Value = Not Dgv.SelectedRows(0).Cells(Dgv.Columns.Count - 1).Value

End If

End If

End Sub

Public Sub CheckAll(ByRef Dgv As DataGridView, ByVal columna As Integer)

Dgv.ClearSelection()

If Dgv.Rows.Count > 0 Then

If columna = Dgv.Columns.Count - 1 Then

Dgv.Rows(0).Cells(Dgv.Columns.Count - 1).Value = Not Dgv.Rows(0).Cells(Dgv.Columns.Count - 1).Value

For Each row As DataGridViewRow In Dgv.Rows

row.Cells(Dgv.Columns.Count - 1).Value = Dgv.Rows(0).Cells(Dgv.Columns.Count - 1).Value

Next

End If

End If

End Sub

Public Function ObtenerCheck(ByRef Tabla As DataTable, ByRef Dgv As DataGridView, Optional ByVal Col As Integer = 0, Optional ByVal text As String = "", Optional comilla As Boolean = False) As String()

Dim UltiCol As Integer = Dgv.Columns.Count - 1

Dim Ids(Dgv.Rows.Count - 1) As String

Dim Valores As Integer = 0

For i As Integer = 0 To Dgv.Rows.Count - 1

If (Dgv.Rows(i).Cells(UltiCol).Value = True) Then

Ids(i) = String.Format(If(comilla, "'{0}'", "{0}"), Tabla.Rows(i).Item(Col).ToString) + text

Valores += 1

Else

Ids(i) = ""

End If

Next

Dim ID(Valores - 1) As String

Dim iter As Integer = 0

For Each i As String In Ids

If (i <> "") Then

ID(iter) = i

iter += 1

End If

Next

Return ID

End Function

Public Sub ActualizarTablaC(ByRef Tabla As DataTable, ByRef Dgv As DataGridView, Optional C As Boolean = True, Optional ByVal col() As Byte = Nothing)

If (IsNothing(col)) Then

col = {0}

Else

Array.Sort(col)

Array.Reverse(col)

End If

If Not IsNothing(Tabla) Then

'MessageBox.Show("Carga")

Dim Tamanos(Dgv.Columns.Count() - 1) As Single

Dim AutoSizeMode(Dgv.Columns.Count() - 1) As DataGridViewAutoSizeColumnMode

For i As Integer = 0 To Dgv.Columns.Count - 1

Tamanos(i) = Dgv.Columns(i).FillWeight()

AutoSizeMode(i) = Dgv.Columns(i).AutoSizeMode

Next

Dim Columna As DataGridViewColumn

Columna = Dgv.Columns(Tamanos.Length - 1)

Columna.ReadOnly = False

Dgv.Columns.Clear()

Dgv.DataSource = Tabla

If (Tabla.Columns.Count > 1 And C And Dgv.Columns.Count > col.Length) Then

For Each k As Byte In col

Dgv.Columns.RemoveAt(k)

Next

End If

If ((Tabla.Columns.Count - col.Length + 1 = Tamanos.Length And C) Or (Not C And Tabla.Columns.Count = Tamanos.Length - 1)) Then

Dgv.Columns.Add(Columna)

End If

For i As Integer = 0 To Dgv.Columns.Count - 1

Dgv.Columns(i).FillWeight() = Tamanos(i)

Dgv.Columns(i).AutoSizeMode() = AutoSizeMode(i)

Dgv.Columns(i).SortMode = DataGridViewColumnSortMode.NotSortable

Dgv.Columns(i).ReadOnly = If(Tabla.Columns.Count = Tamanos.Length, False, True)

Next

Dgv.Refresh()

'MessageBox.Show("Refresca")

Else

If (Dgv.Rows.Count > 0) Then

Dgv.DataSource.Rows.Clear()

Dgv.Refresh()

End If

End If

End Sub

Public Sub PrepararUpdate(ByVal tabla As String, ByVal datos() As String, ByVal id As String)

Dim res As String = ""

Dim dt As DataTable = ESQLSelect("describe " + tabla)

Dim i As Integer = 1

For Each dato In datos

res += String.Format(If(dato = "null", "{0} = {1},", "{0} = '{1}',"), dt.Rows(i).Item(0).ToString, dato)

i += 1

Next

res = res.Remove(res.Length - 1)

USQL(tabla, res, String.Format("{0} = '{1}'", dt.Rows(0).Item(0).ToString, id))

End Sub

Public Sub PrepararInsert(ByVal tabla As String, ByVal datos() As String, Optional Inicio As Integer = 1)

Dim col As String = ""

Dim valu As String = ""

Dim dt As DataTable = ESQLSelect("describe " + tabla)

Dim i As Integer = Inicio

For Each dato In datos

col += String.Format("{0},", dt.Rows(i).Item(0).ToString)

valu += String.Format(If(dato = "null" Or dato.Contains("("), "{0},", "'{0}',"), dato)

i += 1

Next

col = col.Remove(col.Length - 1)

valu = valu.Remove(valu.Length - 1)

ISQL(tabla, col, valu)

End Sub

Public Sub PrepararUpdate(ByVal tabla As String, ByVal Columna() As String, ByVal datos() As String, ByVal Condiciones() As String, ByVal id() As String)

Dim res As String = ""

Dim i As Integer = 0

For Each dato In datos

res += String.Format(If(dato = "null", "{0} = {1},", "{0} = '{1}',"), Columna(i), dato)

i += 1

Next

i = 0

res = res.Remove(res.Length - 1)

Dim condicion As String = ""

For Each con In Condiciones

condicion += String.Format(If(id(i) = "null", "{0} = {1} and ", "{0} = '{1}' and "), con, id(i))

i += 1

Next

condicion = condicion.Remove(condicion.Length - 5)

USQL(tabla, res, condicion)

End Sub

Public Sub PrepararUpdate(ByVal tabla As String, ByVal Columna() As String, ByVal datos() As String, ByVal Condicion As String, ByVal IDs() As String)

Dim res As String = ""

Dim i As Integer = 0

For Each dato In datos

res += String.Format(If(dato = "null", "{0} = {1},", "{0} = '{1}',"), Columna(i), dato)

i += 1

Next

res = res.Remove(res.Length - 1)

Dim con As String = ""

For Each id In IDs

con += String.Format(If(id = "null", "{0} = {1} or ", "{0} = '{1}' or "), Condicion, id)

Next

con = con.Remove(con.Length - 4)

USQL(tabla, res, con)

End Sub

Public Sub PrepararDelete(ByVal tabla As String, ByVal datos() As String, ByVal ids() As String)

Dim res As String = ""

Dim i As Integer = 0

For Each dato In datos

res += String.Format(If(ids(i) = "null", "{0} = {1} and", "{0} = '{1}' and"), datos(i), ids(i))

i += 1

Next

res = res.Remove(res.Length - 3)

BSQL(tabla, res)

End Sub

Public Sub PrepararDelete(ByVal tabla As String, ByVal datos As String, ByVal ids() As String)

Dim res As String = datos + " in ("

For Each id In ids

res += String.Format(If(id = "null", "{0} ,", "'{0}' ,"), id)

Next

res = res.Remove(res.Length - 1)

res += ")"

BSQL(tabla, res)

End Sub

Public Function CreadorCondicion(ByVal datos As String, ByVal ids() As String, Optional fecha As Boolean = False)

Dim res As String = datos + " in ("

For Each id In ids

res += String.Format(If(id = "null", "{0} ,", "'{0}' ,"), If(fecha, Format(CDate(id), "yyyy-MM-dd"), id))

Next

res = res.Remove(res.Length - 1)

res += ")"

Return res

End Function

Public Function BuscarDatos(ByVal tabla As String, ByVal Columnas() As String, ByVal campo As String, ByVal id As String) As String()

Dim res As String = ""

Dim resultado(Columnas.Length - 1) As String

For Each columna In Columnas

res += String.Format("{0}, ", columna)

Next

res = res.Remove(res.Length - 2)

Dim dtN As DataTable = DevolverTabla(PSQL(res, tabla, String.Format("{0} = '{1}'", campo, id)))

ModLog.Guardar(PSQL(res, tabla, String.Format("{0} = '{1}'", campo, id)))

If (Not IsNothing(dtN)) Then

For j As Integer = 0 To dtN.Columns.Count - 1

resultado(j) = dtN.Rows(0).Item(j).ToString

Next

End If

Return resultado

End Function

Public Sub DevolverIds(ByRef lis As CheckedListBox, ByVal dt As DataTable)

For i As Integer = 0 To dt.Rows.Count - 1

lis.Items.Add(dt.Rows(i).Item(1).ToString + dt.Rows(i).Item(1).ToString)

Next

End Sub

Public Sub LlenarCombo(ByRef con As ComboBox, ByVal dt As DataTable, ByVal col As String)

con.Items.Clear()

con.Items.Add("No está relacionado")

If (Not IsNothing(dt)) Then

If (dt.Rows.Count > 0) Then

For j As Integer = 0 To dt.Rows.Count - 1

con.Items.Add(dt.Rows(j).Item(col).ToString)

Next

End If

End If

End Sub

Public Function CompararValores(ByVal s1() As String, ByVal s2() As String) As Boolean

For j As Integer = 0 To s2.Length - 1

If s1(j) <> s2(j) Then

Return False

End If

Next

Return True

End Function

Public Function VaciarNull(ByVal s1() As String) As String()

Dim sN(s1.Length - 1) As String

For j As Integer = 0 To s1.Length - 1

sN(j) = s1(j)

If s1(j) = "null" Then

sN(j) = ""

End If

Next

Return sN

End Function

Public Sub BorrarConfirmar(ByRef form As Form, ByRef dt As DataTable, ByRef dgv As DataGridView, ByVal tabla As Byte, ByRef btn As Button)

If Not IsNothing(dt) Then

If (dt.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt, dgv)

If Not Id.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(tabla, Id, False)

formDelete.ShowDialog(form)

btn.PerformClick()

End If

End If

End If

End Sub

End Module

ModUser.vb

Imports System

Imports System.IO

Imports System.Collections

Module ModUser

':::Ruta donde crearemos nuestro archivo txt

Private ruta As String = "..\User\"

':::Nombre del archivo

Private archivo As String = "User.txt"

Private Barchivo As String = "Origin.txt"

Private karchivo As String = "Key.txt"

'Establece la key y una verificacion

Public Sub Inicio()

If Not Directory.Exists(ruta) Then

Directory.CreateDirectory(ruta)

End If

If File.Exists(ruta & karchivo) Then

Dim leer As New StreamReader(ruta & karchivo)

ModCodificador.EKey(leer.ReadLine())

ModCodificador.EKeyMaestra(leer.ReadLine())

leer.Close()

ModCodificador.Actualizar()

LeeDatos()

Verify()

Else

If File.Exists(ruta & archivo) Then

File.Delete(ruta & archivo)

End If

If File.Exists(ruta & Barchivo) Then

File.Delete(ruta & Barchivo)

End If

Dim escribir As New StreamWriter(ruta & karchivo, False)

Dim r As New Random

escribir.WriteLine(RandomString(r, 8))

escribir.WriteLine(RandomString(r, 32))

escribir.Close()

Inicio()

End If

End Sub

Function RandomString(r As Random, ByVal cnt As Byte)

Dim s As String = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"

Dim sb As New Text.StringBuilder

For i As Integer = 1 To cnt

Dim idx As Integer = r.Next(0, s.Length)

sb.Append(s.Substring(idx, 1))

Next

Return sb.ToString()

End Function

Public Sub Verify()

If Not File.Exists(ruta & archivo) Then

If Not File.Exists(ruta & Barchivo) Then

File.Create(ruta & Barchivo).Close()

Else

File.Copy(ruta & Barchivo, ruta & archivo)

End If

End If

End Sub

Public Sub Borrar()

If File.Exists(ruta & archivo) Then

File.Delete(ruta & archivo)

End If

End Sub

Public Sub LeeDatos()

If File.Exists(ruta & archivo) Then

Dim leer As New StreamReader(ruta & archivo)

Dim Datos(4) As String

Dim n As Byte = 0

While leer.Peek <> -1

'Leemos cada linea del archivo TXT

Dim linea As String = leer.ReadLine()

'Agregramos los datos

Datos(n) = ModCodificador.Desencriptar(linea)

n += 1

End While

ModConector.Crear(Datos(0), Datos(1), Datos(2), Datos(3), Datos(4))

leer.Close()

End If

End Sub

Public Sub Guardar(ByVal Desarrollo As Boolean)

Dim archivos As String

If Desarrollo Then

archivos = Barchivo

Else

archivos = archivo

End If

Dim escribir As New StreamWriter(ruta & Archivos, False)

escribir.WriteLine(ModCodificador.Encriptar(ModConector.GAddress()))

escribir.WriteLine(ModCodificador.Encriptar(ModConector.GPort()))

escribir.WriteLine(ModCodificador.Encriptar(ModConector.GDatabase()))

escribir.WriteLine(ModCodificador.Encriptar(ModConector.GUser()))

escribir.WriteLine(ModCodificador.Encriptar(ModConector.GPass()))

escribir.Close()

End Sub

End Module

Formularios:

frmEmpresa.vb

' TODO: Indicador de editar (?)

Public Class frmEmpresa

Dim empresaID As Integer

Dim editando As Boolean = False ' Controla si se esta en modo de edicion o no

Dim tmpDatos(3) As String

Dim cambio As Boolean = False ' Controla si han habido cambios desde el ultimo modo de edicion

Dim dt\_Publicidad As New DataTable

Dim datos() As String

Public Sub New(ByVal DatosI() As String)

InitializeComponent()

'Los siguientes datos se obtienen de la tabla en el elemento padre

empresaID = DatosI(0)

txtNombre.Text = DatosI(1)

txtTelefono.Text = DatosI(2)

txtMail.Text = DatosI(3)

''ModLog.Guardar(PSQL("id\_video, fecha as Fecha, nombre as Nombre", "video", String.Format("id\_serie = '{0}'", DatosI(0))))

End Sub

Private Sub frmEmpresa\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

ActualizarTabla()

If (empresaID = -1) Then

Alternar()

End If

If Not PoseePermiso("Empresa", "a") Then

btnBorrar.Visible = False

btnSEditar.Visible = False

End If

End Sub

Private Sub CargarDatos()

datos = {txtNombre.Text, txtTelefono.Text, If(ValidarEmail(txtMail.Text), txtMail.Text, tmpDatos(3))}

End Sub

Private Sub btnSEditar\_Click(sender As Object, e As EventArgs) Handles btnSEditar.Click

'' editando = True -> Se guardaran los cambios

'' editando = False -> Se le permitira al usuario escribir en los campos

If empresaID = -1 Then

CargarDatos()

PrepararInsert("Empresa", datos)

Vaciar()

ElseIf editando Then

If cambio Then

CargarDatos()

If Not CompararValores(VaciarNull(datos), tmpDatos) Then

PrepararUpdate("Empresa", datos, empresaID)

End If

AlternarCambioHandlers()

End If

Else

tmpDatos(0) = txtNombre.Text

tmpDatos(1) = txtTelefono.Text

tmpDatos(2) = txtMail.Text

End If

Alternar()

End Sub

Private Sub Vaciar()

txtNombre.Clear()

txtTelefono.Clear()

txtMail.Clear()

End Sub

Private Sub btnSSalir\_Click(sender As Object, e As EventArgs) Handles btnSSalir.Click

If Not editando Or empresaID = -1 Then

Close()

Else

If cambio Then

txtNombre.Text = tmpDatos(0)

txtTelefono.Text = tmpDatos(1)

txtMail.Text = tmpDatos(2)

AlternarCambioHandlers()

End If

Alternar()

End If

End Sub

Private Sub Serie\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

If cambio And empresaID <> -1 Then

CargarDatos()

If Not CompararValores(VaciarNull(datos), tmpDatos) Then

Dim g As New frmGuardarEdicion("Empresa", datos, empresaID)

g.ShowDialog()

If ModInicializador.Cancelar.Contains("Empresa") Then

e.Cancel = True

ModInicializador.Cancelar = ModInicializador.Cancelar.Replace("Empresa", "")

Else

AlternarCambioHandlers()

End If

End If

End If

End Sub

'' Alternar botones

Private Sub Alternar()

If empresaID = -1 Then

btnSEditar.Text = "Ingresar"

btnBorrar.Visible = False

btnSSalir.Text = "Salir"

Text = "Ingresar Empresa"

ElseIf editando Then

btnSEditar.Text = "Editar"

btnSSalir.Text = "Salir"

Text = "Ver Empresa"

Else

btnSSalir.Text = "Cancelar"

btnSEditar.Text = "Guardar"

Text = "Editar Empresa"

End If

editando = Not editando

txtNombre.ReadOnly = Not editando

txtTelefono.ReadOnly = Not editando

txtMail.ReadOnly = Not editando

End Sub

'' Checkean si hay cambios hechos

'' Si cambio = True, no se llamaran

Private Sub txt\_ModifiedChanged(sender As Object, e As EventArgs) Handles txtNombre.ModifiedChanged, txtMail.ModifiedChanged, txtTelefono.ModifiedChanged

If txtNombre.Modified Or txtMail.Modified Or txtTelefono.Modified Then

AlternarCambioHandlers()

End If

End Sub

' Aqui yacIA mi variable, recordatorio de lo que una vez fue Y AUN ES

' cambio: I LIVED BITCH

'' Alterna el estado de cambios y activa/desactiva la deteccion de cambios

Private Sub AlternarCambioHandlers()

'' cambio = True -> Se aniadiran los handlers, se los necesita

'' cambio = False -> Se ha detectado un cambio, y los handlers se removeran hasta la siguiente llamada del metodo

If cambio Then

AddHandler txtNombre.ModifiedChanged, AddressOf txt\_ModifiedChanged

AddHandler txtMail.ModifiedChanged, AddressOf txt\_ModifiedChanged

AddHandler txtTelefono.ModifiedChanged, AddressOf txt\_ModifiedChanged

Else

RemoveHandler txtNombre.ModifiedChanged, AddressOf txt\_ModifiedChanged

RemoveHandler txtMail.ModifiedChanged, AddressOf txt\_ModifiedChanged

RemoveHandler txtTelefono.ModifiedChanged, AddressOf txt\_ModifiedChanged

End If

cambio = Not cambio

End Sub

Private Sub dgvVSM\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvVSM.CellDoubleClick

Dim i As Integer = CargarID(dt\_Video, dgvVSM)

If (i <> -1) And PoseePermiso("Publicidades") Then

Dim formPubli As New frmPublicidad(i)

AddHandler formPubli.FormClosed, AddressOf formPubli\_FormClosed

formPubli.ShowDialog()

End If

End Sub

Private Sub formPubli\_FormClosed(sender As Object, e As FormClosedEventArgs)

ActualizarTabla()

End Sub

'' Actualiza la tabla mostrando los videos asociados cuando el formulario de mostrar video se ha cerrado

Private Sub FormPublicidad\_FormClosed(sender As Object, e As FormClosedEventArgs)

ActualizarTabla()

' TODO: Darle uso.

End Sub

Private Sub ActualizarTabla()

dt\_Publicidad = DevolverTabla(PSQL("ID\_Publicidad, Nombre, Tema", "publicidad", String.Format("ID\_Empresa = '{0}'", empresaID)))

ActualizarTablaC(dt\_Publicidad, dgvVSM)

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

Dim formDelete As New frmConfirmarBorrado(EMPRESA, {empresaID}, True)

formDelete.ShowDialog(Me)

End Sub

End Class

frmEventos.vb

Imports System.ComponentModel

Public Class frmEvento

Private eventoID As Integer

Private editando As Boolean = False

Private datos() As String

Private datosI() As String

Private dtE As DataTable

Private dt\_fechas As DataTable

Private position() As String

Private pos As UInt16 = 0

Private videoID As String = ""

Private dt\_publicidades As DataTable

Public Sub New(ByVal id As Integer)

InitializeComponent()

eventoID = id

End Sub

Public Sub Buscar()

Dim columnas() As String = {"Nombre", "Descripcion", "id\_video"}

datosI = BuscarDatos("evento", columnas, "id\_evento", eventoID)

Rellenar()

End Sub

Private Sub frmEvento\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

If Not PoseePermiso("Evento", "a") Then

btnBorrar.Visible = False

btnEditar.Visible = False

tcP.TabPages.RemoveByKey("tbFechas")

tcP.TabPages.RemoveByKey("tbPublicidad")

End If

If Not PoseePermiso("Publicidad", "a") Then

tcP.TabPages.RemoveByKey("tbPublicidad")

End If

If eventoID <> -1 Then

Buscar()

btnSalir.Select()

Else

tcP.TabPages.RemoveByKey("tbFechas")

tcP.TabPages.RemoveByKey("tbPublicidad")

btnEditar.Text = "Insertar"

btnBorrar.Visible = False

Activar()

CargarCombo()

End If

btnSalir.Select()

End Sub

Private Sub Rellenar()

txtNombre.Text = datosI(0)

txtDescripcion.Text = datosI(1)

If (datosI(2) = "") Then

videoID = 0

Else

videoID = datosI(2)

If (datosI(2) = 0) Then

datosI(2) = ""

End If

End If

CargarCombo()

End Sub

Sub CargarCombo()

dtE = DevolverTabla(PSQL("id\_video, Nombre", "video", "True"))

LlenarCombo(cbVideo, dtE, "Nombre")

If Not IsNothing(dtE) Then

ExtraerDatos()

End If

cbVideo.SelectedIndex = pos

End Sub

Public Sub ExtraerDatos()

ReDim position(dtE.Rows.Count - 1)

For j As Integer = 0 To dtE.Rows.Count - 1

position(j) = dtE.Rows(j).Item(0).ToString

Next

For i As Integer = 0 To position.Length - 1

If videoID = position(i) Then

pos = i + 1

Exit For

End If

Next

End Sub

Private Sub btnSalir\_Click(sender As Object, e As EventArgs) Handles btnSalir.Click

If Not editando Or videoID = -1 Then

Close()

Else

Rellenar()

Alternar()

End If

End Sub

Private Sub ActualizarDatos()

Dim des As String = txtDescripcion.Text

Dim nom As String = txtNombre.Text

Dim vid As String = If(cbVideo.SelectedIndex <= 0, "null", position(cbVideo.SelectedIndex - 1))

datos = {nom, des, vid}

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

Dim formDelete As New frmConfirmarBorrado(EVENTO, {eventoID}, True)

formDelete.ShowDialog(Me)

End Sub

Sub Activar()

txtNombre.ReadOnly = editando

txtDescripcion.ReadOnly = editando

editando = Not editando

cbVideo.Enabled = editando

End Sub

Private Sub Alternar()

Activar()

btnEditar.Text = If(editando, "Guardar", "Editar")

btnSalir.Text = If(editando, "Cancelar", "Salir")

End Sub

Sub Vaciar()

txtNombre.Clear()

txtDescripcion.Clear()

cbVideo.SelectedIndex = -1

End Sub

Private Sub btnEditar\_Click(sender As Object, e As EventArgs) Handles btnEditar.Click

If eventoID = -1 Then

ActualizarDatos()

PrepararInsert("evento", datos)

Vaciar()

ElseIf editando Then

ActualizarDatos()

If Not CompararValores(VaciarNull(datos), datosI) Then

PrepararUpdate("evento", datos, eventoID)

datosI = VaciarNull(datos)

End If

Alternar()

Else

Alternar()

End If

End Sub

Private Sub frmEvento\_Closing(sender As Object, e As CancelEventArgs) Handles Me.Closing

If editando And eventoID <> -1 Then

ActualizarDatos()

If Not CompararValores(VaciarNull(datos), datosI) Then

Dim g As New frmGuardarEdicion("Evento", datos, videoID)

g.ShowDialog()

If ModInicializador.Cancelar.Contains("Evento") Then

e.Cancel = True

ModInicializador.Cancelar = ModInicializador.Cancelar.Replace("Evento", "")

' ModInicializador.frmPrin.btnbuscarv.PerformClick()

End If

End If

End If

End Sub

Private Sub btnAnadir\_Click(sender As Object, e As EventArgs) Handles btnAnadir.Click

Dim datosN() As String

For i As Byte = mcFecha.SelectionStart.Day To mcFecha.SelectionEnd.Day

datosN = {eventoID, String.Format(Format(mcFecha.SelectionStart, "yyyy-MM") + "-{0}", i)}

PrepararInsert("fechaevento", datosN, 0)

Next

bwFechas.RunWorkerAsync()

End Sub

Private Sub btnABorrar\_Click(sender As Object, e As EventArgs) Handles btnABorrar.Click

If Not IsNothing(dt\_fechas) Then

If (dt\_fechas.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_fechas, dgvFechas, 0)

If Not Id.Length = 0 Then

BSQL("fechaevento", String.Format("id\_evento='{0}'", eventoID) + " and " + CreadorCondicion("fecha", Id, True))

If Not (bwFechas.IsBusy) Then

bwFechas.RunWorkerAsync()

End If

End If

End If

End If

End Sub

Private Sub dgvPrograma\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFechas.CellClick, dgvEventoPubli.CellClick

ClickCheck(sender, e.ColumnIndex)

End Sub

Private Sub dgvHeaderClick(sender As Object, e As DataGridViewCellMouseEventArgs) Handles dgvFechas.ColumnHeaderMouseClick, dgvEventoPubli.ColumnHeaderMouseClick

CheckAll(sender, e.ColumnIndex)

End Sub

Private Sub tcP\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles tcP.SelectedIndexChanged

If (tcP.SelectedIndex = 1) Then

bwFechas.RunWorkerAsync()

ElseIf (tcP.SelectedIndex = 2) Then

PubliDeFechaE(dt\_publicidades, dgvEventoPubli, eventoID, dtpFPubli.Value)

End If

End Sub

Private Sub bwFechas\_DoWork(sender As Object, e As DoWorkEventArgs) Handles bwFechas.DoWork

Dim Columna As String = "Fecha"

Dim fecha As String = "year(fecha) = year('{1}')"

Dim Condicion As String = String.Format("id\_evento = {0} and " + fecha, eventoID, Format(dtpYearE.Value, "yyyy-MM-dd"))

Dim Tablas As String = "fechaevento"

dt\_fechas = DevolverTabla(PSQL(Columna, Tablas, Condicion))

End Sub

Private Sub bwFechas\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles bwFechas.RunWorkerCompleted

ActualizarTablaC(dt\_fechas, dgvFechas, False)

End Sub

Private Sub btnBorrarSelect\_Click(sender As Object, e As EventArgs) Handles btnBorrarSelect.Click

If Not IsNothing(dt\_publicidades) Then

If (dt\_publicidades.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_publicidades, dgvEventoPubli, 0)

Dim Id1() As String = ObtenerCheck(dt\_publicidades, dgvEventoPubli, 2)

If Not Id.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(PUBLICIDADEVENTO, Id, False, {eventoID}, Id1)

formDelete.ShowDialog(Me)

PubliDeFechaE(dt\_publicidades, dgvEventoPubli, eventoID, dtpFPubli.Value)

End If

End If

End If

End Sub

Private Sub dtpFPubli\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFPubli.ValueChanged

PubliDeFechaE(dt\_publicidades, dgvEventoPubli, eventoID, dtpFPubli.Value)

End Sub

End Class

frmConfirmarBorrado.vb

Public Class frmConfirmarBorrado

Dim tabla As Byte

Dim id() As String

Dim id2() As String

Dim id3() As String

Dim id4() As String

Dim c As Boolean = False

Public Sub New(ByVal t As Byte, ByVal identificador() As String, ByVal Cerrar As Boolean, Optional ByVal Eid() As String = Nothing, Optional ByVal Fid() As String = Nothing, Optional ByVal Gid() As String = Nothing)

InitializeComponent()

tabla = t

id = identificador

id2 = Eid

id3 = Fid

id4 = Gid

c = Cerrar

End Sub

Private Sub btnCancelar\_Click(sender As Object, e As EventArgs) Handles btnCancelar.Click

Close()

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

Select Case tabla

Case SERIE

PrepararDelete("video", "id\_serie", id)

USQL("evento", "id\_video=null", String.Format("id\_video='{0}'", id))

PrepararDelete("Serie", "id\_serie", id)

Case PUBLICIDAD

PrepararDelete("publicidad", "id\_publicidad", id)

PrepararDelete("eventomuestrapubli", "id\_publicidad", id)

PrepararDelete("publicidadcuota", "id\_publicidad", id)

PrepararDelete("aparecepubli", "id\_publicidad", id)

PrepararDelete("pmustrapubli", "id\_publicidad", id)

Case VIDEO

PrepararDelete("Video", "id\_video", id)

USQL("evento", "id\_video=null", String.Format("id\_video='{0}'", id))

Case FUNTRABAJA

BSQL("funtrabaja", String.Format("id\_programa='{0}'", id(0)) + " and " + CreadorCondicion("ID\_TrabajaComo", id2) + " and " + CreadorCondicion("fecha\_inicio", id3, True))

Case EMPRESA

PrepararDelete("Empresa", "id\_empresa", id)

Case PROGRAMAS

PrepararDelete("Programa", "ID\_Programa", id)

Case FUNCIONARIO

PrepararDelete("Funcionario", "ID\_Funcionario", id)

Dim Borrado As DataTable = DevolverTabla(PSQL("id\_trabajacomo", "trabajacomo", String.Format("ID\_Funcionario='{0}'", id)))

Dim idTrabaja() As String = CargarTodo(Borrado, 0)

PrepararDelete("trabajacomo", "ID\_Funcionario", id)

BSQL("funtrabaja", CreadorCondicion("id\_trabajacomo", idTrabaja))

Case FECHAPROGRAMA

BSQL("Fechaprograma", CreadorCondicion("fecha", id, True) + " and " + CreadorCondicion("Hora\_inicio", id2) + "and id\_programa='" + id3(0) + "'")

Case PUBLICIDADPROGRAMA

BSQL("pmuestrapubli", CreadorCondicion("id\_publicidad", id) + " and " + CreadorCondicion("id\_programa", id2) + " and " + CreadorCondicion("Fecha\_inicio", id3, True))

Case PUBLICIDADEVENTO

BSQL("eventomuestrapubli", CreadorCondicion("id\_publicidad", id) + " and " + CreadorCondicion("id\_evento", id2) + " and " + CreadorCondicion("Fecha\_inicio", id3, True))

Case TANDASHORAS

BSQL("tanda", CreadorCondicion("Hora\_Inicio", id))

Case CUOTA

BSQL("programacuota", CreadorCondicion("id\_programa\_cuota", id))

Case FUNCION

PrepararDelete("Funcion", "ID\_Funcion", id)

Case CUOTAPUBLICIDAD

BSQL("publicidadcuota", CreadorCondicion("id\_publicidadcuota", id))

Case EVENTO

BSQL("evento", CreadorCondicion("id\_evento", id))

BSQL("fechaevento", CreadorCondicion("id\_evento", id))

BSQL("eventomuestrapubli", CreadorCondicion("id\_evento", id))

End Select

If (c) Then

Owner.Close()

End If

Close()

End Sub

Private Sub BtnBorrarTodo\_Click(sender As Object, e As EventArgs) Handles btnBorrarSerie.Click

PrepararUpdate("video", {"id\_serie"}, {"null"}, {"id\_serie"}, id)

PrepararDelete("Serie", "id\_serie", id)

If (Not IsNothing(dt\_Video)) Then

frmPrin.btnlimpiarv.PerformClick()

End If

If (c) Then

Owner.Close()

End If

Close()

End Sub

Private Sub frmConfirmarBorrado\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

If (tabla = SERIE) Then

btnBorrarSerie.Visible = True

End If

End Sub

End Class

frmFuncion.vb

Public Class frmFuncion

Dim ID As Integer

Dim editando As Boolean = False ' Controla si se esta en modo de edicion o no

Dim tmpDatos(1) As String

Dim cambio As Boolean = False ' Controla si han habido cambios desde el ultimo modo de edicion

Dim datos() As String

Public Sub New(ByVal DatosI() As String)

InitializeComponent()

'Los siguientes datos se obtienen de la tabla en el elemento padre

ID = DatosI(0)

txtNombre.Text = DatosI(1)

txtDesc.Text = DatosI(2)

End Sub

Private Sub frmFuncion\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

If Not PoseePermiso("Funcionario", "a") Then

btnBorrar.Visible = False

btnEditar.Visible = False

End If

If (ID = -1) Then

Alternar()

End If

btnSalir.Select()

End Sub

Private Sub CargarDatos()

datos = {txtNombre.Text, txtDesc.Text}

End Sub

Private Sub btnEditar\_Click(sender As Object, e As EventArgs) Handles btnEditar.Click

'' editando = True -> Se guardaran los cambios

'' editando = False -> Se le permitira al usuario escribir en los campos

If ID = -1 Then

CargarDatos()

PrepararInsert("Funcion", datos)

Vaciar()

ElseIf editando Then

If cambio Then

CargarDatos()

If Not CompararValores(VaciarNull(datos), tmpDatos) Then

PrepararUpdate("Funcion", datos, ID)

End If

AlternarCambioHandlers()

End If

Else

tmpDatos = {txtNombre.Text, txtDesc.Text}

End If

Alternar()

End Sub

Private Sub Vaciar()

txtNombre.Clear()

txtDesc.Clear()

End Sub

Private Sub btnSalir\_Click(sender As Object, e As EventArgs) Handles btnSalir.Click

If Not editando Or ID = -1 Then

Close()

Else

If cambio Then

txtNombre.Text = tmpDatos(0)

txtDesc.Text = tmpDatos(1)

AlternarCambioHandlers()

End If

Alternar()

End If

End Sub

Private Sub FrmFuncion\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

If cambio And ID <> -1 Then

CargarDatos()

If Not CompararValores(VaciarNull(datos), tmpDatos) Then

Dim g As New frmGuardarEdicion("Funcion", datos, ID)

g.ShowDialog()

If ModInicializador.Cancelar.Contains("Funcion") Then

e.Cancel = True

ModInicializador.Cancelar = ModInicializador.Cancelar.Replace("Funcion", "")

Else

AlternarCambioHandlers()

End If

End If

End If

End Sub

'' Alternar botones

Private Sub Alternar()

If ID = -1 Then

btnEditar.Text = "Ingresar"

btnBorrar.Visible = False

btnSalir.Text = "Salir"

Text = "Ingresar Funcion"

ElseIf editando Then

btnEditar.Text = "Editar"

btnSalir.Text = "Salir"

Text = "Ver Funcion"

Else

btnSalir.Text = "Cancelar"

btnEditar.Text = "Guardar"

Text = "Editar Funcion"

End If

editando = Not editando

txtNombre.ReadOnly = Not editando

txtDesc.ReadOnly = Not editando

End Sub

'' Checkean si hay cambios hechos

'' Si cambio = True, no se llamaran

Private Sub txt\_ModifiedChanged(sender As Object, e As EventArgs) Handles txtNombre.ModifiedChanged, txtDesc.ModifiedChanged

If txtNombre.Modified Or txtDesc.Modified Then

AlternarCambioHandlers()

End If

End Sub

' Aqui yacIA mi variable, recordatorio de lo que una vez fue Y AUN ES

' cambio: I LIVED BITCH

'' Alterna el estado de cambios y activa/desactiva la deteccion de cambios

Private Sub AlternarCambioHandlers()

'' cambio = True -> Se aniadiran los handlers, se los necesita

'' cambio = False -> Se ha detectado un cambio, y los handlers se removeran hasta la siguiente llamada del metodo

If cambio Then

AddHandler txtNombre.ModifiedChanged, AddressOf txt\_ModifiedChanged

AddHandler txtDesc.ModifiedChanged, AddressOf txt\_ModifiedChanged

Else

RemoveHandler txtNombre.ModifiedChanged, AddressOf txt\_ModifiedChanged

RemoveHandler txtDesc.ModifiedChanged, AddressOf txt\_ModifiedChanged

End If

cambio = Not cambio

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

Dim formDelete As New frmConfirmarBorrado(FUNCION, {ID}, True)

formDelete.ShowDialog(Me)

End Sub

End Class

frmFuncionario.vb

Imports System.ComponentModel

Public Class frmFuncionario

Dim ID As Integer

Dim editando As Boolean = False ' Controla si se esta en modo de edicion o no

Dim tmpDatos(3) As String

Dim cambio As Boolean = False ' Controla si han habido cambios desde el ultimo modo de edicion

Dim dt\_Funciones As New DataTable

Dim datos() As String

Dim TBuscada As Byte

Dim TBusca As DataTable

Dim dt\_BFuncionesAs As DataTable

Private dt\_ProgramasP As DataTable

Dim positionPrograma() As String

Dim pos() As UInt16 = {0}

Public Sub New(ByVal DatosI() As String)

InitializeComponent()

'Los siguientes datos se obtienen de la tabla en el elemento padre

ID = DatosI(0)

txtNombre.Text = DatosI(1)

txtTelefono.Text = DatosI(2)

txtMail.Text = DatosI(3)

End Sub

Private Sub frmFuncionario\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

If Not PoseePermiso("Funcionario", "a") Then

btnBorrar.Visible = False

btnEditar.Visible = False

tcF.TabPages.RemoveByKey("tbAF")

tcF.TabPages.RemoveByKey("TBAP")

End If

If Not PoseePermiso("Programa", "a") Then

tcF.TabPages.RemoveByKey("TBAP")

End If

ActualizarTabla()

If (ID = -1) Then

tcF.TabPages.RemoveByKey("tbAF")

tcF.TabPages.RemoveByKey("TBAP")

Alternar()

End If

btnSalir.Select()

End Sub

Private Sub CargarDatos()

datos = {txtTelefono.Text, txtNombre.Text, If(ValidarEmail(txtMail.Text), txtMail.Text, tmpDatos(3))}

End Sub

Private Sub btnEditar\_Click(sender As Object, e As EventArgs) Handles btnEditar.Click

'' editando = True -> Se guardaran los cambios

'' editando = False -> Se le permitira al usuario escribir en los campos

If ID = -1 Then

CargarDatos()

PrepararInsert("Funcionario", datos)

Vaciar()

ElseIf editando Then

If cambio Then

CargarDatos()

If Not CompararValores(VaciarNull(datos), tmpDatos) Then

PrepararUpdate("Funcionario", datos, ID)

End If

AlternarCambioHandlers()

End If

Else

tmpDatos = {txtTelefono.Text, txtNombre.Text, txtMail.Text}

End If

Alternar()

End Sub

Private Sub Vaciar()

txtNombre.Clear()

txtTelefono.Clear()

txtMail.Clear()

End Sub

Private Sub btnSalir\_Click(sender As Object, e As EventArgs) Handles btnSalir.Click

If Not editando Or ID = -1 Then

Close()

Else

If cambio Then

txtNombre.Text = tmpDatos(1)

txtTelefono.Text = tmpDatos(0)

txtMail.Text = tmpDatos(2)

AlternarCambioHandlers()

End If

Alternar()

End If

End Sub

Private Sub FrmFuncionario\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

If cambio And ID <> -1 Then

CargarDatos()

If Not CompararValores(VaciarNull(datos), tmpDatos) Then

Dim g As New frmGuardarEdicion("Funcionario", datos, ID)

g.ShowDialog()

If ModInicializador.Cancelar.Contains("Funcionario") Then

e.Cancel = True

ModInicializador.Cancelar = ModInicializador.Cancelar.Replace("Funcionario", "")

Else

AlternarCambioHandlers()

End If

End If

End If

End Sub

'' Alternar botones

Private Sub Alternar()

If ID = -1 Then

btnEditar.Text = "Ingresar"

btnBorrar.Visible = False

btnSalir.Text = "Salir"

Text = "Ingresar Funcionario"

ElseIf editando Then

btnEditar.Text = "Editar"

btnSalir.Text = "Salir"

Text = "Ver Funcionario"

Else

btnSalir.Text = "Cancelar"

btnEditar.Text = "Guardar"

Text = "Editar Funcionario"

End If

editando = Not editando

txtNombre.ReadOnly = Not editando

txtTelefono.ReadOnly = Not editando

txtMail.ReadOnly = Not editando

End Sub

'' Checkean si hay cambios hechos

'' Si cambio = True, no se llamaran

Private Sub txt\_ModifiedChanged(sender As Object, e As EventArgs) Handles txtNombre.ModifiedChanged, txtMail.ModifiedChanged, txtTelefono.ModifiedChanged

If txtNombre.Modified Or txtMail.Modified Or txtTelefono.Modified Then

AlternarCambioHandlers()

End If

End Sub

' Aqui yacIA mi variable, recordatorio de lo que una vez fue Y AUN ES

' cambio: I LIVED BITCH

'' Alterna el estado de cambios y activa/desactiva la deteccion de cambios

Private Sub AlternarCambioHandlers()

'' cambio = True -> Se aniadiran los handlers, se los necesita

'' cambio = False -> Se ha detectado un cambio, y los handlers se removeran hasta la siguiente llamada del metodo

If cambio Then

AddHandler txtNombre.ModifiedChanged, AddressOf txt\_ModifiedChanged

AddHandler txtMail.ModifiedChanged, AddressOf txt\_ModifiedChanged

AddHandler txtTelefono.ModifiedChanged, AddressOf txt\_ModifiedChanged

Else

RemoveHandler txtNombre.ModifiedChanged, AddressOf txt\_ModifiedChanged

RemoveHandler txtMail.ModifiedChanged, AddressOf txt\_ModifiedChanged

RemoveHandler txtTelefono.ModifiedChanged, AddressOf txt\_ModifiedChanged

End If

cambio = Not cambio

End Sub

Private Sub dgvFunciones\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFunciones.CellDoubleClick

Dim i() As String = CargarID(dt\_Funciones, dgvFunciones, {0, 1, 2})

If (i(0) > 0) Then

Dim formFuncion As New frmFuncion(i)

AddHandler formFuncion.FormClosed, AddressOf FormFuncion\_FormClosed

formFuncion.ShowDialog()

End If

End Sub

'' Actualiza la tabla mostrando las funciones asociadas cuando el formulario de mostrar funciones se ha cerrado

Private Sub FormFuncion\_FormClosed(sender As Object, e As FormClosedEventArgs)

ActualizarTabla()

End Sub

'' Actualizar tabla de Funciones

Private Sub ActualizarTabla()

dt\_Funciones = DevolverTabla(PSQL("f.ID\_Funcion, f.Nombre, f.Descripcion", "Funcion f join TrabajaComo t on f.ID\_Funcion = t.ID\_Funcion", String.Format("t.ID\_Funcionario = '{0}'", ID)))

ActualizarTablaC(dt\_Funciones, dgvFunciones)

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

Dim formDelete As New frmConfirmarBorrado(FUNCIONARIO, {ID}, True)

formDelete.ShowDialog(Me)

End Sub

Private Sub btnBuscar\_Click(sender As Object, e As EventArgs) Handles btnBuscar.Click

Buscar()

End Sub

Private Sub BWBuscador\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles BWBuscador.RunWorkerCompleted

Select Case TBuscada

Case FUNCION

dt\_BFuncionesAs = TBusca

ActualizarTablaC(dt\_BFuncionesAs, dgvFuncionesBFF)

TBuscada = FUNCIONARIO

BWBuscador.RunWorkerAsync(PSQL("f.ID\_Funcion, t.id\_trabajacomo , f.Nombre, f.Descripcion", "Funcion f join TrabajaComo t on f.ID\_Funcion = t.ID\_Funcion", String.Format("t.ID\_Funcionario = '{0}'", ID)))

Case FUNCIONARIO

dt\_Funciones = TBusca

ActualizarTablaC(dt\_Funciones, dgvFuncionesAs, True, {0, 1})

TBusca = Nothing

TBuscada = 0

Case PROGRAMAS

dt\_ProgramasP = TBusca

CargarComboPrograma()

TBusca = Nothing

TBuscada = 0

Case FUNTRABAJA

dt\_Funciones = TBusca

ActualizarTablaC(dt\_Funciones, dgvFunP, True, {0, 1})

TBusca = Nothing

TBuscada = 0

End Select

End Sub

Private Sub BWBuscador\_DoWork(sender As Object, e As DoWorkEventArgs) Handles BWBuscador.DoWork

TBusca = DevolverTabla(e.Argument)

ModLog.Guardar("Funcionario: " & e.Argument)

End Sub

Private Sub tcF\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles tcF.SelectedIndexChanged

If tcF.SelectedIndex = 0 Then

ActualizarTabla()

ElseIf tcF.SelectedIndex = 1 Then

Buscar()

Else

BuscarFun()

End If

End Sub

Private Sub BuscarFun()

TBuscada = FUNTRABAJA

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("f.ID\_Funcion, t.id\_trabajacomo, f.Nombre, f.Descripcion", "Funcion f join TrabajaComo t on f.ID\_Funcion = t.ID\_Funcion", String.Format("t.ID\_Funcionario = '{0}'", ID)))

End If

End Sub

Private Sub Buscar()

TBuscada = FUNCION

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If (Not String.IsNullOrWhiteSpace(txtNombreBFF.Text)) Then

condicion = String.Format("Nombre like '%{0}%'", txtNombreBFF.Text)

End If

If (Not String.IsNullOrWhiteSpace(txtDescripcionBFF.Text)) Then

condicion += String.Format(" and Descripcion like '%{0}%'", txtDescripcionBFF.Text)

End If

condicion += String.Format(" and ID\_Funcion not in (select ID\_Funcion from TrabajaComo where ID\_Funcionario = {0})", ID)

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("ID\_Funcion, Nombre, Descripcion", "Funcion", condicion))

End If

End Sub

Private Sub btnAsignar\_Click(sender As Object, e As EventArgs) Handles btnAsignar.Click

If ObtenerCheck(dt\_BFuncionesAs, dgvFuncionesBFF).Length > 0 Then

MISQL("TrabajaComo",

"ID\_Funcion, ID\_Funcionario",

ObtenerCheck(dt\_BFuncionesAs,

dgvFuncionesBFF,

0,

String.Format(",'{0}'", ID.ToString),

True))

Buscar()

End If

End Sub

Private Sub dgv\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFuncionesBFF.CellClick, dgvFuncionesAs.CellClick, dgvFunP.CellClick

ClickCheck(sender, e.ColumnIndex)

End Sub

Private Sub dgvHeaderClick(sender As Object, e As DataGridViewCellMouseEventArgs) Handles dgvFuncionesBFF.ColumnHeaderMouseClick, dgvFuncionesAs.ColumnHeaderMouseClick, dgvFunP.ColumnHeaderMouseClick

CheckAll(sender, e.ColumnIndex)

End Sub

Private Sub btnDesasignar\_Click(sender As Object, e As EventArgs) Handles btnDesasignar.Click

Dim Checked() As String = ObtenerCheck(dt\_Funciones, dgvFuncionesAs)

If Checked.Length > 0 Then

BSQL("TrabajaComo", String.Format("ID\_Funcionario = '{0}' and ", ID) + CreadorCondicion("ID\_Funcion", Checked))

Buscar()

End If

End Sub

Private Sub btnPrograma\_Click(sender As Object, e As EventArgs) Handles btnPrograma.Click

BuscarPrograma()

End Sub

Private Sub BuscarPrograma()

TBuscada = PROGRAMAS

Dim condicion = "false"

If Not String.IsNullOrWhiteSpace(txtNombreP.Text) Then

condicion = String.Format("Nombre\_Programa like '%{0}%'", txtNombreP.Text)

End If

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("id\_programa, Nombre\_Programa", "programa", condicion))

End If

End Sub

Private Sub dtpFIP\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFIP.ValueChanged

If (dtpFFP.Value < dtpFIP.Value) Then

dtpFFP.Value = dtpFIP.Value

End If

dtpFFP.MinDate = dtpFIP.Value

End Sub

Sub CargarComboPrograma()

LlenarCombo(cbPrograma, dt\_ProgramasP, "Nombre\_Programa")

If Not IsNothing(dt\_ProgramasP) Then

ExtraerDatosProg()

Else

MessageBox.Show("No se encontró el programa")

End If

cbPrograma.SelectedIndex = pos(0)

End Sub

Public Sub ExtraerDatosProg()

ReDim positionPrograma(dt\_ProgramasP.Rows.Count - 1)

For j As Integer = 0 To dt\_ProgramasP.Rows.Count - 1

positionPrograma(j) = dt\_ProgramasP.Rows(j).Item(0).ToString

Next

End Sub

Private Sub btnAsignarProg\_Click(sender As Object, e As EventArgs) Handles btnAsignarProg.Click

If (cbPrograma.SelectedIndex > 0) Then

If ObtenerCheck(dt\_Funciones, dgvFunP).Length > 0 Then

MISQL("funtrabaja", "id\_trabajacomo, id\_programa, fecha\_inicio, fecha\_finalizacion", ObtenerCheck(

dt\_Funciones,

dgvFunP,

1, String.Format(",'{0}','{1}',", positionPrograma(cbPrograma.SelectedIndex - 1), Format(dtpFIP.Value, "yyyy-MM-dd")) + If(cbFF.Checked, String.Format("'{0}'", Format(dtpFFP.Value, "yyyy-MM-dd")), "null"), True))

BuscarFun()

dtpFFP.Value = Now

dtpFIP.Value = Now

cbFF.Checked = False

End If

Else

MessageBox.Show("Debe seleccionar un programa para asignar")

End If

End Sub

Private Sub btnMP\_Click(sender As Object, e As EventArgs) Handles btnMP.Click

If (cbPrograma.SelectedIndex > 0) Then

Dim formPrograma As New frmPrograma(positionPrograma(cbPrograma.SelectedIndex - 1))

AddHandler formPrograma.FormClosed, AddressOf FormPrograma\_FormClosed

formPrograma.ShowDialog()

End If

End Sub

Private Sub FormPrograma\_FormClosed()

BuscarPrograma()

End Sub

Private Sub dgvFunP\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFunP.CellDoubleClick

Dim i() As String = CargarID(dt\_Funciones, dgvFunP, {0, 2, 3})

If (i(0) > 0) Then

Dim formFuncion As New frmFuncion(i)

AddHandler formFuncion.FormClosed, AddressOf FormFuncion\_FormClosed

formFuncion.ShowDialog()

End If

End Sub

Private Sub dgvFuncionesBFF\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFuncionesBFF.CellDoubleClick

Dim i() As String = CargarID(dt\_BFuncionesAs, dgvFuncionesBFF, {0, 1, 2})

If (i(0) > 0) Then

Dim formFuncion As New frmFuncion(i)

AddHandler formFuncion.FormClosed, AddressOf Buscar

formFuncion.ShowDialog()

End If

End Sub

Private Sub dgvFuncionesAs\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFuncionesAs.CellDoubleClick

Dim i() As String = CargarID(dt\_Funciones, dgvFuncionesAs, {0, 2, 3})

If (i(0) > 0) Then

Dim formFuncion As New frmFuncion(i)

AddHandler formFuncion.FormClosed, AddressOf Buscar

formFuncion.ShowDialog()

End If

End Sub

End Class

frmPrograma.vb

Imports System.ComponentModel

Public Class frmPrograma

Dim programaID As Integer

Private editando As Boolean = False

Dim datos() As String

Dim datosI() As String

Private TBusca As DataTable

Private dt\_funcionario As DataTable

Private dt\_fechas As DataTable

Private dt\_FechaPrograma As DataTable

Private dt\_publicidades As DataTable

Private TBuscada As Byte

Private dt\_Cuotas As DataTable

Private CuotaId As Integer = -1

Public Sub New(ByVal pid As Integer)

InitializeComponent()

programaID = pid

End Sub

Public Sub Buscar()

Dim columnas() As String = {"Nombre\_Programa", "Descripcion", "DATE\_FORMAT(Fecha\_Finalizacion,'%Y-%m-%d') as Fecha\_Finalizacion"}

datosI = BuscarDatos("programa", columnas, "id\_programa", programaID)

End Sub

Private Sub Rellenar()

txtNombre.Text = datosI(0)

If datosI(2) <> "" Then

dtpFecha.Value = CDate(datosI(2))

Else

txtTapar.Visible = True

End If

txtDescripcion.Text = datosI(1)

End Sub

Private Sub FormPubli\_FormClosed(sender As Object, e As FormClosedEventArgs)

PubliDeFecha(dt\_publicidades, dgvProgramaPubli, programaID, dtpFPubli.Value)

End Sub

Private Sub frmPrograma\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

If programaID <> -1 Then

bwDatos.RunWorkerAsync()

HORAIF(txtHI)

HORAIF(txtHF)

If Not PoseePermiso("Programa", "a") Then

btnBorrar.Visible = False

btnPEditar.Visible = False

ocultar()

End If

Else

ocultar()

btnPEditar.Text = "Insertar"

btnBorrar.Visible = False

Activar()

End If

dtpFecha.BackColor = Color.FromArgb(64, 64, 64)

dtpFecha.ForeColor = Color.White

btnPSalir.Select()

End Sub

Private Sub ocultar()

tcP.TabPages.RemoveByKey("tbFuncionarios")

tcP.TabPages.RemoveByKey("tbPublicidades")

tcP.TabPages.RemoveByKey("tbAlquiler")

tcP.TabPages.RemoveByKey("tbFechas")

tcP.TabPages.RemoveByKey("tbFechasAgendadas")

End Sub

Private Sub bwDatos\_DoWork(sender As Object, e As DoWorkEventArgs) Handles bwDatos.DoWork

Buscar()

End Sub

Private Sub bwDatos\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles bwDatos.RunWorkerCompleted

Rellenar()

End Sub

Sub Activar()

txtNombre.ReadOnly = editando

txtDescripcion.ReadOnly = editando

editando = Not editando

dtpFecha.Enabled = editando

chbTieneFecha.Visible = editando

End Sub

Private Sub Alternar()

Activar()

btnPEditar.Text = If(editando, "Guardar", "Editar")

btnPSalir.Text = If(editando, "Cancelar", "Salir")

chbTieneFecha.Checked = datosI(2) <> ""

If editando Then

txtTapar.Visible = False

Else

If datosI(2) = "" Then

txtTapar.Visible = True

chbTieneFecha.Visible = False

End If

End If

End Sub

Sub Vaciar()

txtNombre.Text = ""

txtDescripcion.Text = ""

dtpFecha.Value = Now.Date

chbTieneFecha.Checked = False

End Sub

Private Sub ActualizarDatos()

Dim des As String = txtDescripcion.Text

Dim nom As String = txtNombre.Text

Dim dat As String = If(chbTieneFecha.Checked, Format(dtpFecha.Value, "yyyy-MM-dd"), "null")

datos = {nom, des, dat}

End Sub

Private Sub btnSEditar\_Click(sender As Object, e As EventArgs) Handles btnPEditar.Click

If programaID = -1 Then

ActualizarDatos()

PrepararInsert("Programa", datos)

vaciar()

ElseIf editando Then

ActualizarDatos()

If Not CompararValores(VaciarNull(datos), datosI) Then

PrepararUpdate("Programa", datos, programaID)

datosI = VaciarNull(datos)

End If

Alternar()

Else

Alternar()

End If

End Sub

Private Sub frmPrograma\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

If editando And programaID <> -1 Then

ActualizarDatos()

If Not CompararValores(VaciarNull(datos), datosI) Then

Dim g As New frmGuardarEdicion("Programa", datos, programaID)

g.ShowDialog()

If ModInicializador.Cancelar.Contains("Programa") Then

e.Cancel = True

ModInicializador.Cancelar = ModInicializador.Cancelar.Replace("Programa", "")

' ModInicializador.frmPrin.btnbuscarv.PerformClick()

End If

End If

End If

End Sub

Private Sub btnPSalir\_Click(sender As Object, e As EventArgs) Handles btnPSalir.Click

If Not editando Or programaID = -1 Then

Close()

Else

Rellenar()

Alternar()

End If

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

Dim formDelete As New frmConfirmarBorrado(PROGRAMAS, {programaID}, True)

formDelete.ShowDialog(Me)

End Sub

Private Sub bwCargador\_DoWork(sender As Object, e As DoWorkEventArgs) Handles bwCargador.DoWork

TBusca = DevolverTabla(e.Argument)

ModLog.Guardar(e.Argument)

End Sub

Private Sub tcP\_DoubleClick(sender As Object, e As EventArgs) Handles tcP.DoubleClick

If editando And programaID <> -1 Then

ActualizarDatos()

If Not CompararValores(VaciarNull(datos), datosI) Then

Dim g As New frmGuardarEdicion("Programa", datos, programaID)

g.ShowDialog()

If ModInicializador.Cancelar.Contains("Programa") Then

tcP.SelectedIndex = 0

ModInicializador.Cancelar = ModInicializador.Cancelar.Replace("Programa", "")

' ModInicializador.frmPrin.btnbuscarv.PerformClick()

End If

End If

End If

End Sub

Private Sub tcP\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles tcP.SelectedIndexChanged

TBuscada = 0

Select Case tcP.SelectedIndex

Case 1

BuscarFuncionario()

Case 2

PubliDeFecha(dt\_publicidades, dgvProgramaPubli, programaID, Now.Date)

Case 3

BuscarCuota()

Case 4

BFecha(cbFMes.Checked)

Case 5

BFechaRango(cbBMA.Checked)

End Select

End Sub

Public Sub BFecha(Optional ByVal mes As Boolean = False)

Dim Columna As String = "hora\_inicio as 'Inicio', hora\_fin as 'Final'"

Dim fecha As String = If(mes, "month(fecha) = month('{1}')", "fecha = '{1}'")

Dim Condicion As String = String.Format("id\_programa = {0} and " + fecha, programaID, Format(dtpBP.Value, "yyyy-MM-dd"))

Dim Tablas As String = "fechaprograma"

TBuscada = FECHAPROGRAMA

If Not (bwCargador.IsBusy) Then

bwCargador.RunWorkerAsync(PSQL(Columna, Tablas, Condicion))

End If

End Sub

Private Sub BuscarFuncionario()

Dim Condicion As String = String.Format("ifnull(fecha\_finalizacion,curdate()){0}curdate()", If(cbRP.Checked, "<", ">="))

TBuscada = FUNCIONARIO

Dim Columna As String = "fun.id\_funcionario, ft.ID\_TrabajaComo, fun.Nombre, f.Nombre as Función, fecha\_inicio as 'Inicio de la función', fecha\_finalizacion as 'Fin de la función', Telefono, Mail"

Dim Tablas As String = "(select \* from funtrabaja where id\_Programa = {0}) ft inner join trabajacomo tc on ft.id\_trabajacomo = tc.id\_trabajacomo inner join funcion f on f.id\_funcion = tc.id\_funcion inner join funcionario fun on fun.id\_funcionario = tc.id\_funcionario"

Tablas = String.Format(Tablas, programaID)

If Not (bwCargador.IsBusy) And TBuscada <> 0 Then

bwCargador.RunWorkerAsync(PSQL(Columna, Tablas, Condicion))

End If

End Sub

Private Sub BuscarCuota()

Dim Condicion As String = String.Format("id\_programa='{0}' and Fecha\_Pago is{1}null and year(fecha\_emision)={2}", programaID, If(cbPagados.Checked, " not ", " "), Year(dtpYearCuota.Value))

TBuscada = CUOTA

Dim Columna As String = "id\_programa\_cuota, fecha\_emision as 'Fecha de emisión', fecha\_pago as 'Fecha de pago', precio as 'Valor'"

Dim Tablas As String = "programacuota"

If Not (bwCargador.IsBusy) Then

bwCargador.RunWorkerAsync(PSQL(Columna, Tablas, Condicion))

End If

End Sub

Private Sub bwCargador\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles bwCargador.RunWorkerCompleted

Select Case TBuscada

Case FUNCIONARIO

dt\_funcionario = TBusca

ActualizarTablaC(dt\_funcionario, dgvFuncionarios, True, {1, 0, 6, 7})

Case FECHAPROGRAMA

dt\_fechas = TBusca

ActualizarTablaC(dt\_fechas, dgvPrograma, False)

Case PROGRAMAS

dt\_FechaPrograma = TBusca

ActualizarTablaC(dt\_FechaPrograma, dgvFechaPrograma, False)

Case CUOTA

dt\_Cuotas = TBusca

ActualizarTablaC(dt\_Cuotas, dgvVerCuota, True)

End Select

TBusca = Nothing

TBuscada = 0

End Sub

Private Sub btnAnadir\_Click(sender As Object, e As EventArgs) Handles btnAnadir.Click

Dim datos() As String = {Format(dtpAP.Value().Date, "yyyy-MM-dd"), MysqlHM(txtHI.Value), MysqlHM(txtHF.Value), programaID}

PrepararInsert("fechaprograma", datos, 0)

txtHI.Value = Now

txtHF.value = Now

BFecha(cbFMes.Checked)

End Sub

Private Sub dtpBP\_ValueChanged(sender As Object, e As EventArgs) Handles dtpBP.ValueChanged

BFecha(cbFMes.Checked)

End Sub

Private Sub btnABorrar\_Click(sender As Object, e As EventArgs) Handles btnABorrar.Click

If Not IsNothing(dt\_fechas) Then

If (dt\_fechas.Rows.Count > 0) Then

Dim RId() As String = ObtenerCheck(dt\_fechas, dgvPrograma, 0)

If Not RId.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(FECHAPROGRAMA, {dtpBP.Value}, False, RId, {programaID})

formDelete.ShowDialog(Me)

BFecha(cbFMes.Checked)

End If

End If

End If

End Sub

Private Sub dgv\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvProgramaPubli.CellClick, dgvPrograma.CellClick, dgvFechaPrograma.CellClick, dgvFuncionarios.CellClick, dgvVerCuota.CellClick

ClickCheck(sender, e.ColumnIndex)

End Sub

Private Sub dgvHeaderClick(sender As Object, e As DataGridViewCellMouseEventArgs) Handles dgvProgramaPubli.ColumnHeaderMouseClick, dgvPrograma.ColumnHeaderMouseClick, dgvFechaPrograma.ColumnHeaderMouseClick, dgvFuncionarios.ColumnHeaderMouseClick, dgvVerCuota.ColumnHeaderMouseClick

CheckAll(sender, e.ColumnIndex)

End Sub

Private Sub dtpFPubli\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFPubli.ValueChanged

PubliDeFecha(dt\_publicidades, dgvProgramaPubli, programaID, dtpFPubli.Value)

End Sub

Private Sub btnBorrarSelect\_Click(sender As Object, e As EventArgs) Handles btnBorrarSelect.Click

If Not IsNothing(dt\_publicidades) Then

If (dt\_publicidades.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_publicidades, dgvProgramaPubli, 0)

Dim Id1() As String = ObtenerCheck(dt\_publicidades, dgvProgramaPubli, 2)

If Not Id.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(PUBLICIDADPROGRAMA, Id, False, {programaID}, Id1)

formDelete.ShowDialog(Me)

PubliDeFecha(dt\_publicidades, dgvProgramaPubli, programaID, dtpFPubli.Value)

End If

End If

End If

End Sub

Private Sub dgvProgramaPubli\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvProgramaPubli.CellDoubleClick

Dim i As String = CargarID(dt\_publicidades, dgvProgramaPubli)

If (i <> 0) Then

Dim formPubli As New frmPublicidad(i)

AddHandler formPubli.FormClosed, AddressOf FormPubli\_FormClosed

formPubli.ShowDialog()

End If

End Sub

Private Sub btnBorrarAgenda\_Click(sender As Object, e As EventArgs) Handles btnBorrarAgenda.Click

If Not IsNothing(dt\_FechaPrograma) Then

If (dt\_FechaPrograma.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_FechaPrograma, dgvFechaPrograma, 0)

Dim Id2() As String = ObtenerCheck(dt\_FechaPrograma, dgvFechaPrograma, 1)

If Not Id.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(FECHAPROGRAMA, Id, False, Id2, {programaID})

formDelete.ShowDialog(Me)

BFechaRango(cbBMA.Checked)

End If

End If

End If

End Sub

Private Sub BFechaRango(ByVal mes As Boolean)

Dim Columna As String = "Fecha, hora\_inicio as 'Inicio', hora\_fin as 'Final'"

Dim fechasm As String = String.Format("fecha >= '{0}' and fecha <='{1}'", Format(mcFecha.SelectionStart, "yyyy-MM-dd"), Format(mcFecha.SelectionEnd, "yyyy-MM-dd"))

Dim fecham As String = String.Format("month(fecha) = month('{0}')", Format(mcFecha.SelectionStart, "yyyy-MM-dd"))

Dim fecha As String = If(mes, fecham, fechasm)

Dim Condicion As String = String.Format("id\_programa = {0} and " + fecha, programaID)

Dim Tablas As String = "fechaprograma"

TBuscada = PROGRAMAS

If Not (bwCargador.IsBusy) Then

bwCargador.RunWorkerAsync(PSQL(Columna, Tablas, Condicion))

ModLog.Guardar(PSQL(Columna, Tablas, Condicion))

End If

End Sub

Private Sub cbFMes\_CheckedChanged(sender As Object, e As EventArgs) Handles cbFMes.CheckedChanged

BFecha(cbFMes.Checked)

End Sub

Private Sub cbBMA\_CheckedChanged(sender As Object, e As EventArgs) Handles cbBMA.CheckedChanged

BFechaRango(cbBMA.Checked)

End Sub

Private Sub mcFecha\_DateSelected(sender As Object, e As DateRangeEventArgs) Handles mcFecha.DateSelected

BFechaRango(cbBMA.Checked)

End Sub

Private Sub btnBorrarF\_Click(sender As Object, e As EventArgs) Handles btnBorrarF.Click

If Not IsNothing(dt\_funcionario) Then

If (dt\_funcionario.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_funcionario, dgvFuncionarios, 1)

Dim Id1() As String = ObtenerCheck(dt\_funcionario, dgvFuncionarios, 4)

If Not Id.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(FUNTRABAJA, {programaID}, False, Id, Id1)

formDelete.ShowDialog(Me)

BuscarFuncionario()

End If

End If

End If

End Sub

Private Sub cbRP\_CheckedChanged(sender As Object, e As EventArgs) Handles cbRP.CheckedChanged

BuscarFuncionario()

End Sub

Private Sub btnTerminarF\_Click(sender As Object, e As EventArgs) Handles btnTerminarF.Click

If Not IsNothing(dt\_funcionario) Then

If (dt\_funcionario.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_funcionario, dgvFuncionarios, 1)

Dim Id1() As String = ObtenerCheck(dt\_funcionario, dgvFuncionarios, 4)

If Not Id.Length = 0 Then

USQL("funtrabaja", String.Format("fecha\_finalizacion='{0}'", Format(dtpTF.Value, "yyyy-MM-dd")), String.Format("id\_programa='{0}'", programaID) + " and " + CreadorCondicion("ID\_TrabajaComo", Id) + " and " + CreadorCondicion("fecha\_inicio", Id1, True) + String.Format(" and fecha\_inicio<='{0}'", Format(dtpTF.Value, "yyyy-MM-dd")))

BuscarFuncionario()

End If

End If

End If

dtpTF.Value = Now

End Sub

Private Sub cbPagados\_CheckedChanged(sender As Object, e As EventArgs) Handles cbPagados.CheckedChanged

BuscarCuota()

End Sub

Private Sub VaciarCuota()

dtpFE.Value = Now

dtpFP.Value = Now

cbP.Checked = False

nudValor.Value = 0

End Sub

Private Sub CambiarICuota()

If (CuotaId = -1) Then

btnBorrarC.Text = "Borrar"

btnInsertarC.Text = "Añadir"

gbAlquiler.Text = "Ingreso"

VaciarCuota()

Else

btnBorrarC.Text = "Cancelar"

btnInsertarC.Text = "Actualizar"

gbAlquiler.Text = "Edición"

End If

End Sub

Private Sub btnInsertarC\_Click(sender As Object, e As EventArgs) Handles btnInsertarC.Click

If (CuotaId = -1) Then

ISQL("programacuota", "id\_programa,fecha\_emision, fecha\_pago,precio", String.Format("'{0}','{1}',{2},'{3}'", programaID, Format(dtpFE.Value, "yyyy-MM-dd"), If(cbP.Checked, "'" + Format(dtpFP.Value, "yyyy-MM-dd") + "'", "null"), nudValor.Value))

VaciarCuota()

BuscarCuota()

Else

USQL("programacuota", String.Format("fecha\_emision='{0}', fecha\_pago={1},precio='{2}'", Format(dtpFE.Value, "yyyy-MM-dd"), If(cbP.Checked, "'" + Format(dtpFP.Value, "yyyy-MM-dd") + "'", "null"), nudValor.Value), String.Format("id\_programa\_cuota='{0}'", CuotaId))

CuotaId = -1

CambiarICuota()

BuscarCuota()

End If

End Sub

Private Sub btnBorrarC\_Click(sender As Object, e As EventArgs) Handles btnBorrarC.Click

If (CuotaId = -1) Then

If Not IsNothing(dt\_Cuotas) Then

If (dt\_Cuotas.Rows.Count > 0) Then

Dim RId() As String = ObtenerCheck(dt\_Cuotas, dgvVerCuota, 0)

If Not RId.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(CUOTA, RId, False)

formDelete.ShowDialog(Me)

BuscarCuota()

End If

End If

End If

Else

CuotaId = -1

CambiarICuota()

End If

End Sub

Private Sub dgvVerCuota\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvVerCuota.CellDoubleClick

Dim i() As String = CargarID(dt\_Cuotas, dgvVerCuota, {0, 1, 2, 3})

If (i(0) <> 0) Then

CuotaId = i(0)

dtpFE.Value = i(1)

dtpFP.Value = If(i(2) = "", Now, i(2))

cbP.Checked = If(i(2) = "", False, True)

nudValor.Value = i(3)

End If

CambiarICuota()

End Sub

Private Sub dtpYearCuota\_ValueChanged(sender As Object, e As EventArgs) Handles dtpYearCuota.ValueChanged

BuscarCuota()

End Sub

Private Sub dtpFE\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFE.ValueChanged

If (dtpFE.Value > dtpFP.Value) Then

dtpFP.Value = dtpFE.Value

End If

dtpFP.MinDate = dtpFE.Value

End Sub

Private Sub txtHI\_ValueChanged(sender As Object, e As EventArgs) Handles txtHI.ValueChanged

If (txtHI.Value > txtHF.Value) Then

txtHF.Value = txtHI.Value

End If

txtHF.MinDate = txtHI.Value

End Sub

Private Sub dgvFuncionarios\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFuncionarios.CellDoubleClick

Dim i() As String = CargarID(dt\_funcionario, dgvFuncionarios, {0, 2, 6, 7})

If (i(0) > 0) Then

Dim formFunc As New frmFuncionario(i)

AddHandler formFunc.FormClosed, AddressOf FormFunc\_FormClosed

formFunc.ShowDialog()

End If

End Sub

Private Sub FormFunc\_FormClosed(sender As Object, e As FormClosedEventArgs)

BuscarFuncionario()

End Sub

Private Sub btnCF\_Click(sender As Object, e As EventArgs) Handles btnCF.Click

If Not IsNothing(dt\_funcionario) Then

If (dt\_funcionario.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_funcionario, dgvFuncionarios, 1)

Dim Id1() As String = ObtenerCheck(dt\_funcionario, dgvFuncionarios, 4)

If Not Id.Length = 0 Then

USQL("funtrabaja", "fecha\_finalizacion=null", String.Format("id\_programa='{0}'", programaID) + " and " + CreadorCondicion("ID\_TrabajaComo", Id) + " and " + CreadorCondicion("fecha\_inicio", Id1, True) + String.Format(" and fecha\_inicio<='{0}'", Format(dtpTF.Value, "yyyy-MM-dd")))

BuscarFuncionario()

End If

End If

End If

End Sub

End Class

frmPublicidad.vb

Imports System.ComponentModel

Public Class frmPublicidad

Dim publicidadID As Integer

private editando As Boolean = False

Dim datos() As String

Dim datosI() As String

Private dt\_ProgramasP As DataTable

Private dt\_FProgramas As DataTable

Private dt\_FEventos As DataTable

Private TBuscada As Byte = 0

Private dt\_fechas As DataTable

Private dt\_fechasA As DataTable

Dim position() As String

Dim positionTanda() As String

Dim positionPrograma() As String

Dim positionEvento() As String

Dim pos() As UInt16 = {0, 0, 0, 0}

Dim empresaID As Integer = 0

Dim fecha1 As String

Dim fecha2 As String

Dim fechaP1 As String

Dim fechaP2 As String

Dim fechaE1 As String

Dim fechaE2 As String

Private dt\_Cuotas As DataTable

Private CuotaId As Integer = -1

Public Sub New(ByVal pid As Integer)

InitializeComponent()

publicidadID = pid

End Sub

Public Sub Buscar()

Dim columnas() As String = {"Nombre", "Tema", "id\_empresa"}

datosI = BuscarDatos("publicidad", columnas, "id\_publicidad", publicidadID)

Rellenar()

End Sub

Private Sub Rellenar()

txtNombre.Text = datosI(0)

txtTema.Text = datosI(1)

empresaID = datosI(2)

CargarCombo()

End Sub

Sub CargarCombo()

dt\_Empresa = DevolverTabla(PSQL("id\_Empresa, Nombre, Telefono, Mail", "Empresa", "True"))

LlenarCombo(cbEmpresa, dt\_Empresa, "Nombre")

If Not IsNothing(dt\_Empresa) Then

ExtraerDatos()

Else

MessageBox.Show("Debe tener empresas ingresadas")

Close()

End If

cbEmpresa.SelectedIndex = pos(0)

End Sub

Sub CargarComboTandas()

LlenarCombo(cbTanda, dt\_tandasCon, "Horas")

If Not IsNothing(dt\_tandasCon) Then

ExtraerDatosTan()

Else

MessageBox.Show("Debe tener tandas ingresadas")

tcP.SelectedIndex = 0

End If

cbTanda.SelectedIndex = pos(1)

End Sub

Sub CargarComboPrograma()

LlenarCombo(cbPrograma, dt\_ProgramasP, "Nombre\_Programa")

If Not IsNothing(dt\_ProgramasP) Then

ExtraerDatosProg()

Else

MessageBox.Show("No se encontró el programa")

End If

cbPrograma.SelectedIndex = pos(2)

End Sub

Sub CargarComboEvento()

LlenarCombo(cbEvento, dt\_FEventos, "Nombre")

If Not IsNothing(dt\_FEventos) Then

ExtraerDatosEven()

Else

MessageBox.Show("No se encontró el evento")

End If

cbEvento.SelectedIndex = pos(3)

End Sub

Public Sub ExtraerDatosProg()

ReDim positionPrograma(dt\_ProgramasP.Rows.Count - 1)

For j As Integer = 0 To dt\_ProgramasP.Rows.Count - 1

positionPrograma(j) = dt\_ProgramasP.Rows(j).Item(0).ToString

Next

End Sub

Public Sub ExtraerDatosEven()

ReDim positionEvento(dt\_FEventos.Rows.Count - 1)

For j As Integer = 0 To dt\_FEventos.Rows.Count - 1

positionEvento(j) = dt\_FEventos.Rows(j).Item(0).ToString

Next

End Sub

Public Sub ExtraerDatosTan()

ReDim positionTanda(dt\_tandasCon.Rows.Count - 1)

For j As Integer = 0 To dt\_tandasCon.Rows.Count - 1

positionTanda(j) = dt\_tandasCon.Rows(j).Item(0).ToString

Next

End Sub

Public Sub ExtraerDatos()

ReDim position(dt\_Empresa.Rows.Count - 1)

For j As Integer = 0 To dt\_Empresa.Rows.Count - 1

position(j) = dt\_Empresa.Rows(j).Item(0).ToString

Next

For i As Integer = 0 To position.Length - 1

If empresaID = position(i) Then

pos(0) = i + 1

Exit For

End If

Next

End Sub

Private Sub frmPublicidad\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

If publicidadID <> -1 Then

Buscar()

btnSalir.Select()

If Not PoseePermiso("Publicidad", "a") Then

btnBorrar.Visible = False

btnEditar.Visible = False

btnBorrarC.Visible = False

btnInsertarC.Visible = False

ocultar()

ElseIf Not PoseePermiso("Evento", "a") Then

tcP.TabPages.RemoveByKey("tbEventos")

ElseIf Not PoseePermiso("Programa", "a") Then

tcP.TabPages.RemoveByKey("tbProgramas")

End If

Else

ocultar()

tcP.TabPages.RemoveByKey("tbCuotas")

btnEditar.Text = "Insertar"

btnBorrar.Visible = False

Activar()

CargarCombo()

End If

btnSalir.Select()

End Sub

Private Sub ocultar()

tcP.TabPages.RemoveByKey("tbTandas")

tcP.TabPages.RemoveByKey("tbTandasE")

tcP.TabPages.RemoveByKey("tbProgramas")

tcP.TabPages.RemoveByKey("tbEventos")

End Sub

Sub Activar()

txtNombre.ReadOnly = editando

txtTema.ReadOnly = editando

editando = Not editando

cbEmpresa.Enabled = editando

End Sub

Private Sub btnSalir\_Click(sender As Object, e As EventArgs) Handles btnSalir.Click

If Not editando Or publicidadID = -1 Then

Close()

Else

Rellenar()

Alternar()

End If

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

PrepararDelete("publicidad", "id\_publicidad", {publicidadID})

Close()

End Sub

Private Sub Alternar()

Activar()

btnEditar.Text = If(editando, "Guardar", "Editar")

btnSalir.Text = If(editando, "Cancelar", "Salir")

End Sub

Sub vaciar()

txtNombre.Text = ""

txtTema.Text = ""

cbEmpresa.SelectedIndex = -1

End Sub

Private Sub ActualizarDatos()

Dim tem As String = txtTema.Text

Dim nom As String = txtNombre.Text

Dim emp As String = If(cbEmpresa.SelectedIndex <= 0, "null", position(cbEmpresa.SelectedIndex - 1))

datos = {nom, tem, emp}

End Sub

Private Sub btnEditar\_Click(sender As Object, e As EventArgs) Handles btnEditar.Click

If (cbEmpresa.SelectedIndex <= 0) Then

MessageBox.Show("Debe seleccionar una empresa")

Exit Sub

End If

If publicidadID = -1 Then

ActualizarDatos()

PrepararInsert("publicidad", datos)

vaciar()

ElseIf editando Then

ActualizarDatos()

If Not CompararValores(VaciarNull(datos), datosI) Then

PrepararUpdate("publicidad", datos, publicidadID)

datosI = VaciarNull(datos)

End If

Alternar()

Else

Alternar()

End If

End Sub

Private Sub tcP\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles tcP.SelectedIndexChanged

If (tcP.SelectedIndex() = 1 Or tcP.SelectedIndex() = 2 Or tcP.SelectedIndex() = 5) Then

If Not (bwDatos.IsBusy) Then

bwDatos.RunWorkerAsync(tcP.SelectedIndex())

End If

End If

End Sub

Private Sub bwDatos\_DoWork(sender As Object, e As DoWorkEventArgs) Handles bwDatos.DoWork

Select Case e.Argument

Case 1

TBuscada = TANDASHORAS

dt\_tandasCon = DevolverTabla(PSQL("Hora\_inicio, concat('Tanda ',Hora\_Inicio,' ',Hora\_fin) as 'Horas'", "Tanda", "True"))

ModLog.Guardar(PSQL("Hora\_inicio, concat('Tanda ',Hora\_Inicio,' ',Hora\_fin) as 'Horas'", "Tanda", "True"))

Case 2

TBuscada = PUBLICIDAD

Dim condicion = "true"

dt\_fechasA = DevolverTabla(PSQL("distinct a.hora\_inicio as 'Hora de tanda', fecha\_inicio as 'Fecha Inicio', fecha\_finalizacion as 'Fecha Finalización'", "aparecepubli a left join tanda t on t.hora\_inicio = null", condicion))

Case 3

TBuscada = PROGRAMAS

Dim condicion = "false"

If Not String.IsNullOrWhiteSpace(txtNombreP.Text) Then

condicion = String.Format("Nombre\_Programa like '%{0}%'", txtNombreP.Text)

End If

dt\_ProgramasP = DevolverTabla(PSQL("id\_programa, Nombre\_Programa", "programa", condicion))

Case 4

TBuscada = EVENTO

Dim condicion = "false"

If Not String.IsNullOrWhiteSpace(txtNEvento.Text) Then

condicion = String.Format("nombre like '%{0}%'", txtNEvento.Text)

End If

dt\_FEventos = DevolverTabla(PSQL("id\_evento, nombre", "evento", condicion))

ModLog.Guardar(PSQL("id\_evento, nombre", "evento", condicion))

Case 5

Dim Condicion As String = String.Format("id\_publicidad='{0}' and Fecha\_Pago is{1}null and year(fecha\_emision)={2}", publicidadID, If(cbPagados.Checked, " not ", " "), Year(dtpYearCuota.Value))

TBuscada = CUOTAPUBLICIDAD

Dim Columna As String = "id\_publicidadcuota, fecha\_emision as 'Fecha de emisión', fecha\_pago as 'Fecha de pago', precio as 'Valor'"

Dim Tablas As String = "publicidadcuota"

dt\_Cuotas = DevolverTabla(PSQL(Columna, Tablas, Condicion))

ModLog.Guardar(PSQL(Columna, Tablas, Condicion))

End Select

End Sub

Private Sub ATAntigua()

ActualizarTablaC(dt\_fechasA, dgvTE, False)

End Sub

Private Sub bwDatos\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles bwDatos.RunWorkerCompleted

Select Case TBuscada

Case TANDASHORAS

CargarComboTandas()

Case PROGRAMAS

CargarComboPrograma()

Case PUBLICIDAD

ATAntigua()

Case EVENTO

CargarComboEvento()

Case CUOTAPUBLICIDAD

ActualizarTablaC(dt\_Cuotas, dgvVerCuota, True)

End Select

End Sub

Private Sub cbTanda\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cbTanda.SelectedIndexChanged

BuscarF()

End Sub

Private Sub BuscarF()

CargarPubliT(dt\_fechas, dgvFechas, publicidadID, If(cbTanda.SelectedIndex <= 0, "00:00:00", positionTanda(cbTanda.SelectedIndex - 1)), Format(dtpFI.Value, "yyyy-MM-dd"), Format(dtpFF.Value, "yyyy-MM-dd"))

End Sub

Private Sub BuscarFP()

CargarPubliP(dt\_FProgramas, dgvProgramaP, publicidadID, If(cbPrograma.SelectedIndex <= 0, "0", positionPrograma(cbPrograma.SelectedIndex - 1)), Format(dtpFIP.Value, "yyyy-MM-dd"), Format(dtpFFP.Value, "yyyy-MM-dd"))

End Sub

Private Sub BuscarFE()

CargarPubliE(dt\_FEventos, dgvAEvento, publicidadID, If(cbEvento.SelectedIndex <= 0, "0", positionEvento(cbEvento.SelectedIndex - 1)), Format(dtpFIE.Value, "yyyy-MM-dd"), Format(dtpFFE.Value, "yyyy-MM-dd"))

End Sub

Private Sub dtpFI\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFI.ValueChanged

If (dtpFF.Value < dtpFI.Value) Then

dtpFF.Value = dtpFI.Value

End If

dtpFF.MinDate = dtpFI.Value

BuscarF()

End Sub

Private Sub dtpFF\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFF.ValueChanged

BuscarF()

End Sub

Private Sub btnIngresar\_Click(sender As Object, e As EventArgs) Handles btnIngresar.Click

FechasMax()

End Sub

Private Sub FechasMaxP(Optional ingresar As Boolean = True)

fechaP1 = Format(dtpFIP.Value, "yyyy-MM-dd")

fechaP2 = Format(dtpFFP.Value, "yyyy-MM-dd")

If (cbPrograma.SelectedIndex > 0) Then

If (dgvProgramaP.Rows.Count > 0) Then

For i As Integer = 0 To dgvProgramaP.Rows.Count - 1

Dim fechaN1 As String = dgvProgramaP.Rows(i).Cells(0).Value().ToString

Dim fechaN2 As String = dgvProgramaP.Rows(i).Cells(1).Value().ToString

If (CDate(fechaN1) < CDate(fechaP1)) Then

fechaP1 = Format(CDate(fechaN1), "yyyy-MM-dd")

End If

If (CDate(fechaN2) > CDate(fechaP2)) Then

fechaP2 = Format(CDate(fechaN2), "yyyy-MM-dd")

End If

Next

BSQL("pmuestrapubli", String.Format("id\_publicidad='{0}' and id\_programa='{1}' and fecha\_inicio >= '{2}' and fecha\_finalizacion <= '{3}'", publicidadID, positionPrograma(cbPrograma.SelectedIndex - 1), fechaP1, fechaP2))

End If

If (ingresar) Then

ISQL("pmuestrapubli", "id\_publicidad, id\_programa, fecha\_inicio, fecha\_finalizacion", String.Format("'{0}','{1}','{2}','{3}'", publicidadID, positionPrograma(cbPrograma.SelectedIndex - 1), fechaP1, fechaP2))

End If

BuscarFP()

Else

MessageBox.Show("Debe seleccionar un programa para agendar la publicidad")

End If

End Sub

Private Sub FechasMaxE(Optional ingresar As Boolean = True)

fechaE1 = Format(dtpFIE.Value, "yyyy-MM-dd")

fechaE2 = Format(dtpFFE.Value, "yyyy-MM-dd")

If (cbEvento.SelectedIndex > 0) Then

If (dgvAEvento.Rows.Count > 0) Then

For i As Integer = 0 To dgvAEvento.Rows.Count - 1

Dim fechaN1 As String = dgvAEvento.Rows(i).Cells(0).Value().ToString

Dim fechaN2 As String = dgvAEvento.Rows(i).Cells(1).Value().ToString

If (CDate(fechaN1) < CDate(fechaE1)) Then

fechaE1 = Format(CDate(fechaN1), "yyyy-MM-dd")

End If

If (CDate(fechaN2) > CDate(fechaE2)) Then

fechaE2 = Format(CDate(fechaN2), "yyyy-MM-dd")

End If

Next

BSQL("eventomuestrapubli", String.Format("id\_publicidad='{0}' and id\_evento='{1}' and fecha\_inicio >= '{2}' and fecha\_finalizacion <= '{3}'", publicidadID, positionEvento(cbEvento.SelectedIndex - 1), fechaE1, fechaE2))

End If

If (ingresar) Then

ISQL("eventomuestrapubli", "id\_publicidad, id\_evento, fecha\_inicio, fecha\_finalizacion", String.Format("'{0}','{1}','{2}','{3}'", publicidadID, positionEvento(cbEvento.SelectedIndex - 1), fechaE1, fechaE2))

End If

BuscarFE()

Else

MessageBox.Show("Debe seleccionar un evento para agendar la publicidad")

End If

End Sub

Private Sub FechasMax(Optional ingresar As Boolean = True)

fecha1 = Format(dtpFI.Value, "yyyy-MM-dd")

fecha2 = Format(dtpFF.Value, "yyyy-MM-dd")

If (cbTanda.SelectedIndex > 0) Then

If (dgvFechas.Rows.Count > 0) Then

For i As Integer = 0 To dgvFechas.Rows.Count - 1

Dim fechaN1 As String = dgvFechas.Rows(i).Cells(0).Value().ToString

Dim fechaN2 As String = dgvFechas.Rows(i).Cells(1).Value().ToString

If (CDate(fechaN1) < CDate(fecha1)) Then

fecha1 = Format(CDate(fechaN1), "yyyy-MM-dd")

End If

If (CDate(fechaN2) > CDate(fecha2)) Then

fecha2 = Format(CDate(fechaN2), "yyyy-MM-dd")

End If

Next

BSQL("aparecepubli", String.Format("id\_publicidad='{0}' and Hora\_Inicio='{1}' and fecha\_inicio >= '{2}' and fecha\_finalizacion <= '{3}'", publicidadID, positionTanda(cbTanda.SelectedIndex - 1), fecha1, fecha2))

End If

If (ingresar) Then

ISQL("aparecepubli", "id\_publicidad, hora\_inicio, fecha\_inicio, fecha\_finalizacion", String.Format("'{0}','{1}','{2}','{3}'", publicidadID, positionTanda(cbTanda.SelectedIndex - 1), fecha1, fecha2))

End If

BuscarF()

Else

MessageBox.Show("Debe seleccionar una tanda para agendar la publicidad")

End If

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles btnBorrrT.Click

FechasMax(False)

End Sub

Private Sub btnPrograma\_Click(sender As Object, e As EventArgs) Handles btnPrograma.Click

If Not bwDatos.IsBusy Then

bwDatos.RunWorkerAsync(3) 'PSQL("ID\_Programa, Nombre\_Programa as Nombre, Descripcion", "programa", condicion))

End If

End Sub

Private Sub dtpFIP\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFIP.ValueChanged

If (dtpFFP.Value < dtpFIP.Value) Then

dtpFFP.Value = dtpFIP.Value

End If

dtpFFP.MinDate = dtpFIP.Value

BuscarFP()

End Sub

Private Sub cbPrograma\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cbPrograma.SelectedIndexChanged

BuscarFP()

End Sub

Private Sub dtpFFP\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFFP.ValueChanged

BuscarFP()

End Sub

Private Sub btnIngresarP\_Click(sender As Object, e As EventArgs) Handles btnIngresarP.Click

FechasMaxP()

End Sub

Private Sub btnBorrarP\_Click(sender As Object, e As EventArgs) Handles btnBorrarP.Click

FechasMaxP(False)

End Sub

Private Sub btnBPubli\_Click(sender As Object, e As EventArgs) Handles btnBPubli.Click

If Not IsNothing(dt\_fechasA) Then

If (dt\_fechasA.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_fechasA, dgvTE, 0)

Dim Id2() As String = ObtenerCheck(dt\_fechasA, dgvTE, 1)

Dim Id3() As String = ObtenerCheck(dt\_fechasA, dgvTE, 2)

If Not Id.Length = 0 Then

BSQL("aparecepubli", String.Format("id\_publicidad='{0}'", publicidadID) + " and " + CreadorCondicion("Hora\_Inicio", Id) + " and " + CreadorCondicion("fecha\_inicio", Id2, True) + " and " + CreadorCondicion("fecha\_finalizacion", Id3, True))

If Not (bwDatos.IsBusy) Then

bwDatos.RunWorkerAsync(2)

End If

End If

End If

End If

End Sub

Private Sub dgvTE\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvTE.CellClick, dgvVerCuota.CellClick

ClickCheck(sender, e.ColumnIndex)

End Sub

Private Sub dgvHeaderClick(sender As Object, e As DataGridViewCellMouseEventArgs) Handles dgvTE.ColumnHeaderMouseClick, dgvVerCuota.ColumnHeaderMouseClick

CheckAll(sender, e.ColumnIndex)

End Sub

Private Sub BuscarCuota()

If Not (bwDatos.IsBusy) Then

bwDatos.RunWorkerAsync(5)

End If

End Sub

Private Sub dtpYearCuota\_ValueChanged(sender As Object, e As EventArgs) Handles dtpYearCuota.ValueChanged

BuscarCuota()

End Sub

Private Sub cbPagados\_CheckedChanged(sender As Object, e As EventArgs) Handles cbPagados.CheckedChanged

BuscarCuota()

End Sub

Private Sub CambiarICuota()

If (CuotaId = -1) Then

btnBorrarC.Text = "Borrar"

btnInsertarC.Text = "Añadir"

gbAlquiler.Text = "Ingreso"

VaciarCuota()

Else

btnBorrarC.Text = "Cancelar"

btnInsertarC.Text = "Actualizar"

gbAlquiler.Text = "Edición"

End If

End Sub

Private Sub btnBorrarC\_Click(sender As Object, e As EventArgs) Handles btnBorrarC.Click

If (CuotaId = -1) Then

If Not IsNothing(dt\_Cuotas) Then

If (dt\_Cuotas.Rows.Count > 0) Then

Dim RId() As String = ObtenerCheck(dt\_Cuotas, dgvVerCuota, 0)

If Not RId.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(CUOTAPUBLICIDAD, RId, False)

formDelete.ShowDialog(Me)

BuscarCuota()

End If

End If

End If

Else

CuotaId = -1

CambiarICuota()

End If

End Sub

Private Sub VaciarCuota()

dtpFE.Value = Now

dtpFP.Value = Now

cbP.Checked = False

nudValor.Value = 0

End Sub

Private Sub btnInsertarC\_Click(sender As Object, e As EventArgs) Handles btnInsertarC.Click

If (CuotaId = -1) Then

ISQL("publicidadcuota", "id\_publicidad,fecha\_emision, fecha\_pago,precio", String.Format("'{0}','{1}',{2},'{3}'", publicidadID, Format(dtpFE.Value, "yyyy-MM-dd"), If(cbP.Checked, "'" + Format(dtpFP.Value, "yyyy-MM-dd") + "'", "null"), nudValor.Value))

VaciarCuota()

BuscarCuota()

Else

USQL("publicidadcuota", String.Format("fecha\_emision='{0}', fecha\_pago={1},precio='{2}'", Format(dtpFE.Value, "yyyy-MM-dd"), If(cbP.Checked, "'" + Format(dtpFP.Value, "yyyy-MM-dd") + "'", "null"), nudValor.Value), String.Format("id\_publicidadcuota='{0}'", CuotaId))

CuotaId = -1

CambiarICuota()

BuscarCuota()

End If

End Sub

Private Sub dgvVerCuota\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvVerCuota.CellDoubleClick

Dim i() As String = CargarID(dt\_Cuotas, dgvVerCuota, {0, 1, 2, 3})

If (i(0) <> 0) Then

CuotaId = i(0)

dtpFE.Value = i(1)

dtpFP.Value = If(i(2) = "", Now, i(2))

cbP.Checked = i(2) IsNot ""

nudValor.Value = i(3)

End If

CambiarICuota()

End Sub

Private Sub btnBEvento\_Click(sender As Object, e As EventArgs) Handles btnBEvento.Click

If Not bwDatos.IsBusy Then

bwDatos.RunWorkerAsync(4) 'PSQL("ID\_Programa, Nombre\_Programa as Nombre, Descripcion", "programa", condicion))

End If

End Sub

Private Sub cbEvento\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cbEvento.SelectedIndexChanged

BuscarFE()

End Sub

Private Sub dtpFFE\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFFE.ValueChanged

BuscarFE()

End Sub

Private Sub dtpFIE\_ValueChanged(sender As Object, e As EventArgs) Handles dtpFIE.ValueChanged

If (dtpFFE.Value < dtpFIE.Value) Then

dtpFFE.Value = dtpFIE.Value

End If

dtpFFE.MinDate = dtpFIE.Value

BuscarFE()

End Sub

Private Sub btnBPEvento\_Click(sender As Object, e As EventArgs) Handles btnBPEvento.Click

FechasMaxE(False)

End Sub

Private Sub btnIEvento\_Click(sender As Object, e As EventArgs) Handles btnIEvento.Click

FechasMaxE()

End Sub

Private Sub btnMP\_Click(sender As Object, e As EventArgs) Handles btnMP.Click

If (cbPrograma.SelectedIndex > 0) Then

Dim formPrograma As New frmPrograma(positionPrograma(cbPrograma.SelectedIndex - 1))

AddHandler formPrograma.FormClosed, AddressOf FormPrograma\_FormClosed

formPrograma.ShowDialog()

End If

End Sub

Private Sub FormPrograma\_FormClosed()

If Not bwDatos.IsBusy Then

bwDatos.RunWorkerAsync(3) 'PSQL("ID\_Programa, Nombre\_Programa as Nombre, Descripcion", "programa", condicion))

End If

End Sub

Private Sub btnMostrarE\_Click(sender As Object, e As EventArgs) Handles btnMostrarE.Click

If (cbEvento.SelectedIndex > 0) Then

Dim formEvento As New frmEvento(positionEvento(cbEvento.SelectedIndex - 1))

AddHandler formEvento.FormClosed, AddressOf FormEvento\_FormClosed

formEvento.ShowDialog()

End If

End Sub

Private Sub FormEvento\_FormClosed()

If Not bwDatos.IsBusy Then

bwDatos.RunWorkerAsync(4) 'PSQL("ID\_Programa, Nombre\_Programa as Nombre, Descripcion", "programa", condicion))

End If

End Sub

End Class

frmPrincipal.vb

Imports System.ComponentModel

Imports System.Drawing.Drawing2D

Imports System.IO

Public Class frmPrincipal

Private DescripcionP As String

Private PNombre As String = "Datos del programa"

Private TBuscada As Byte = 0

Private TBusca As New DataTable

Public dt\_Buscada As New DataTable

#Region "Carga y descarga"

Private Sub frmPrincipal\_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing

ModConector.desconectar()

End

End Sub

Private Sub frmPrincipal\_Closing(sender As Object, e As CancelEventArgs) Handles Me.Closing

GuardarNotas()

End Sub

Public Sub EPropiedades()

Width = Screen.PrimaryScreen.WorkingArea.Width \* 0.85

Height = Screen.PrimaryScreen.WorkingArea.Height \* 0.8

CenterToScreen()

dtp.BackColor = Color.FromArgb(64, 64, 64)

dtp.ForeColor = Color.White

dtpTanda.BackColor = Color.FromArgb(64, 64, 64)

dtpTanda.ForeColor = Color.White

End Sub

Public Sub ONBW()

BWProgramas.RunWorkerAsync()

BWTandas.RunWorkerAsync(True)

BWEventos.RunWorkerAsync()

End Sub

Private Sub frmPrincipal\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

EPropiedades()

ONBW()

LeerNotas()

QuitarTab()

End Sub

Private Sub QuitarTab()

If Not (PoseePermiso("Programa")) Then

tcSecciones.TabPages.RemoveByKey("tbMenu")

Else

If Not (PoseePermiso("Serie")) Then

tcSecciones.TabPages.RemoveByKey("tbSeries")

End If

End If

If Not (PoseePermiso("Video")) Then

TabControl1.TabPages.RemoveByKey("tbVideo")

End If

If Not (PoseePermiso("Empresa")) Then

tcSecciones.TabPages.RemoveByKey("tbEmpresa")

End If

If Not (PoseePermiso("Publicidad")) Then

tcSecciones.TabPages.RemoveByKey("tbPublicidad")

End If

If Not (PoseePermiso("Funcionario")) Then

tcSecciones.TabPages.RemoveByKey("tbFuncionario")

End If

If Not (PoseePermiso("Evento")) Then

tcSecciones.TabPages.RemoveByKey("tbEvento")

End If

End Sub

#End Region

Private Sub BWNumberOne\_DoWork(sender As Object, e As DoWorkEventArgs) Handles BWProgramas.DoWork

dt\_programa = ModConector.APrograma(dtp.Value.Date)

End Sub

Private Sub ActualizarEvento()

ActualizarTablaC(dt\_evento, dgvEventos)

dgvEventos.ClearSelection()

End Sub

Public Sub dgvProgramaColor()

Dim fin As TimeSpan

Dim inicio As TimeSpan

Dim colores As Color = dgvPrograma.DefaultCellStyle.ForeColor

Dim colorNuevo As Color

Dim estado As String

Dim Activo As Integer = -1

If (Now.Date >= dtp.Value.Date) And dgvPrograma.Rows.Count > 0 Then

For i As Integer = 0 To dgvPrograma.Rows.Count - 1

If Not (TimeSpan.TryParse(dgvPrograma.Rows(i).Cells(0).Value().ToString, inicio) And TimeSpan.TryParse(dgvPrograma.Rows(i).Cells(1).Value().ToString, fin)) Then

Exit Sub

End If

If (inicio < Now.TimeOfDay Or Now.Date > dtp.Value.Date) Then

If (Now.TimeOfDay < fin And Now.Date = dtp.Value.Date) Then

colorNuevo = Color.FromArgb(245, 94, 94)

estado = "Activo"

dgvPrograma.Rows(i).Selected = True

Activo = i

Else

estado = "Finalizado"

colorNuevo = Color.FromArgb(154, 94, 245)

End If

dgvPrograma.Rows(i).Cells(dgvPrograma.Rows(i).Cells.Count - 1).Style.ForeColor = colorNuevo

Else

If Activo = -1 Then

Activo = i

dgvPrograma.Rows(i).Selected = True

End If

estado = "Activo"

End If

dgvPrograma.Rows(i).Cells(dgvPrograma.Rows(i).Cells.Count - 1).Value = estado

Next

Else

For i As Integer = 0 To dgvPrograma.Rows.Count - 1

dgvPrograma.Rows(i).Cells(dgvPrograma.Rows(i).Cells.Count - 1).Value = "Proximo"

Next

End If

End Sub

Private Sub BWNumberOne\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles BWProgramas.RunWorkerCompleted

'ActualizarProgramas()

ActualizarTablaC(dt\_programa, dgvPrograma)

NumerTime(dt\_programa, RecPA, 2)

dgvProgramaColor()

Funcionarios()

End Sub

Public Sub GuardarNotas()

Dim ruta As String = "..\User\"

Dim archivo As String = "Notas.txt"

Dim escribir As New StreamWriter(ruta & archivo, False)

escribir.Write(TBNotas.Text)

escribir.Close()

End Sub

Public Sub LeerNotas()

Dim ruta As String = "..\User\"

Dim archivo As String = "Notas.txt"

If File.Exists(ruta & archivo) Then

TBNotas.Text = ""

Dim leer As New StreamReader(ruta & archivo)

TBNotas.Text = leer.ReadToEnd()

leer.Close()

End If

End Sub

Private Sub BWDPRogramas\_DoWork(sender As Object, e As DoWorkEventArgs) Handles BWDPRogramas.DoWork

If (dgvPrograma.Rows.Count > 0) And dgvPrograma.SelectedRows.Count > 0 Then

Dim idRow As Integer = dgvPrograma.SelectedRows(0).Index

If (idRow >= 0) And Not IsNothing(dt\_programa) Then

Dim id As Integer = CInt(dt\_programa.Rows(idRow)(0).ToString)

dt\_dprograma = ModConector.AFPrograma(dtp.Value.Date, id)

DescripcionP = ModConector.ADPrograma(id)

dt\_Ppubli = ModConector.APPublicidad(dtp.Value.Date, id)

PNombre = "Datos del programa : " + dt\_programa.Rows(idRow)(3).ToString

End If

Else

PNombre = "Datos del programa"

dt\_Ppubli = Nothing

dt\_dprograma = Nothing

End If

End Sub

Private Sub BWDPRogramas\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles BWDPRogramas.RunWorkerCompleted

ActualizarTablaC(dt\_dprograma, dgvFuncionarios, True, {0, 3})

ActualizarTablaC(dt\_Ppubli, dgvPPublicidades) 'fix

GBFuncionarios.Text = PNombre

TBDescripcion.Text = DescripcionP

dgvPrograma.ClearSelection()

End Sub

Public Sub Funcionarios()

If Not (BWDPRogramas.IsBusy) Then

If Not (BWProgramas.IsBusy) Then

BWDPRogramas.RunWorkerAsync()

End If

End If

End Sub

Private Sub dgvPrograma\_Click(sender As Object, e As EventArgs)

Funcionarios()

End Sub

Private Sub BWEventos\_DoWork(sender As Object, e As DoWorkEventArgs) Handles BWEventos.DoWork

If PoseePermiso("Eventos") Then

dt\_evento = ModConector.AEventos()

End If

End Sub

Private Sub BWEventos\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles BWEventos.RunWorkerCompleted

ActualizarEvento()

End Sub

Private Sub BWPublicidades\_DoWork(sender As Object, e As DoWorkEventArgs) Handles BWPublicidades.DoWork

If dgvTandas.Rows.Count > 0 And dgvTandas.SelectedRows.Count > 0 Then

Dim Horas As String = MysqlHM(CDate(CargarID(dt\_tandas, dgvTandas, {0})(0)))

If Not IsNothing(dt\_tandas) Then

dt\_publi = ModConector.APublicidad(dtpTanda.Value, Horas)

End If

Else

dt\_publi = Nothing

End If

End Sub

Private Sub BWPublicidades\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles BWPublicidades.RunWorkerCompleted

ActualizarTablaC(dt\_publi, dgvPublicidades) 'fix

End Sub

Private Sub BWTandas\_DoWork(sender As Object, e As DoWorkEventArgs) Handles BWTandas.DoWork

dt\_tandas = ModConector.ATandas(e.Argument)

End Sub

Public Sub Tandas()

ActualizarTablaC(dt\_tandas, dgvTandas, False) 'fix

NumerTime(dt\_tandas, RecTan, 1)

End Sub

Private Sub BWTandas\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles BWTandas.RunWorkerCompleted

Tandas()

If Not BWPublicidades.IsBusy Then

BWPublicidades.RunWorkerAsync()

End If

End Sub

Private Sub dgvTandas\_Click(sender As Object, e As EventArgs) Handles dgvTandas.Click

If Not BWPublicidades.IsBusy Then

BWPublicidades.RunWorkerAsync()

End If

End Sub

Private Sub dtp\_ValueChanged(sender As Object, e As EventArgs) Handles dtp.ValueChanged

If Not BWProgramas.IsBusy Then

BWProgramas.RunWorkerAsync()

End If

End Sub

Private Sub dtpTanda\_ValueChanged(sender As Object, e As EventArgs) Handles dtpTanda.ValueChanged

If Not BWPublicidades.IsBusy Then

BWPublicidades.RunWorkerAsync()

End If

End Sub

Private Sub BWBuscador\_DoWork(sender As Object, e As DoWorkEventArgs) Handles BWBuscador.DoWork

TBusca = DevolverTabla(e.Argument)

ModLog.Guardar(e.Argument)

End Sub

Private Sub btnbuscarv\_Click(sender As Object, e As EventArgs) Handles btnbuscarv.Click

TBuscada = VIDEO

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If (Not String.IsNullOrWhiteSpace(txtVnombre.Text)) Then

condicion = String.Format("v.nombre like '%{0}%'", txtVnombre.Text)

End If

If (cbFecha.Checked) Then

condicion += String.Format(" and fecha = '{0}'", Format(dtpfechavideo.Value, "yyyy-MM-dd").ToString)

End If

If (Not String.IsNullOrWhiteSpace(txtVcontenido.Text)) Then

condicion += String.Format(" and v.contenido like '%{0}%'", txtVcontenido.Text)

End If

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("id\_video, fecha as Fecha, v.nombre as Nombre, (select s.nombre from serie s where s.id\_serie=v.id\_serie) as Serie", "video v", condicion))

End If

End Sub

Private Sub BWBuscador\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles BWBuscador.RunWorkerCompleted

Select Case TBuscada

Case VIDEO

dt\_Video = TBusca

ActualizarTablaC(dt\_Video, dgvVB)

Case SERIE

dt\_Serie = TBusca

ActualizarTablaC(dt\_Serie, dgvBS)

Case EMPRESA

dt\_Empresa = TBusca

ActualizarTablaC(dt\_Empresa, dgvClientes)

Case PROGRAMAS

dt\_BPrograma = TBusca

ActualizarTablaC(dt\_BPrograma, dgvBProgramas)

Case PUBLICIDAD

dt\_BPubli = TBusca

ActualizarTablaC(dt\_BPubli, dgvPubliB)

Case FUNCIONARIO

dt\_BFuncionario = TBusca

ActualizarTablaC(dt\_BFuncionario, dgvFuncionarioBF)

Case FUNCION

dt\_BFuncion = TBusca

ActualizarTablaC(dt\_BFuncion, dgvFuncionesBFF)

Case EVENTO

dt\_BEvento = TBusca

ActualizarTablaC(dt\_BEvento, dgvBEvento)

End Select

TBusca = Nothing

TBuscada = 0

End Sub

Private Sub DgvVB\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvVB.CellDoubleClick

Dim i As Integer = CargarID(dt\_Video, dgvVB)

If (i <> -1 And i <> 0) Then

Dim formVideo As New frmVideo(i)

AddHandler formVideo.FormClosed, AddressOf FormVideo\_FormClosed

formVideo.ShowDialog()

End If

End Sub

Private Sub btnLimpiarBS\_Click(sender As Object, e As EventArgs) Handles btnLimpiarBS.Click

txtBSnombre.Clear()

ctpSerie.Value = Now.Date

cbS.Checked = False

ActualizarTablaC(Nothing, dgvBS, False) 'fix

End Sub

Private Sub frmPrincipal\_Resize(sender As Object, e As EventArgs) Handles Me.Resize

Refresh()

End Sub

Private Sub btnlimpiarv\_Click(sender As Object, e As EventArgs) Handles btnlimpiarv.Click

txtVcontenido.Clear()

txtVnombre.Clear()

dtpfechavideo.Value = Now.Date

cbFecha.Checked = False

ActualizarTablaC(Nothing, dgvVB, False) 'fix

End Sub

Private Sub btnBuscarBS\_Click(sender As Object, e As EventArgs) Handles btnBuscarBS.Click

TBuscada = SERIE

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If (Not String.IsNullOrWhiteSpace(txtBSnombre.Text)) Then

condicion = "nombre like '%" + txtBSnombre.Text + "%'"

End If

If (cbS.Checked) Then

condicion += String.Format(" and fecha\_finalizacion = '{0}'", Format(ctpSerie.Value, "yyyy-MM-dd").ToString)

End If

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("id\_serie, fecha\_finalizacion as Fecha, nombre as Nombre", "serie", condicion))

End If

End Sub

Private Sub dgvBS\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvBS.CellDoubleClick

Dim i() As String = CargarID(dt\_Serie, dgvBS, {0, 1, 2})

If (i(0) > 0) Then

Dim formSerie As New frmSerie(i)

AddHandler formSerie.FormClosed, AddressOf FormSerie\_FormClosed

formSerie.ShowDialog()

Else

' FIXME: Remove this, es de debug.

'TBNotas.Text += "a"

'Removido

End If

End Sub

Private Sub FormVideo\_FormClosed(sender As Object, e As FormClosedEventArgs)

btnbuscarv.PerformClick()

End Sub

Private Sub FormSerie\_FormClosed(sender As Object, e As FormClosedEventArgs)

btnBuscarBS.PerformClick()

End Sub

'' Nahuel del futuro aqui, vengo a decir gracias al Nahuel del pasado por hacer esto

Private Sub dgv\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvVB.CellClick, dgvBS.CellClick, dgvClientes.CellClick, dgvBProgramas.CellClick, dgvPubliB.CellClick, dgvFuncionarioBF.CellClick, dgvFuncionesBFF.CellClick, dgvBEvento.CellClick

ClickCheck(sender, e.ColumnIndex)

End Sub

Private Sub dgvHeaderClick(sender As Object, e As DataGridViewCellMouseEventArgs) Handles dgvVB.ColumnHeaderMouseClick, dgvBS.ColumnHeaderMouseClick, dgvClientes.ColumnHeaderMouseClick, dgvBProgramas.ColumnHeaderMouseClick, dgvPubliB.ColumnHeaderMouseClick, dgvFuncionarioBF.ColumnHeaderMouseClick, dgvFuncionesBFF.ColumnHeaderMouseClick, dgvBEvento.ColumnHeaderMouseClick

CheckAll(sender, e.ColumnIndex)

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

BorrarConfirmar(Me, dt\_Video, dgvVB, VIDEO, btnbuscarv)

End Sub

Private Sub btnBorrarS\_Click(sender As Object, e As EventArgs) Handles btnBorrarS.Click

BorrarConfirmar(Me, dt\_Serie, dgvBS, SERIE, btnBuscarBS)

End Sub

Private Sub btnIngresarV\_Click(sender As Object, e As EventArgs) Handles btnIngresarV.Click

Dim formVideo As New frmVideo(-1)

AddHandler formVideo.FormClosed, AddressOf FormVideo\_FormClosed

formVideo.ShowDialog()

End Sub

Private Sub btnsIngresar\_Click(sender As Object, e As EventArgs) Handles btnsIngresar.Click

Dim formSerie As New frmSerie({-1, Now.Date.ToString, ""})

AddHandler formSerie.FormClosed, AddressOf FormSerie\_FormClosed

formSerie.ShowDialog()

End Sub

Private Sub btncBuscar\_Click(sender As Object, e As EventArgs) Handles btncBuscar.Click

TBuscada = EMPRESA

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If (Not String.IsNullOrWhiteSpace(txtCNombre.Text)) Then

condicion = "Nombre like '%" + txtCNombre.Text + "%'"

End If

If (Not String.IsNullOrWhiteSpace(txtCTel.Text)) Then

condicion += " and Telefono = '" + txtCTel.Text + "'"

End If

If (Not String.IsNullOrWhiteSpace(txtCMail.Text)) Then

condicion += " and Mail = '" + txtCMail.Text + "'"

End If

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("id\_Empresa, Nombre, Telefono, Mail as 'E-Mail'", "empresa", condicion))

End If

End Sub

Private Sub btncBorrar\_Click(sender As Object, e As EventArgs) Handles btncBorrar.Click

BorrarConfirmar(Me, dt\_Empresa, dgvClientes, EMPRESA, btncBuscar)

End Sub

Private Sub FormCliente\_FormClosed(sender As Object, e As FormClosedEventArgs)

btncBuscar.PerformClick()

End Sub

Private Sub dgvClientes\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvClientes.CellDoubleClick

Dim i() As String = CargarID(dt\_Empresa, dgvClientes, {0, 1, 2, 3})

If (i(0) > 0) Then

Dim formCliente As New frmEmpresa(i)

AddHandler formCliente.FormClosed, AddressOf FormCliente\_FormClosed

formCliente.ShowDialog()

Else

TBNotas.Text += "a"

End If

End Sub

Private Sub btncIngresar\_Click(sender As Object, e As EventArgs) Handles btncIngresar.Click

Dim formCliente As New frmEmpresa({-1, "", "", ""})

AddHandler formCliente.FormClosed, AddressOf FormCliente\_FormClosed

formCliente.ShowDialog()

End Sub

Private Sub btncLimpiar\_Click(sender As Object, e As EventArgs) Handles btncLimpiar.Click

txtCNombre.Clear()

txtCMail.Clear()

txtCTel.Clear()

ActualizarTablaC(Nothing, dgvClientes, False) 'fix

End Sub

Private Sub btnLimpiarBP\_Click(sender As Object, e As EventArgs) Handles btnLimpiarBP.Click

txtNombreBP.Clear()

txtDescripcionBP.Clear()

ActualizarTablaC(Nothing, dgvBProgramas, False) 'fix

End Sub

Private Sub btnBuscarBP\_Click(sender As Object, e As EventArgs) Handles btnBuscarBP.Click

TBuscada = PROGRAMAS

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If Not String.IsNullOrWhiteSpace(txtNombreBP.Text) Then

condicion = String.Format("Nombre\_Programa like '%{0}%'", txtNombreBP.Text)

End If

If Not String.IsNullOrWhiteSpace(txtDescripcionBP.Text) Then

condicion += String.Format(" and Descripcion like '%{0}%'", txtDescripcionBP.Text)

End If

If Not BWBuscador.IsBusy Then

BWBuscador.RunWorkerAsync(PSQL("ID\_Programa, Nombre\_Programa as Nombre, Descripcion", "programa", condicion))

End If

End Sub

Private Sub btnBorrarBP\_Click(sender As Object, e As EventArgs) Handles btnBorrarBP.Click

BorrarConfirmar(Me, dt\_BPrograma, dgvBProgramas, PROGRAMAS, btnBuscarBP)

End Sub

Private Sub dgvBProgramas\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvBProgramas.CellDoubleClick

Dim i() As String = CargarID(dt\_BPrograma, dgvBProgramas, {0})

If (i(0) <> 0) Then

Dim formPrograma As New frmPrograma(i(0))

AddHandler formPrograma.FormClosed, AddressOf FormPrograma\_FormClosed

formPrograma.ShowDialog()

End If

End Sub

Private Sub dgvPrograma\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvPrograma.CellDoubleClick

Dim i() As String = CargarID(dt\_programa, dgvPrograma, {0})

If (i(0) <> 0) Then

Dim formPrograma As New frmPrograma(i(0))

AddHandler formPrograma.FormClosed, AddressOf ACPrograma

'TODO: Crear otro handler para actualizar la tabla de programas agendados

formPrograma.ShowDialog()

End If

End Sub

Private Sub ACPrograma()

If Not BWProgramas.IsBusy Then

BWProgramas.RunWorkerAsync()

End If

End Sub

Private Sub btnIngresarBP\_Click(sender As Object, e As EventArgs) Handles btnIngresarBP.Click

Dim formPrograma As New frmPrograma(-1)

AddHandler formPrograma.FormClosed, AddressOf FormPrograma\_FormClosed

formPrograma.ShowDialog()

End Sub

Private Sub FormPrograma\_FormClosed(sender As Object, e As FormClosedEventArgs)

btnBuscarBP.PerformClick()

If Not (BWProgramas.IsBusy) Then

BWProgramas.RunWorkerAsync()

End If

End Sub

Private Sub FormEvento\_FormClosed(sender As Object, e As FormClosedEventArgs)

btnBuscarE.PerformClick()

If Not (BWEventos.IsBusy) Then

BWEventos.RunWorkerAsync()

End If

End Sub

Private Sub btnagendarEvento\_Click(sender As Object, e As EventArgs) Handles btnagendarEvento.Click

Dim formEvento As New frmEvento(-1)

AddHandler formEvento.FormClosed, AddressOf FormEvento\_FormClosed

formEvento.ShowDialog()

End Sub

Private Sub btnLimpiarPubliB\_Click(sender As Object, e As EventArgs) Handles btnLimpiarPubliB.Click

txtNombre.Clear()

txtEmpresa.Clear()

ctpPubli.Value = Now.Date

cbPubli.Checked = False

ActualizarTablaC(Nothing, dgvClientes, False) 'fix

End Sub

Private Sub btnBuscarPubliB\_Click(sender As Object, e As EventArgs) Handles btnBuscarPubliB.Click

TBuscada = PUBLICIDAD

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If (Not String.IsNullOrWhiteSpace(txtNombre.Text)) Then

condicion = "p.nombre like '%" + txtNombre.Text + "%'"

End If

If (cbPubli.Checked) Then

condicion += String.Format(" and p.id\_publicidad in (select id\_publicidad from aparecepubli where fecha\_inicio <= '{0}' and fecha\_finalizacion >= '{0}')", Format(ctpPubli.Value, "yyyy-MM-dd").ToString)

End If

If (Not String.IsNullOrWhiteSpace(txtEmpresa.Text)) Then

condicion += " and e.nombre like '%" + txtEmpresa.Text + "%'"

End If

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("p.id\_publicidad, p.nombre as Nombre, e.nombre As Empresa", "publicidad p inner join empresa e on p.id\_empresa = e.id\_empresa", condicion))

End If

'ModLog.Guardar(PSQL("p.id\_publicidad, p.nombre as Nombre, e.nombre As Empresa", "publicidad p inner join empresa e on p.id\_empresa = e.id\_empresa", condicion))

End Sub

Private Sub cbTTodas\_CheckedChanged(sender As Object, e As EventArgs) Handles cbTTodas.CheckedChanged

If Not BWTandas.IsBusy Then

BWTandas.RunWorkerAsync(Not cbTTodas.Checked)

End If

End Sub

Private Sub dgvPubliB\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvPubliB.CellDoubleClick

Dim i As String = CargarID(dt\_BPubli, dgvPubliB)

If (i <> 0) Then

Dim formPubli As New frmPublicidad(i)

AddHandler formPubli.FormClosed, AddressOf FormPubli\_FormClosed

formPubli.ShowDialog()

End If

End Sub

Private Sub FormPubli\_FormClosed(sender As Object, e As FormClosedEventArgs)

btnBuscarPubliB.PerformClick()

If Not BWTandas.IsBusy Then

BWTandas.RunWorkerAsync(Not cbTTodas.Checked)

End If

End Sub

Private Sub btnIngresarPubliB\_Click(sender As Object, e As EventArgs) Handles btnIngresarPubliB.Click

Dim formPubli As New frmPublicidad(-1)

AddHandler formPubli.FormClosed, AddressOf FormPubli\_FormClosed

formPubli.ShowDialog()

End Sub

Private Sub btnLimpiarBF\_Click(sender As Object, e As EventArgs) Handles btnLimpiarBF.Click

txtNombreBF.Clear()

txtMailBF.Clear()

txtTelefonoBF.Clear()

ActualizarTablaC(Nothing, dgvFuncionarioBF, False) 'fix

End Sub

Private Sub btnBuscarBF\_Click(sender As Object, e As EventArgs) Handles btnBuscarBF.Click

TBuscada = ModTablas.FUNCIONARIO

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If (Not String.IsNullOrWhiteSpace(txtNombreBF.Text)) Then

condicion = "Nombre like '%" + txtNombreBF.Text + "%'"

End If

If (Not String.IsNullOrWhiteSpace(txtTelefonoBF.Text)) Then

condicion += " and Telefono = '" + txtTelefonoBF.Text + "'"

End If

If (Not String.IsNullOrWhiteSpace(txtMailBF.Text)) Then

condicion += " and Mail = '" + txtMailBF.Text + "'"

End If

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("id\_Funcionario, Nombre, Telefono, Mail", "funcionario", condicion))

End If

End Sub

Private Sub btnBorrarPubliB\_Click(sender As Object, e As EventArgs) Handles btnBorrarPubliB.Click

BorrarConfirmar(Me, dt\_BPubli, dgvPubliB, PUBLICIDAD, btnBuscarPubliB)

End Sub

Private Sub btnBorrarBF\_Click(sender As Object, e As EventArgs) Handles btnBorrarBF.Click

BorrarConfirmar(Me, dt\_BFuncionario, dgvFuncionarioBF, ModTablas.FUNCIONARIO, btnBuscarBF)

End Sub

Private Sub btnIngresarBF\_Click(sender As Object, e As EventArgs) Handles btnIngresarBF.Click

Dim formFuncionario As New frmFuncionario({-1, "", "", ""})

AddHandler formFuncionario.FormClosed, AddressOf FormFuncrio\_FormClosed

formFuncionario.ShowDialog()

End Sub

Private Sub btnTanda\_Click(sender As Object, e As EventArgs) Handles btnTanda.Click

Dim formTanda As New frmTandas()

AddHandler formTanda.FormClosed, AddressOf tandasClosed

formTanda.ShowDialog()

End Sub

Private Sub tandasClosed()

If Not BWProgramas.IsBusy Then

BWTandas.RunWorkerAsync(Not cbTTodas.Checked)

End If

End Sub

Private Sub dgvFuncionarioBF\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFuncionarioBF.CellDoubleClick

Dim i() As String = CargarID(dt\_BFuncionario, dgvFuncionarioBF, {0, 1, 2, 3})

If (i(0) > 0) Then

Dim formFuncrio As New frmFuncionario(i)

AddHandler formFuncrio.FormClosed, AddressOf FormFuncrio\_FormClosed

formFuncrio.ShowDialog()

End If

End Sub

Private Sub FormFuncrio\_FormClosed(sender As Object, e As FormClosedEventArgs)

btnBuscarBF.PerformClick()

End Sub

Private Sub btnBuscarBFF\_Click(sender As Object, e As EventArgs) Handles btnBuscarBFF.Click

TBuscada = FUNCION

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If (Not String.IsNullOrWhiteSpace(txtNombreBFF.Text)) Then

condicion = String.Format("Nombre like '%{0}%'", txtNombreBFF.Text)

End If

If (Not String.IsNullOrWhiteSpace(txtDescripcionBFF.Text)) Then

condicion += String.Format(" and Descripcion like '%{0}%'", txtDescripcionBFF.Text)

End If

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("ID\_Funcion, Nombre, Descripcion", "Funcion", condicion))

End If

End Sub

Private Sub btnBorrarBFF\_Click(sender As Object, e As EventArgs) Handles btnBorrarBFF.Click

BorrarConfirmar(Me, dt\_BFuncion, dgvFuncionesBFF, FUNCION, btnBuscarBFF)

End Sub

Private Sub btnIngresarBFF\_Click(sender As Object, e As EventArgs) Handles btnIngresarBFF.Click

Dim formFuncion As New frmFuncion({-1, "", "", ""})

AddHandler formFuncion.FormClosed, AddressOf FormFunc\_FormClosed

formFuncion.ShowDialog()

End Sub

Private Sub btnLimpiarBFF\_Click(sender As Object, e As EventArgs) Handles btnLimpiarBFF.Click

txtNombreBFF.Clear()

txtDescripcionBFF.Clear()

ActualizarTablaC(Nothing, dgvFuncionesBFF, False) 'fix

End Sub

Private Sub dgvFuncionesBFF\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFuncionesBFF.CellDoubleClick

Dim i() As String = CargarID(dt\_BFuncion, dgvFuncionesBFF, {0, 1, 2})

If (i(0) > 0) Then

Dim formFunc As New frmFuncion(i)

AddHandler formFunc.FormClosed, AddressOf FormFunc\_FormClosed

formFunc.ShowDialog()

End If

End Sub

Private Sub FormFunc\_FormClosed(sender As Object, e As FormClosedEventArgs)

btnBuscarBFF.PerformClick()

End Sub

Private Sub btnBuscarE\_Click(sender As Object, e As EventArgs) Handles btnBuscarE.Click

TBuscada = EVENTO

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If Not String.IsNullOrWhiteSpace(txtNombreBP.Text) Then

condicion = String.Format("nombre like '%{0}%'", txtNombreBP.Text)

End If

If Not String.IsNullOrWhiteSpace(txtDescripcionBP.Text) Then

condicion += String.Format(" and descripcion like '%{0}%'", txtDescripcionBP.Text)

End If

If Not BWBuscador.IsBusy Then

BWBuscador.RunWorkerAsync(PSQL("ID\_Evento, nombre as Nombre, descripcion as Descripcion", "evento", condicion))

End If

End Sub

Private Sub btnBEvento\_Click(sender As Object, e As EventArgs) Handles btnBEvento.Click

BorrarConfirmar(Me, dt\_BEvento, dgvBEvento, EVENTO, btnBEvento)

End Sub

Private Sub btnIngresarE\_Click(sender As Object, e As EventArgs) Handles btnIngresarE.Click

Dim formEvento As New frmEvento(-1)

AddHandler formEvento.FormClosed, AddressOf FormEvento\_FormClosed

formEvento.ShowDialog()

End Sub

Private Sub dgvBEvento\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvBEvento.CellDoubleClick

Dim i As Integer = CargarID(dt\_BEvento, dgvBEvento)

If (i <> -1 And i <> 0) Then

Dim formEvento As New frmEvento(i)

AddHandler formEvento.FormClosed, AddressOf FormEvento\_FormClosed

formEvento.ShowDialog()

End If

End Sub

Private Sub btnLimpiarEvento\_Click(sender As Object, e As EventArgs) Handles btnLimpiarEvento.Click

txtNombreE.Clear()

txtDescripcionE.Clear()

ActualizarTablaC(Nothing, dgvBEvento, False) 'fix

End Sub

Private Sub dgvEventos\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvEventos.CellDoubleClick

Dim i As Integer = CargarID(dt\_evento, dgvEventos)

If (i <> -1 And i <> 0) Then

Dim formEvento As New frmEvento(i)

AddHandler formEvento.FormClosed, AddressOf FormEvento\_FormClosed

formEvento.ShowDialog()

End If

End Sub

Private Sub dgvPublicidades\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvPublicidades.CellDoubleClick

Dim i As Integer = CargarID(dt\_publi, dgvPublicidades)

If (i <> -1 And i <> 0) And PoseePermiso("Publicidad") Then

Dim formPubli As New frmPublicidad(i)

AddHandler formPubli.FormClosed, AddressOf FormPubli\_Close

formPubli.ShowDialog()

End If

End Sub

Private Sub dgvPPublicidades\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvPPublicidades.CellDoubleClick

Dim i As Integer = CargarID(dt\_Ppubli, dgvPPublicidades)

If (i <> -1 And i <> 0) Then

Dim formPubli As New frmPublicidad(i)

AddHandler formPubli.FormClosed, AddressOf FormPPubli\_FormClosed

formPubli.ShowDialog()

End If

End Sub

Private Sub FormPPubli\_FormClosed()

If Not (BWDPRogramas.IsBusy) Then

BWDPRogramas.RunWorkerAsync()

End If

End Sub

Private Sub FormPubli\_Close()

If Not (BWPublicidades.IsBusy) Then

BWPublicidades.RunWorkerAsync()

End If

End Sub

Private Sub dgvFuncionarios\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvFuncionarios.CellDoubleClick

Dim i() As String = CargarID(dt\_dprograma, dgvFuncionarios, {0, 1, 2, 3})

If (i(0) > 0) Then

Dim formFun As New frmFuncionario(i)

AddHandler formFun.FormClosed, AddressOf FormPPubli\_FormClosed

formFun.ShowDialog()

End If

End Sub

Private Sub RecTan\_Tick(sender As Object, e As EventArgs) Handles RecTan.Tick

If Not BWTandas.IsBusy Then

BWTandas.RunWorkerAsync(Not cbTTodas.Checked)

End If

End Sub

Private Sub RecPA\_Tick(sender As Object, e As EventArgs) Handles RecPA.Tick

If Not BWProgramas.IsBusy Then

BWProgramas.RunWorkerAsync()

End If

End Sub

Private Sub tProgramas\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles tProgramas.SelectedIndexChanged

If (tProgramas.SelectedIndex = 0) Then

If Not BWProgramas.IsBusy Then

BWProgramas.RunWorkerAsync()

End If

End If

End Sub

Private Sub tcSecciones\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles tcSecciones.SelectedIndexChanged

Select Case tcSecciones.SelectedIndex

Case 0

If Not BWProgramas.IsBusy Then

BWProgramas.RunWorkerAsync()

End If

End Select

End Sub

Private Sub dgvPrograma\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvPrograma.CellClick

If Not BWDPRogramas.IsBusy Then

BWDPRogramas.RunWorkerAsync()

End If

End Sub

End Class

frmAcceso.vb

Public Class frmAcceso

Private UID As Integer = 0

Public Sub New(Optional ByVal Uid As Integer = 1)

InitializeComponent()

Me.UID = Uid

End Sub

Private Sub Permisos\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

CargarPermisosAll(UID)

CheckearPermisos()

EstablecerList(cblP)

End Sub

Private Sub btnSalir\_Click(sender As Object, e As EventArgs) Handles btnSalir.Click

Close()

End Sub

Private Sub btnActualizar\_Click(sender As Object, e As EventArgs) Handles btnActualizar.Click

CargaActualizacionP(cblP)

CompararPermisos(cblP)

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

EstablecerAll(cblP)

End Sub

End Class

frmConfiguracion.vb

Public Class frmConfiguracion

Private UserID As Integer = Nothing

Public dt As DataTable

Private Sub btnRestablecer\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnRestablecer.Click

ModUser.Borrar()

ModUser.Inicio()

End Sub

Private Sub btnGuardar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGuardar.Click

If (txtIp.Text <> "") Then

ModConector.EAddress(txtIp.Text)

End If

If (txtPuerto.Text <> "") Then

ModConector.EPort(txtPuerto.Text)

End If

If (txtUsuario.Text <> "") Then

ModConector.EUser(txtUsuario.Text)

End If

ModConector.EPass(txtContraseña.Text)

If (txtBD.Text <> "") Then

ModConector.EDatabase(txtBD.Text)

End If

ModUser.Guardar(False)

ModUser.LeeDatos()

End Sub

Private Sub DebugCrear\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles DebugCrear.Click

If (txtIp.Text <> "") Then

ModConector.EAddress(txtIp.Text)

End If

If (txtPuerto.Text <> "") Then

ModConector.EPort(txtPuerto.Text)

End If

If (txtUsuario.Text <> "") Then

ModConector.EUser(txtUsuario.Text)

End If

ModConector.EPass(txtContraseña.Text)

If (txtBD.Text <> "") Then

ModConector.EDatabase(txtBD.Text)

End If

ModUser.Guardar(True)

ModUser.LeeDatos()

Me.Dispose()

End Sub

Private Sub DebugCrearUsuario\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

ModConector.IUsuario(txtUsuario.Text, txtContraseña.Text)

End Sub

Private Sub frmConfiguracion\_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing

ModUser.Inicio()

ModConector.Inicio()

End Sub

Private Sub frmConfiguracion\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

If ModConector.GDebug Then

DebugCrear.Visible = True

btnGenerador.Visible = True

btnGenerador.Enabled = True

DebugCrear.Enabled = True

End If

If Not PoseePermiso("Configuracion", "a") And Not ModConector.GDebug Then

TAB.TabPages.RemoveAt(1)

End If

Control.CheckForIllegalCrossThreadCalls = False

End Sub

Private Sub TabPage2\_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles TPusuarios.Enter

'ActualizarUsuarios(False)

BW.RunWorkerAsync(False)

LimpiarEditar()

limpiar()

End Sub

#Region "Limpiadores"

Private Sub limpiar()

txtNombre.Text = ""

txtContrasena.Text = ""

dgvNombreUsuario.ClearSelection()

End Sub

Private Sub LimpiarUsuario\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LimpiarUsuario.Click

limpiar()

End Sub

Private Sub LimpiarEditar()

txtENombre.Text = ""

txtEContrasena.Text = ""

UserID = Nothing

End Sub

Private Sub UAplicar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles UAplicar.Click

If Not IsNothing(UserID) Then

USQL("usuarios", "nombre ='" + txtENombre.Text + "',contrasena = AES\_ENCRYPT('" + txtEContrasena.Text() + "',sha2('" + ModCodificador.GKey + "',256))", "id\_usuario ='" + UserID.ToString() + "'")

'ActualizarUsuarios(False)

BW.RunWorkerAsync(False)

If UserID = ModConector.GUsuarioID Then

ModConector.BorrarUsuario()

Me.Dispose()

End If

End If

LimpiarEditar()

End Sub

#End Region

#Region "Modificar Usuarios"

Private Sub UBorrar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles UBorrar.Click

If Not IsNothing(UserID) Then

BSQL("usuarios", "id\_usuario ='" + UserID.ToString + "'")

'ActualizarUsuarios(False)

BW.RunWorkerAsync(False)

If UserID = ModConector.GUsuarioID Then

ModConector.BorrarUsuario()

Me.Dispose()

End If

End If

LimpiarEditar()

End Sub

Private Sub ActualizarUsuarios()

If Not IsNothing(dt) Then

Me.dgvNombreUsuario.DataSource = dt

Else

MessageBox.Show("No se cargo correctamente.")

End If

End Sub

Private Sub CrearUsuario\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CrearUsuario.Click

ModConector.IUsuario(txtNombre.Text, txtContrasena.Text)

'ActualizarUsuarios(True)

BW.RunWorkerAsync(True)

limpiar()

End Sub

#End Region

Private Sub dgvNombreUsuario\_CellMouseClick(ByVal sender As Object, ByVal e As System.Windows.Forms.DataGridViewCellMouseEventArgs) Handles dgvNombreUsuario.CellMouseClick

Try

LimpiarEditar()

txtENombre.Text = dgvNombreUsuario.CurrentRow.Cells("Nombre Usuarios").Value.ToString

UserID = dgvNombreUsuario.CurrentRow.Cells("ID").Value

Catch es As Exception

End Try

End Sub

Private Sub BW\_DoWork(ByVal sender As Object, ByVal e As System.ComponentModel.DoWorkEventArgs) Handles BW.DoWork

dt = ModConector.AUsuarios(e.Argument)

End Sub

Private Sub BW\_RunWorkerCompleted(ByVal sender As Object, ByVal e As System.ComponentModel.RunWorkerCompletedEventArgs) Handles BW.RunWorkerCompleted

ActualizarUsuarios()

End Sub

Private Sub btnGenerador\_Click(sender As Object, e As EventArgs) Handles btnGenerador.Click

ModInicializador.Generadores()

End Sub

Private Sub btnAccesos\_Click(sender As Object, e As EventArgs) Handles btnAccesos.Click

If Not IsNothing(UserID) Then

Dim formAccesos As New frmAcceso(UserID)

formAccesos.ShowDialog()

End If

End Sub

End Class

frmLUsuario.vb

Public Class frmLUsuario

Private Sub btnOpciones\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOpciones.Click

If (BUsuario(txtUsuario.Text, txtPass.Text) And PoseePermiso("Configuracion")) Or ModConector.GDebug Then

ModInicializador.Configuracion()

End If

End Sub

Private Sub btnSalir\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

ModConector.desconectar()

End

End Sub

Private Sub btnEntrar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEntrar.Click

If ModConector.BUsuario(txtUsuario.Text, txtPass.Text) Then

If PoseePermiso("Inicio") Then

ModInicializador.Principal()

Me.Close()

Else

MessageBox.Show("No posee permisos para ingresar")

End If

End If

End Sub

Private Sub frmLUsuario\_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing

ModConector.desconectar()

End Sub

Private Sub frmLUsuario\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Try

ModUser.Inicio()

ModConector.Inicio()

Catch m As Exception

If (ModConector.GDebug) Then

MessageBox.Show(e.ToString)

End If

End Try

End Sub

End Class

frmVideo.vb

Public Class frmVideo

Dim videoID As Integer

Private editando As Boolean = False

Dim datos() As String

Dim datosI() As String

Dim dtV As DataTable

Dim position() As String

Dim pos As UInt16 = 0

Dim serieID As String = ""

Public Sub New(ByVal vid As Integer)

InitializeComponent()

videoID = vid

End Sub

Public Sub Buscar()

Dim columnas() As String = {"contenido", "Nombre", "id\_serie", "DATE\_FORMAT(Fecha,'%Y-%m-%d') as Fecha"}

datosI = BuscarDatos("video", columnas, "id\_video", videoID)

Rellenar()

End Sub

Private Sub Rellenar()

txtNombre.Text = datosI(1)

If datosI(3) <> "" Then

dtpFecha.Value = CDate(datosI(3))

Else

txtTapar.Visible = True

End If

txtContenido.Text = datosI(0)

If (datosI(2) = "") Then

serieID = 0

Else

serieID = datosI(2)

If (datosI(2) = 0) Then

datosI(2) = ""

End If

End If

CargarCombo()

End Sub

Sub CargarCombo()

dtV = DevolverTabla(PSQL("id\_serie, nombre", "Serie", "True"))

LlenarCombo(cbSerie, dtV, "nombre")

If Not IsNothing(dtV) Then

ExtraerDatos()

End If

cbSerie.SelectedIndex = pos

End Sub

Public Sub ExtraerDatos()

ReDim position(dtV.Rows.Count - 1)

For j As Integer = 0 To dtV.Rows.Count - 1

position(j) = dtV.Rows(j).Item(0).ToString

Next

For i As Integer = 0 To position.Length - 1

If serieID = position(i) Then

pos = i + 1

Exit For

End If

Next

End Sub

Private Sub btnEditar\_Click(sender As Object, e As EventArgs) Handles btnEditar.Click

If videoID = -1 Then

ActualizarDatos()

PrepararInsert("video", datos)

Vaciar()

ElseIf editando Then

ActualizarDatos()

If Not CompararValores(VaciarNull(datos), datosI) Then

PrepararUpdate("video", datos, videoID)

datosI = VaciarNull(datos)

End If

Alternar()

Else

Alternar()

End If

End Sub

Sub Activar()

txtNombre.ReadOnly = editando

txtContenido.ReadOnly = editando

editando = Not editando

cbSerie.Enabled = editando

dtpFecha.Enabled = editando

chbTieneFecha.Visible = editando

End Sub

Private Sub Alternar()

Activar()

btnEditar.Text = If(editando, "Guardar", "Editar")

btnSalir.Text = If(editando, "Cancelar", "Salir")

chbTieneFecha.Checked = datosI(3) <> ""

If editando Then

txtTapar.Visible = False

Else

If datosI(3) = "" Then

txtTapar.Visible = True

chbTieneFecha.Visible = False

End If

End If

End Sub

Private Sub btnSalir\_Click(sender As Object, e As EventArgs) Handles btnSalir.Click

If Not editando Or videoID = -1 Then

Close()

Else

Rellenar()

Alternar()

End If

End Sub

Private Sub frmVideo\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

If Not PoseePermiso("Video", "a") Then

btnBorrar.Visible = False

btnEditar.Visible = False

End If

If videoID <> -1 Then

Buscar()

btnSalir.Select()

Else

btnEditar.Text = "Insertar"

btnBorrar.Visible = False

Activar()

CargarCombo()

End If

dtpFecha.BackColor = Color.FromArgb(64, 64, 64)

dtpFecha.ForeColor = Color.White

btnSalir.Select()

End Sub

Sub Vaciar()

txtNombre.Clear()

txtContenido.Clear()

cbSerie.SelectedIndex = -1

dtpFecha.Value = Now.Date

chbTieneFecha.Checked = False

End Sub

Private Sub frmVideo\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

If editando And videoID <> -1 Then

ActualizarDatos()

If Not CompararValores(VaciarNull(datos), datosI) Then

Dim g As New frmGuardarEdicion("Video", datos, videoID)

g.ShowDialog()

If ModInicializador.Cancelar.Contains("Video") Then

e.Cancel = True

ModInicializador.Cancelar = ModInicializador.Cancelar.Replace("Video", "")

' ModInicializador.frmPrin.btnbuscarv.PerformClick()

End If

End If

End If

End Sub

Private Sub ActualizarDatos()

Dim con As String = txtContenido.Text

Dim nom As String = txtNombre.Text

Dim dat As String = If(chbTieneFecha.Checked, Format(dtpFecha.Value, "yyyy-MM-dd"), "null")

Dim ser As String = If(cbSerie.SelectedIndex <= 0, "null", position(cbSerie.SelectedIndex - 1))

datos = {con, nom, ser, dat}

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

Dim formDelete As New frmConfirmarBorrado(VIDEO, {videoID}, True)

formDelete.ShowDialog(Me)

End Sub

End Class

frmSerie.vb

' TODO: Indicador de editar e ingresado correctamente.

Imports System.ComponentModel

Public Class frmSerie

Dim serieID As Integer

Dim editando As Boolean = False ' Controla si se esta en modo de edicion o no

Dim tmpDatos(1) As String

Dim cambio As Boolean = False ' Controla si han habido cambios desde el ultimo modo de edicion

Dim dt\_Video As New DataTable

Dim TBuscada As Byte = 0

Dim dt\_VideoBV As DataTable

Dim TBusca As DataTable

Public Sub New(ByVal DatosI() As String)

InitializeComponent()

'Los siguientes datos se obtienen de la tabla en el elemento padre

serieID = DatosI(0)

txtNombre.Text = DatosI(2)

If DatosI(1) <> "" Then

RemoveHandler dtpFecha.ValueChanged, AddressOf dtpFecha\_ValueChanged

dtpFecha.Value = CDate(DatosI(1))

AddHandler dtpFecha.ValueChanged, AddressOf dtpFecha\_ValueChanged

' Fix temporal... no me maten

Else

txtTapar.Visible = True

End If

RemoveHandler chbIncluir.CheckedChanged, AddressOf chbIncluir\_CheckedChanged

chbIncluir.Checked = DatosI(1) <> ""

AddHandler chbIncluir.CheckedChanged, AddressOf chbIncluir\_CheckedChanged

' ...Esto tambien, supongo.....

''ModLog.Guardar(PSQL("id\_video, fecha as Fecha, nombre as Nombre", "video", String.Format("id\_serie = '{0}'", DatosI(0))))

End Sub

Private Sub frmSerie\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

dtpFecha.BackColor = Color.FromArgb(64, 64, 64)

dtpFecha.ForeColor = Color.White

ActualizarTabla()

If (serieID = -1) Then

ocultar()

Alternar()

End If

If Not PoseePermiso("Serie", "a") Then

btnBorrar.Visible = False

btnSEditar.Visible = False

ocultar()

End If

If Not PoseePermiso("Video", "a") Then

ocultar()

End If

End Sub

Private Sub ocultar()

tcS.TabPages.RemoveByKey("tbAV")

End Sub

Private Sub btnSEditar\_Click(sender As Object, e As EventArgs) Handles btnSEditar.Click

'' editando = True -> Se guardaran los cambios

'' editando = False -> Se le permitira al usuario escribir en los campos

If serieID = -1 Then

Dim datos() As String = {If(chbIncluir.Checked, Format(dtpFecha.Value, "yyyy-MM-dd"), "null"), txtNombre.Text}

PrepararInsert("Serie", datos)

Vaciar()

ElseIf editando Then

If cambio Then

Dim datos() As String = {If(chbIncluir.Checked, Format(dtpFecha.Value, "yyyy-MM-dd"), "null"), txtNombre.Text}

If Not CompararValores(VaciarNull(datos), tmpDatos) Then

PrepararUpdate("Serie", datos, serieID)

End If

AlternarCambioHandlers()

End If

Alternar()

Else

tmpDatos = {If(chbIncluir.Checked, Format(dtpFecha.Value, "yyyy-MM-dd"), ""), txtNombre.Text}

Alternar()

End If

End Sub

Private Sub Vaciar()

txtNombre.Clear()

dtpFecha.Value = Now.Date

chbIncluir.Checked = True

End Sub

Private Sub btnSSalir\_Click(sender As Object, e As EventArgs) Handles btnSSalir.Click

If Not editando Or serieID = -1 Then

Close()

Else

If cambio Then

If tmpDatos(0) <> "" Then

dtpFecha.Value = CDate(tmpDatos(0))

Else

dtpFecha.Value = Now.Date

chbIncluir.Checked = False

End If

txtNombre.Text = tmpDatos(1)

AlternarCambioHandlers()

End If

Alternar()

End If

End Sub

Private Sub Serie\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

If cambio And serieID <> -1 Then

Dim datos() As String = {If(chbIncluir.Checked, Format(dtpFecha.Value, "yyyy-MM-dd"), "null"), txtNombre.Text}

If Not CompararValores(VaciarNull(datos), tmpDatos) Then

Dim g As New frmGuardarEdicion("Serie", datos, serieID)

g.ShowDialog()

If ModInicializador.Cancelar.Contains("Serie") Then

e.Cancel = True

ModInicializador.Cancelar = ModInicializador.Cancelar.Replace("Serie", "")

Else

AlternarCambioHandlers()

End If

End If

End If

End Sub

'' Alternar botones

Private Sub Alternar()

If serieID = -1 Then

btnSEditar.Text = "Ingresar"

btnBorrar.Visible = False

btnSSalir.Text = "Salir"

Text = "Ingresar Serie"

txtTapar.Visible = False

ElseIf editando Then

btnSEditar.Text = "Editar"

btnSSalir.Text = "Salir"

Text = "Ver Serie"

txtTapar.Visible = Not chbIncluir.Checked

Else

btnSSalir.Text = "Cancelar"

btnSEditar.Text = "Guardar"

Text = "Editar Serie"

RemoveHandler chbIncluir.CheckedChanged, AddressOf chbIncluir\_CheckedChanged

chbIncluir.Checked = tmpDatos(0) <> ""

AddHandler chbIncluir.CheckedChanged, AddressOf chbIncluir\_CheckedChanged

' ...Sufro con poner esas dos lineas

txtTapar.Visible = False

End If

editando = Not editando

txtNombre.ReadOnly = Not editando

dtpFecha.Enabled = editando

chbIncluir.Enabled = editando

End Sub

'' Checkean si hay cambios hechos

'' Si cambio = True, no se llamaran

Private Sub txtNombre\_ModifiedChanged(sender As Object, e As EventArgs)

If txtNombre.Modified Then

AlternarCambioHandlers()

End If

End Sub

Private Sub dtpFecha\_ValueChanged(sender As Object, e As EventArgs)

AlternarCambioHandlers()

End Sub

Private Sub chbIncluir\_CheckedChanged(sender As Object, e As EventArgs)

AlternarCambioHandlers()

End Sub

' Aqui yacIA mi variable, recordatorio de lo que una vez fue Y AUN ES

' cambio: I LIVED BITCH

'' Alterna el estado de cambios y activa/desactiva la deteccion de cambios

Private Sub AlternarCambioHandlers()

'' cambio = True -> Se aniadiran los handlers, se los necesita

'' cambio = False -> Se ha detectado un cambio, y los handlers se removeran hasta la siguiente llamada del metodo

If cambio Then

AddHandler dtpFecha.ValueChanged, AddressOf dtpFecha\_ValueChanged

AddHandler txtNombre.ModifiedChanged, AddressOf txtNombre\_ModifiedChanged

AddHandler chbIncluir.CheckedChanged, AddressOf chbIncluir\_CheckedChanged

Else

RemoveHandler dtpFecha.ValueChanged, AddressOf dtpFecha\_ValueChanged

RemoveHandler txtNombre.ModifiedChanged, AddressOf txtNombre\_ModifiedChanged

RemoveHandler chbIncluir.CheckedChanged, AddressOf chbIncluir\_CheckedChanged

End If

cambio = Not cambio

End Sub

Private Sub dgvVSM\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvVSM.CellDoubleClick, dgvVideoAs.CellDoubleClick

Dim i As Integer = CargarID(dt\_Video, sender)

If (i <> -1) Then

Dim formVideo As New frmVideo(i)

AddHandler formVideo.FormClosed, AddressOf FormVideo\_FormClosed

formVideo.ShowDialog()

End If

End Sub

Private Sub dgvVSB\_CellDoubleClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvVideoBV.CellDoubleClick

Dim i As Integer = CargarID(dt\_VideoBV, sender)

If (i <> -1) Then

Dim formVideo As New frmVideo(i)

AddHandler formVideo.FormClosed, AddressOf FormVideo\_FormClosed

formVideo.ShowDialog()

End If

End Sub

'' Actualiza la tabla mostrando los videos asociados cuando el formulario de mostrar video se ha cerrado

Private Sub FormVideo\_FormClosed(sender As Object, e As FormClosedEventArgs)

ActualizarTabla()

End Sub

Private Sub ActualizarTabla()

dt\_Video = DevolverTabla(PSQL("id\_video, fecha as Fecha, nombre as Nombre", "video", String.Format("id\_serie = '{0}'", serieID)))

ActualizarTablaC(dt\_Video, dgvVSM)

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

Dim formDelete As New frmConfirmarBorrado(SERIE, {serieID}, True)

formDelete.ShowDialog(Me)

End Sub

Private Sub btnBuscar\_Click(sender As Object, e As EventArgs) Handles btnBuscar.Click

Buscar()

End Sub

Private Sub Buscar()

TBuscada = VIDEO

Dim condicion As String = "true" ' FIXME: Al poner limit 50 no sirve buscar solo por fecha. Asi que lo he quitado por ahora.

If (Not String.IsNullOrWhiteSpace(txtNombreBV.Text)) Then

condicion = String.Format("nombre like '%{0}%'", txtNombreBV.Text)

End If

If (chkFecha.Checked) Then

condicion += String.Format(" and fecha = '{0}'", Format(dtpFechaBV.Value, "yyyy-MM-dd").ToString)

End If

condicion += " and ID\_Serie is null"

If Not (BWBuscador.IsBusy) Then

BWBuscador.RunWorkerAsync(PSQL("id\_video, fecha as Fecha, nombre as Nombre", "video", condicion))

End If

End Sub

Private Sub BWBuscador\_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles BWBuscador.RunWorkerCompleted

Select Case TBuscada

Case VIDEO

dt\_VideoBV = TBusca

ActualizarTablaC(dt\_VideoBV, dgvVideoBV)

TBuscada = SERIE

BWBuscador.RunWorkerAsync(PSQL("id\_video, fecha as Fecha, nombre as Nombre", "video", String.Format("id\_serie = '{0}'", serieID)))

Case SERIE

dt\_Video = TBusca

ActualizarTablaC(dt\_Video, dgvVideoAs)

TBusca = Nothing

TBuscada = 0

End Select

End Sub

Private Sub BWBuscador\_DoWork(sender As Object, e As DoWorkEventArgs) Handles BWBuscador.DoWork

TBusca = DevolverTabla(e.Argument)

ModLog.Guardar("Serie: " & e.Argument)

End Sub

Private Sub tcS\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles tcS.SelectedIndexChanged

If tcS.SelectedIndex = 0 Then

ActualizarTabla()

Else tcS.SelectedIndex = 1

Buscar()

End If

End Sub

Private Sub dgv\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvVideoBV.CellClick, dgvVideoAs.CellClick

ClickCheck(sender, e.ColumnIndex)

End Sub

Private Sub dgvHeaderClick(sender As Object, e As DataGridViewCellMouseEventArgs) Handles dgvVideoBV.ColumnHeaderMouseClick, dgvVideoAs.ColumnHeaderMouseClick

CheckAll(sender, e.ColumnIndex)

End Sub

Private Sub btnAsignar\_Click(sender As Object, e As EventArgs) Handles btnAsignar.Click

Dim Checked() As String = ObtenerCheck(dt\_VideoBV, dgvVideoBV)

If Checked.Length > 0 Then

PrepararUpdate("Video", {"ID\_Serie"}, {serieID.ToString}, "ID\_Video", Checked)

Buscar()

End If

End Sub

Private Sub btnDesasignar\_Click(sender As Object, e As EventArgs) Handles btnDesasignar.Click

Dim Checked() As String = ObtenerCheck(dt\_Video, dgvVideoAs)

If Checked.Length > 0 Then

PrepararUpdate("Video", {"ID\_Serie"}, {"null"}, "ID\_Video", Checked)

Buscar()

End If

End Sub

End Class

frmTandas.vb

Public Class frmTandas

Private hora1 As DateTime

Private hora2 As DateTime

Private dt\_MTandas As DataTable

Private Sub btnsalirt\_Click(sender As Object, e As EventArgs) Handles btnsalirt.Click

Close()

End Sub

Private Sub BuscarT()

dt\_MTandas = ATandas(False)

ActualizarTablaC(dt\_MTandas, dgvTandas, False)

End Sub

Private Sub FechasMax(Optional ingresar As Boolean = True)

hora1 = Convert.ToDateTime(Dia(Now.Date) + " " + MysqlHM(dtpHI.Value))

hora2 = Convert.ToDateTime(Dia(Now.Date) + " " + MysqlHM(dtpHF.Value))

If (dgvTandas.Rows.Count > 0) Then

For i As Integer = 0 To dgvTandas.Rows.Count - 1

Dim horaN1 As DateTime = Convert.ToDateTime(Dia(Now.Date) + " " + dgvTandas.Rows(i).Cells(0).Value().ToString)

Dim horaN2 As DateTime = Convert.ToDateTime(Dia(Now.Date) + " " + dgvTandas.Rows(i).Cells(1).Value().ToString)

If (horaN1 < hora1 And horaN2 > hora1) Then

hora1 = horaN1

End If

If (horaN2 > hora2 And horaN1 < hora2) Then

hora2 = horaN2

End If

Next

BSQL("tanda", String.Format("Hora\_Inicio>='{0}' and Hora\_fin<='{1}'", MysqlHM(hora1), MysqlHM(hora2)))

End If

If (ingresar) Then

ISQL("tanda", "hora\_inicio, hora\_fin", String.Format("'{0}','{1}'", MysqlHM(hora1), MysqlHM(hora2)))

USQL("aparecepubli", String.Format("hora\_inicio='{0}'", MysqlHM(hora1)), String.Format("Hora\_Inicio='{0}'", MysqlHM(hora1)))

End If

BuscarT()

End Sub

Private Sub frmTandas\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

BuscarT()

HORAIF(dtpHI)

HORAIF(dtpHF)

If Not PoseePermiso("Tanda", "a") Then

btnBorrar.Visible = False

btnIngresar.Visible = False

End If

End Sub

Private Sub btnIngresar\_Click(sender As Object, e As EventArgs) Handles btnIngresar.Click

FechasMax()

End Sub

Private Sub btnBorrar\_Click(sender As Object, e As EventArgs) Handles btnBorrar.Click

If Not IsNothing(dt\_MTandas) Then

If (dt\_MTandas.Rows.Count > 0) Then

Dim Id() As String = ObtenerCheck(dt\_MTandas, dgvTandas, 0)

If Not Id.Length = 0 Then

Dim formDelete As New frmConfirmarBorrado(TANDASHORAS, Id, False)

formDelete.ShowDialog(Me)

BuscarT()

End If

End If

End If

End Sub

Private Sub dgvTandas\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvTandas.CellClick

ClickCheck(sender, e.ColumnIndex)

End Sub

Private Sub dgvHeaderClick(sender As Object, e As DataGridViewCellMouseEventArgs) Handles dgvTandas.ColumnHeaderMouseClick

CheckAll(sender, e.ColumnIndex)

End Sub

Private Sub dtpHI\_ValueChanged(sender As Object, e As EventArgs) Handles dtpHI.ValueChanged

If (dtpHF.Value < dtpHI.Value) Then

dtpHF.Value = dtpHI.Value

End If

dtpHF.MinDate = dtpHI.Value

End Sub

End Class

ColorDateTimePicker.vb

Imports System.Windows.Forms.VisualStyles

Public Class ColorDateTimePicker

Inherits DateTimePicker

Public Sub New()

MyBase.New()

SetStyle(ControlStyles.UserPaint, True)

ForeColor = Color.Black

BackColor = Color.White

DropDownAlign = LeftRightAlignment.Right

End Sub

Public Overrides Property ForeColor As Color

Public Overrides Property BackColor As Color

Protected Overrides Sub OnPaint(e As PaintEventArgs)

Dim g As Graphics = Me.CreateGraphics()

Dim dropDownRectangle As Rectangle = New Rectangle(ClientRectangle.Width - 17, 0, 17, ClientRectangle.Height)

Dim bkgBrush As Brush = New SolidBrush(BackColor)

Dim visualState As ComboBoxState = ComboBoxState.Normal

g.FillRectangle(bkgBrush, 0, 0, ClientRectangle.Width, ClientRectangle.Height)

Dim textBrush As Brush = New SolidBrush(ForeColor)

g.DrawString(Text, Font, textBrush, 0, 2)

ComboBoxRenderer.DrawDropDownButton(g, dropDownRectangle, visualState)

g.DrawRectangle(Pens.Gray, 0, 0, ClientRectangle.Width - 1, ClientRectangle.Height - 1)

g.Dispose()

bkgBrush.Dispose()

textBrush.Dispose()

End Sub

End Class

# Manual de usuario

# Presentación del manual:

* Justificación del manual bilingüe.

Un manual bilingüe tiene ventajas ya que normalmente se lo crea en el idioma que el producto fue creado y en un idioma universal (ingles) para que todos los usuarios estén al mismo nivel de compresión del funcionamiento de dicho producto.

El usuario que tenga la posibilidad de acceder a un manual bilingüe ayuda a la compresión del programa sin que este sepa el idioma que originalmente el manual fue redactado. Además de ayudar a la compresión de cada función que el programa brinda en su idioma, sin tener la necesidad de estar traduciendo. También el manual bilingüe ayuda a una mejor comunicación y comprensión de dos usuarios que no tengan el mismo idioma y estén interesados en el programa.

* Formas de crear un manual bilingüe accesible.

Para la creación del manual bilingüe hay que realizar ciertos pasos ayudándonos con algunas herramientas que nos permitirán una mejor accesibilidad para el usuario.

Primero hay que identificar a que tipos de usuarios irá dirigido el manual, en el caso de ser un manual bilingüe hay que tener en cuenta los dos idiomas en los cuales será redactado, en nuestro caso español e inglés.

Al momento de la identificación de los usuarios que tendrán conocimientos sobre el manual, hay que tener en cuenta la capacitación que estos tienen, normalmente es mejor crear el manual a un nivel de usuario estándar, ya que puede haber un usuario que no tenga una gran capacitación.

El formato y el estilo del manual debe ser apropiado, debe tener un formato sencillo pero coherente, el estilo de la escritura puede ser formal (si está dirigido para usuarios ya con conocimientos) o semi-formal. El seguimiento de la escritura debe ser fácil para que el usuario pueda seguir y encontrar con facilidad la información necesaria, además de implementar imágenes de guía. También se diferencia entre los títulos, subtítulos, advertencias, cuidados, etc.

El estilo y el formato del manual son elegidos por los miembros del equipo que se pondrán de acuerdo luego reconocer el tipo de usuario al cual este manual será dirigido. A raíz de eso se comenzará con la creación del mismo, preferentemente se sugiere ir creando el manual en un idioma para luego traducirlo.

Debido a que el destinatario del manual poseería conocimientos previos sobre el uso de dispositivos y herramientas en relación a la informática y la comunicación, nuestro manual omitirá explicaciones introductorias al uso de sistemas. Esto nos lleva a describir exclusivamente la organización, acceso, permisos y posibilidades implementadas dentro del software.

Primero se comienza con identificación de las partes que tendrá el mismo, especificando que irá primero antes de la inicialización del programa, que debe hacer el usuario para que este inicie correctamente, adjuntando imágenes de guía y el listado de pasos. Luego se deberá especificar que hará el usuario luego de haberlo hecho, en nuestro caso que es lo primero que ve el usuario al ingresar al programa, explicando que es lo que el usuario está viendo y que es lo que debe hacer. El manual irá especificando el uso de cada interfaz que el usuario está visualizando, con sus respectivos títulos, sus imágenes de guía, pasos a seguir, etc.

Luego de la realización de la especificación del programa y su funcionamiento, se deberá realizar un índice, para una mejor ayuda al momento de buscar alguna información concreta que el usuario desee, sin que este tenga que estar leyendo todo el manual para encontrarla.

Al tener finalizado el manual en un idioma se comienza con la traducción de este, teniendo en cuenta la estructura y organización, haciendo el mismo procedimiento que se realizó al comienzo.

Para poder finalizar la realización del manual bilingüe se exporta el mismo en los formatos pdf, docx y txt (sin imágenes en este último formato), además para una mejor distribución se lo imprime. En conjunto poseería un archivo del manual en formato de audio.

## Manual:

* 1. Primera ventana:



El primer formulario que se observa requiere el ingreso del usuario en el primer campo en blanco, la contraseña en el segundo.

Luego de esto y si los datos son válidos se podría abrir la ventana de configuración o entrar al sistema.

* 1. Ventana de configuración



Lo primero que se notará es que posee varios campos de texto para ingresar los datos de la conexión, con el correspondiente tipo de dato que se requiere. Posee la opción de restablecer para volver a los valores por defecto, o guardar los datos ingresados en el momento.

También posee en el panel una pestaña para administrar los usuarios (solo está disponible si el usuario tiene dicho permiso).

* 1. Pestaña de usuario



En la pestaña de usuario se puede observar una sección para la edición y otra para la creación de usuarios.

En la sección de creación hay un campo para el nombre y la contraseña para el usuario a crear, además de dos botones uno para crear y otro para limpiar.

En la sección de edición se observa una tabla con el id y el nombre de usuario, y al seleccionar alguno este se cargará en la sección de edición, pudiendo modificar el nombre y/o la contraseña. Siguiente a esto se puede aplicar los cambios a los usuarios o eliminar dicho usuario.

1. Pantalla de inicio





* 1. La región izquierda de la interfaz principal está dedicada a las tandas.

La división superior se puede visualizar las próximas tandas, enseñando la hora de comienzo y la hora de final. Mientras que la división inferior se visualizan los temas de las publicidades, junto a la fecha, la cual el usuario pude buscar una en especifico.



* 1. La región derecha de la interfaz, en la división superior está dedicada a los próximos eventos, mostrando el nombre y la fecha del mismo. En la división inferior hay un cuadro de notas para que el usuario pueda escribir algún recordatorio.
  2. En la región central encontramos dos pestañas: programas y multimedia.



Pestaña programas: Se visualizan los próximos programas, mostrando la hora de inicio, la hora final, el programa y el estado del mismo. El usuario podrá seleccionar una fecha en específico si desea ver qué programas se emitirán.

Seleccionando un programa, a su derecha se muestran los datos del mismo, los cuales son: la descripción, los funcionarios que trabajan en él y los temas de las publicidades.



Pestaña multimedia: Esta pestaña está dedicada para la búsqueda de vídeos y series.

# Medidas de Seguridad

# Medidas de seguridad

Medidas para la ejecución del software:

El programa será ejecutado por el usuario administrado del sistema operativo.

Medidas de seguridad al momento de la utilización del Software:

Existen dos tipos de usuarios, “administrador” y “funcionarios”, que tienen implementadas funciones específicas.

Al momento de la inicialización del programa el consumidor debe ingresar su usuario y contraseña correspondiente.



## 

*Usuario administrador:*

El usuario administrador, es el que tiene todos los permisos asignados en el programa (Edición, selección, actualización, eliminar), es el único usuario que puede crear, eliminar y editar más usuarios, y es el que asigna los permisos a los mismos. Además de tener permitido todos los accesos a las áreas del programa. Este usuario será utilizado por el dueño de la empresa.

*Usuario Funcionario:*

Los usuarios funcionaros son creados por el usuario administrador, el cual le asigna los permisos correspondientes a su función y la utilización que le dará al programa. Además de que estos usuarios pueden estar restringidos a ciertas áreas del programa. Esos usuarios serán utilizados por funcionarios.

Sistema de Logueo:

Las credenciales de usuario están guardadas en la base de datos, en una tabla llamada Usuarios.

Los nombres de usuario están guardados en texto simple. Las contraseñas, por otro lado, están encriptados usando AES con una key convertida por SHA-256 al momento de ser usadas.

Las keys están almacenadas en un archivo llamado Key.txt dentro de la carpeta bin/User/ (Sujeto a cambios). Encontrándose allí una key común de 8 caracteres y una key maestra de 32 caracteres. Ambas son un string de letras y números aleatorios cada una. Si dicho archivo no existe, el programa generara un nuevo conjunto de keys y creara el archivo.

Al momento de loguearse en el programa, la contraseña ingresada es convertida con el método ya mencionado y comparado con la base de datos.

Tanto como para entrar al programa o cambiar la configuración de la base de datos, el usuario deberá ingresar credenciales validas.

Datos de Conexión a la Base de Datos:

Los datos requeridos para que el programa se conecte a la base de datos (usuario, contraseña, IP, puerto, nombre) se guardan en el archivo User.txt dentro de la carpeta bin/User/ (sujeto a cambios).

Los datos dentro de dicho archivo se encuentran encriptados en Base64 por encima de TripleDES, usando ambas keys del archivo Key.txt.

El administrador puede cambiar los datos de conexión en la ventana de Configuración. Los cambios serán guardados en el archivo mencionado anteriormente.

Además del archivo User.txt, se encontrara un archivo Origin.txt dentro de la misma carpeta, conteniendo los datos por defecto. Si el administrador no ha hecho ningún cambio, ambos archivos serán idénticos; si el archivo User.txt no existe, uno nuevo será creado con los datos de Origin.txt.

# Infraestructura de la empresa – Red de la empresa

# Configuración de red:

## Planos de la distribución:



## Conexión a la red:



Sala de programas:



**Router:**

Rutas estáticas:

IP: 192.168.4.0 - 255.255.255.0 10.0.2.3

**PC’S:**

PCPrograma:

Dirección IP: 192.168.4.2

Máscara: 255.255.255.0

Gateway por defecto: 192.168.4.1

PCPrograma:

Dirección IP: 192.168.4.3

Máscara: 255.255.255.0

Gateway por defecto: 192.168.4.1

Sala de administración:



**Router:**

Rutas estáticas:

IP: 192.168.3.0 - 255.255.255.0 10.0.2.1

**PC’S:**

PCAdministración:

Dirección IP: 192.168.3.2

Máscara: 255.255.255.0

Gateway por defecto: 192.168.3.1

Sala de radio:



**Router:**

Rutas estáticas:

IP: 192.168.1.0 - 255.255.255.0 10.0.2.2

**PC’S:**

PCRadio:

Dirección IP: 192.168.1.2

Máscara: 255.255.255.0

Gateway por defecto: 192.168.1.1

PCRadio1:

Dirección IP: 192.168.1.3

Máscara: 255.255.255.0

Gateway por defecto: 192.168.1.1

PCRadio2:

Dirección IP: 192.168.1.4

Máscara: 255.255.255.0

Gateway por defecto: 192.168.1.1

PCRadio5:

Dirección IP: 192.168.1.5

Máscara: 255.255.255.0

Gateway por defecto: 192.168.1.1

## Configuración de router:

***Router Radio:***

Router#configure terminal

Router(config)#hostname Router Radio

Router(config)#enable secret redes

Router(config)#interface fastEthemet 0/0

Router(config-if)#ip address 192.168.1.0 - 255.255.255.0

Router(config-if)#no shutdown

Router(config)#exit

Router(config)#interface serial 0/1

Router(config-if)#ip address 10.0.2.2 255.255.255.252

Router(config-if)#no shutdown

Router(config-if)#exit

Router(config)#line console 0

Router(config-line)#password redes

Router(config-line)#login

Router(config-line)#line vty 0 4

Router(config-line)#password redes

Router(config-line)#login

Router(config-line)#exit

Router(config)#exit

***Router Programa:***

Router#configuracion terminal

Router(config)#hostname Router Admnistración

Router(config)#enable secret redes

Router(config)#interface fastEthemet 0/0

Router(config-if)#ip address 192.168.4.0 - 255.255.255.0

Router(config-if)#no shutdown

Router(config)#exit

Router(config)#interface serial 0/1

Router(config-if)#ip address 10.0.2.3 255.255.255.252

Router(config-if)#no shutdown

Router(config-if)#exit

Router(config)#line console 0

Router(config-line)#password redes

Router(config-line)#login

Router(config-line)#line vty 0 4

Router(config-line)#password redes

Router(config-line)#login

Router(config-line)#exit

Router(config)#exit

***Router Administración:***

Router#configuracion terminal

Router(config)#hostname Router Admnistración

Router(config)#enable secret redes

Router(config)#interface fastEthemet 0/0

Router(config-if)#ip address 192.168.3.0 - 255.255.255.0

Router(config-if)#no shutdown

Router(config)#exit

Router(config)#interface serial 0/1

Router(config-if)#ip address 10.0.2.1 255.255.255.252

Router(config-if)#no shutdown

Router(config-if)#exit

Router(config)#line console 0

Router(config-line)#password redes

Router(config-line)#login

Router(config-line)#line vty 0 4

Router(config-line)#password redes

Router(config-line)#login

Router(config-line)#exit

Router(config)#exit

# 

# Anexo

## S.R.L.

ContraroSociedad de Responsabilidad Limitada

En la ciudad de Salto, en fecha de 6/4/2020 reunidos los señores Nahuel Pacheco con el número de cedula 5.432.959-1, Milagros Núñez con el número de cedula 5.257.879-8, Mateo Cabral con el número de cedula 5.538.392-2 y Manuel Buslón con la cedula número 5.328.350-8 resuelven celebrar el siguiente contrato de Sociedad de Responsabilidad Limitada que se regirá por las cláusulas que se indican a continuación:

PRIMERO. (Denominación) La naturaleza de la sociedad será de responsabilidad limitada, para la que adoptan la denominación de Nu-CaBuPa.SRL. Se regirá por la Ley 16.060, el decreto 155/010 de 24 de mayo de 2010, y demás normas acordadas. Los socios

SEGUNDO. (Domicilio) La sociedad tendrá su domicilio en el departamento de Salto, pudiendo establecer sucursales, filiales, agencias o representaciones en todo el país o en el extranjero.

TERCERO. (Plazo) El plazo de duración de la sociedad será de 8 años a contar de hoy, no obstante, cada año, a contar desde hoy, cualesquiera de los socios podrá hacer uso del derecho de renuncia, debiendo en tal caso hacer saber su decisión con una anticipación no menor de tres meses al vencimiento del período respectivo, a los demás socios por telegrama colacionado.

CUARTO. (Objeto) La sociedad tendrá por objeto la realización de la siguiente actividad: A) Desarrollo de Software. B) Gestión, administración, soporte de los productos desarrollados por nuestra empresa; C) Ayuda técnica con el uso del sistema informático D) realización de toda clase de actos y/u operaciones civiles, comerciales, industriales, ya sea por cuenta propia, de terceros o de ambos a la vez, pudiendo a tales efectos comprar, vender, hipotecar, caucionar, administrar, explotar, arrendar y dar en arrendamiento, cualquier clase de bienes y derechos, sean muebles, inmuebles, corporales o incorporales, pudiendo a tal efecto participar en toda clase de sociedades y empresas existentes o que se constituyan en el futuro. La sociedad podrá realizar si lo estima conveniente para el mejor desarrollo de la empresa o para un integral aprovechamiento de su capital, cualquier acto, negocio, o contrato de cualquier naturaleza y de cualquier ramo de actividad comercial. Como medios adecuados para el cumplimiento de sus objetivos, la sociedad podrá realizar todos los actos jurídicos referidos para el desarrollo de sus negocios, tanto sean de disposición, afectación y administración, otorgando y efectuando cualquier clase de operaciones ya fueran de enajenación, adquisición, afectación y gravámenes sobre toda clase de bienes y derechos, sin limitación alguna, pudiendo al efecto participar en toda clase de sociedades y empresas existentes o que se constituyan en el futuro; E) Implementación, diseño, compra, venta de equipos informáticos y sistemas de redes de ordenadores; F) Desarrollo de contenido audio visual con fines de entrenamiento; G) Traducción y exposición de cursos y conferencias en relación al ámbito tecnológico-psicológico-filosófico-matemático; H) Instalación y obtención de software relacionado a herramientas de edición de texto.

QUINTO. (Capital) El capital de la sociedad se fija en la suma de $288.640 (pesos uruguayos que quedan divididos en 40 cuotas)

Cuotas de $7.216 cada una, correspondiéndole a los socios Sres.

La sociedad estará conformada por: Manuel Buslón, Mateo Cabral, Milagros Nuñez, Nahuel Pacheco. Cada uno aportando 10 cuotas, equivalentes a $72.160, lo cual ya fue pagado.

SEXTO. (Aportes) Por su obligación de aportar, cada uno de los socios manifiesta que aportó a la sociedad antes de este acto y en efectivo, la suma correspondiente a sus cuotas, con lo que queda integrado en su totalidad el capital societario previsto, razón por la cual se otorgan recíprocas cartas de pago por sí y en representación de la sociedad.

SEPTIMO. (Calidad de los socios) La responsabilidad de los socios quedará limitada a la cantidad o cantidades estipuladas como aporte de capital en el contrato social. ————————————–

OCTAVO. (Administración) La administración de la sociedad y el uso de la firma social, con las más amplias facultades y con el ejercicio de la representación de la empresa estarán a cargo de cualquiera de las socias, actuando indistintamente, sin perjuicio de la facultad de otorgar mandatos a terceros. A título enunciativo se establece que el administrador en nombre y representación de la sociedad podrá: a) realizar toda clase de actos de disposición, administración y afectación quedando facultado para enajenar y/o gravar toda clase de bienes; b) celebrar toda clase de contratos; c) ejecutar todo género de actos de administración; d) otorgar todo tipo de mandatos; e) representar a la sociedad ante cualquier autoridad judicial o administrativa; f)suscribir todo tipo de documentos civiles y/o comerciales; g) registrar todo tipo de Marcas y Patentes. El administrador tendrá todas las facultades necesarias para el gobierno, administración y disposición de los bienes de la sociedad, así como también la representación judicial, extrajudicial y administrativa de la misma, y podrá actuar por sí o hacerse representar por apoderado con facultades suficientes. El domicilio del administrador será el indicado como suyo en la comparecencia de este contrato. El administrador deberá suscribir la documentación de la siguiente forma: por S.R.L. seguido de su firma habitual. ——————————————–

NOVENO. (Inventario-balance) Anualmente se efectuará un inventario-balance al cierre del ejercicio económico el que deberá estar concluido dentro de los ciento veinte (120) días siguientes a la finalización del ejercicio. Los socios tendrán las más amplias facultades para controlar la confección de los balances y formular por escrito las observaciones que les merezcan. Los socios acuerdan que el ejercicio económico finalice cuando lo establezca la asamblea de socios. ————

DECIMO. (Fondo de reserva.) De las ganancias líquidas de cada ejercicio se deducirá un diez por ciento (10%) para formar un fondo de reserva, hasta que éste alcance un cien por ciento (100%) del capital social. ———————————————————————————–

DECIMO PRIMERO. (Distribución de ganancias y pérdidas) El resto de las ganancias serán repartidas y las pérdidas soportadas, entre los socios en proporción de sus aportes. Las pérdidas de cada ejercicio serán compensadas con las utilidades del o de los ejercicios subsiguientes.—————————————————————

DECIMO SEGUNDO. En caso de ausencia, fallecimiento o incapacidad de cualquiera de los socios, la sociedad continuará entre los demás socios.——– ——————————————————————

DECIMO TERCERO. La sociedad se disolverá cuando así lo decida la Asamblea Extraordinaria convocada a tales efectos y será ésta quien decidirá la o las personas que realizarán la liquidación de los bienes sociales fijando sus atribuciones y remuneraciones. Una vez liquidados los bienes sociales del activo y abonadas todas las obligaciones de la sociedad, el remanente será adjudicado a los socios en proporción a sus respectivos capitales. La sociedad puede disolverse por las siguientes causales: A); Por decisión unánime de los socios; B) Por expiración del plazo; C) Por pérdidas que reduzcan el patrimonio social a una cifra inferior a la cuarta parte del capital social integrado; D) Por reducción a uno del número de socios según lo dispuesto en el art. 156 de la ley 16.060. ————————————————————

DECIMO CUARTO. (Cesión de cuotas sociales). Las cuotas sociales podrán ser cedidas por cualquiera de los socios conforme a las disposiciones legales. Cuando un socio desee ceder su cuota a un tercero ajeno a la sociedad, se convocará a una Asamblea Extraordinaria, diez (10) días después de que el socio interesado comunique por telegrama colacionado la intención de ceder su cuota. En tal caso la cesión deberá ser aceptada por unanimidad. Si no se notificara la oposición se presumirá el consentimiento. Si se formulara alguna oposición se seguirá el procedimiento previsto por el art. 232 de la ley 16.060. La sociedad tendrá prioridad frente a los socios para la adquisición de las cuotas. Si hubiera varios socios interesados en la adquisición, las cuotas se distribuirán a prorrata y si no fuera posible, se atribuirán por sorteo.———————————————————————————–

DECIMO QUINTO. Los socios podrán por unanimidad, fijarse remuneraciones en calidad de sueldo y otros conceptos, así como autorizar retiros a cuenta de utilidades con débito a sus respectivas cuentas. —————————————————————-

DECIMO SEXTO. Los socios se obligan muy especialmente a aportar su trabajo a los negocios de la sociedad, pudiendo no obstante realizar operaciones mercantiles por su cuenta o como integrante de otras empresas, siempre que su realización no perjudique a la sociedad, ni menoscabe la actividad que le corresponda desarrollar en la misma. —-

DECIMO SEPTIMO. Anualmente dentro de los ciento veinte (120) días de cerrado el ejercicio económico se celebrará una Asamblea Ordinaria que tendrá por objeto aprobar el balance, examinar y juzgar la gestión de la administración, así como la cuenta de ganancias y pérdidas, establecer el porcentaje destinado al capital de reserva cuando éste fuera mayor al fijado, y resolver cualquier otro punto que se encuentre en el orden del día. Los miembros de la Asamblea serán citados por telegrama colacionado, con una anticipación de por lo menos diez (10) días a la fecha de realización de la misma. La Asamblea Extraordinaria podrá ser convocada en cualquier momento, por mayoría de socios, debiendo comunicar por escrito a los restantes la realización de la misma con una antelación no menor a diez (10) días. El quórum necesario para sesionar y tomar resoluciones en Asambleas Ordinarias o Extraordinarias, salvo disposición en contrario, estará constituido por la mayoría de los socios que representen la mayoría del capital social, correspondiéndoles a tales efectos un voto por cuota social. ——————————————

DECIMO OCTAVO. Queda prohibido a la sociedad constituirse fiadora de los socios y de terceras personas. ———————————–

DECIMO NOVENO. La asamblea social, por decisión de la mayoría de socios que represente las tres cuartas partes del capital social podrá tomar las siguientes decisiones: a) transformar a la SRL en sociedad anónima; b) fusionarla con otra; c) modificar el objeto social; d) revocar y nombrar administradores; e) prorrogar el plazo de la sociedad. Los socios disidentes o ausentes tendrán derecho a receso. ———————————————————————————-

VIGESIMO. Los socios podrán renunciar a la sociedad en los siguientes casos: a) en caso establecido en la cláusula tercera de este contrato; b) si el ejercicio social diere una pérdida que excediere el 30% del capital social; c) si no se conformaren con la resolución que tomen los restantes socios respecto a lo previsto en la cláusula décimo novena de la presente. En tales casos la participación del socio saliente se determinará conforme al balance practicado del día de la manifestación de voluntad, y la cuota que correspondiere le será abonada por la sociedad hasta en cuatro mensualidades consecutivas, iguales a contar de la fecha del balance, las que devengarán el máximo interés legal, calculado sobre los saldos deudores pagadero conjuntamente con el capital. ———————————————————————————

VIGESIMO PRIMERO. Cualquier diferencia o controversia entre los socios, sea por aplicación o interpretación de este contrato, será resuelto inapelablemente por árbitros, designados uno por cada una de las partes discrepantes y un tercero designado por los árbitros ya designados. La designación se hará dentro del plazo de 30 días a contar del telegrama colacionado que el discrepante deberá enviar a la administración de la sociedad, manifestando su disconformidad. —–

VIGESIMO SEGUNDO. En todo lo que no estuviere previsto en este contrato se aplicarán las disposiciones de la ley 16.060 sección IV, “De las sociedades de responsabilidad limitadas” y sus concordantes.———

ENTREVISTA

# Entrevista:

Luego de la presentación de nuestro grupo, contando quiénes somos y lo que podemos ofrecer al cliente se dio la entrevista de requerimientos.

Grupo: ¿Puede contarnos qué tipo de actividades realiza su empresa?

Cliente: Principalmente la gestión y el control de la radio Bemba

Grupo: ¿Y cómo se estuvo administrando los datos de la radio hasta ahora?

Cliente: Tengo una libreta en el que anoto los pagos de cada programa y más o menos los horarios son sobre lo que se recuerda y lo que queda hablado con el dueño de cada programa. Las publicidades dependen de la tanda en que se contrataron y queda guardado lo que se debe decir o mostrar en la computadora

Grupo: ¿Y de esos programas le interesaría almacenar alguna información?

Cliente: Solo más o menos saber que conductores hay y los pagos de cada mes por parte del dueño del programa. También estaría el operador pero es casi siempre el mismo conductor el que hace esa función y actúa como el dueño.

Grupo: ¿Los precios para los programas están definidos antes o se acuerdan con el que alquilaría el espacio?

Cliente: Se fijan en el momento teniendo en cuenta lo que duraría, repeticiones por semana y de que trata

Grupo: Entonces ¿un programa puede tener varios días distintos cada semana?

Cliente: Sí, y eso puede cambiar

Grupo: ¿Y el tema de publicidades cómo es?

Cliente: En el momento de la tanda entre cada programa se comentan o muestran las publicidades que contratan las empresas, cada una con las repeticiones que pagó. De las publicidades que aparecen en cada programa no interesaría, ya que el dueño del programa se encarga de eso.

Grupo: De eso ¿solo le interesaría saber lo de los pagos, la tanda en que aparecen y las repeticiones entonces?

Cliente: Sí, me gustaría que quede guardado algo sobre publicidades que han sido contratadas y los pagos para luego calcular. ¿Se podrá?

Grupo: Sí, se podría a base de la información del pago de los programas y las publicidades hacer que el software calcule las ganancias de los meses o días que se quiera saber. ¿En la empresa quienes usarían el software?

Cliente: Principalmente yo y otro que esté ahí ayudando a registrar las cosas, el operador de la radio puede ser. También quisiera poder usarlo desde casa u otro lugar.

Grupo: Bueno, podemos hacer que se administre de forma online. ¿Tiene un servidor propio?

Cliente: Sí, tengo uno en el que está la página web de la radio y en el que se retransmite la misma

Grupo: ¿Qué equipos tiene? ¿Son equipos de uso propio o exclusivos para la administración de la radio?

Cliente: Mis equipos de administración son de alta eficiencia, en la radio las computadoras son exclusivos para los programas, mientras que la computadora que tengo en mi casa está más para el trabajo.

Grupo: Está bien, en una futura entrevista acordamos como implementamos eso y qué características tiene el servidor si se puede usar ese. ¿Hay algo más que quisiera mantener en los registros?

Cliente: Cada tanto hay eventos que se realizan con algún tema por la empresa, lo importante sería saber la fecha y de que trata más o menos. Después está lo de Mil voces y otras series que produciríamos, de eso es solo tener que video de la serie se va a emitir y de que trata.

Grupo: ¿Esos eventos tienen relación con algo de la radio o las series?

Cliente: Puede tratar de eso sí, estar en conjunto con algún programa sobre el que se realice o por algún video o tema de la serie para promocionarlo. En el evento se mostraría publicidades también.

Grupo: ¿Algo más que quiera poder administrar?

Cliente: Por ahora serían solo esas cosas, si pienso algo más les envío un mensaje.

Grupo: ¿Le gustaría mantener control de las publicidades que aparecerían en algún programa de la radio?

Eso dentro del software

Cliente: Sí, me gustaría, claro si es posible.

# Consultas de SQL

# Consultas de SQL

#Se utilizan los “describe” directamente en el software para obtener nombre y tipo de las columnas de las tablas, recibiéndolas y aplicando funciones y métodos. Es necesario para poder generalizar funciones y que sean independientes, ahorrando el incluir todo.

describe Empresa

describe video

describe Serie

#Los delete tienen su propia sección en la cual permite editar o borrar los datos del software, para eso existe un método que recibe el nombre de la tabla, la columna(s) a comparar un array de los datos a borrar. Esta crea la sentencia sql y utiliza un in ya que usualmente recibe varios datos como se ve abajo.

DELETE FROM video WHERE id\_serie in ('9', ‘6’)

DELETE FROM Video WHERE id\_video in ('6', ‘1’, ‘9’ )

DELETE FROM Serie WHERE id\_serie in ('8' , ‘5’ )

DELETE FROM Empresa WHERE id\_empresa in ('2' )

#Esta sentencia se utiliza para verificar si existe el usuario ingresado en el login, utiliza el @nombre remplazando eso por el nombre ingresado y el @contraseña para lo mismo, a esto se utiliza un sha2 en conjunto a una key almacenada en los archivos del software, sin la cual el login no funcionará, a esto a su vez se aplica un aesencrypt.

select id\_usuario FROM usuarios as User WHERE nombre = @nombre AND contrasena = AES\_ENCRYPT(@contrasena,sha2(@key,256))

#Se selecciona la hora de inicio y final de las próximas tandas, esto se mostrará en el menú principal y permite seleccionar una para ver sus publicidades

selecttime\_format(hora\_inicio, '%H:%i') as 'Inicio', time\_format(hora\_fin, '%H:%i') as 'Final' FROM tanda WHERE (hora\_inicio<= curtime() and hora\_fin>= curtime()) orhora\_inicio>= curtime()

#Se selecciona el id, la hora de inicio, de fin y el nombre de los programas del día actual, aunque el dato cambia dependiendo de lo seleccionado en un datatimepicker. Esto se modificará, se oculta el id y se lo carga en una tabla.

select p.id\_programa, time\_format(hora\_inicio, '%H:%i') as 'Inicio', time\_format(hora\_fin, '%H:%i') as 'Final', Nombre\_programa as 'Programa' FROM fechaprograma f inner join programa p on f.id\_programa=p.id\_programa WHERE fecha = '2020-09-26'

#Se selecciona la fecha de los próximos eventos para mostrarlos en una lista

select e.id\_Evento, DATE\_FORMAT(Fecha,'%d/%m/%Y') as Fecha, Nombre FROM evento e inner join fechaevento f on f.id\_evento=e.id\_evento WHERE f.fecha>= now()

#Se cargan los temas de una publicidad de la tanda seleccionada en el día seleccionado, por defecto es el día actual.

select Tema FROM publicidad p innerjoinaparecepubli a onp.id\_publicidad=a.id\_publicidadinnerjoin tanda t ont.Hora\_Inicio = a.hora\_inicio WHERE a.fecha\_inicio<= '2020-09-26' and a.fecha\_finalizacion>= '2020-09-26' and t.hora\_inicio = '15:00:00'

#Muestra el nombre y teléfonos de los funcionarios que trabajan en un programa seleccionado

select Nombre, Telefono FROM programa p innerjoinfuntrabaja f onf.id\_programa=p.id\_programainnerjoin funcionario ffonff.id\_funcionario = f.id\_funcionario WHERE p.id\_programa = '3'

#Se utiliza para mostrar la descripción del programa seleccionado

select Descripcion FROM programa WHERE id\_programa = '3'

#Similar a las publicidades cargadas para las tandas en esta sección se cargan las publicidades del programa seleccionado en el día que se desee

select Tema FROM programa p inner join pmuestrapubli pp on p.id\_programa=pp.id\_programa inner join publicidadppp on pp.id\_publicidad = ppp.id\_publicidad WHERE pp.fecha\_inicio<= '2020-08-10' and pp.fecha\_finalizacion>= '2020-08-10' and pp.id\_programa = '1'

#Esta es necesaria para mostrar los datos de los videos, utilizamos en where true debido a que las consultas están generalizadas y requieren tres parámetros para el método, se puede simplemente eliminar, pero preferimos mostrar tal como quedan los logs del software

#Realizamos un select para la búsqueda de videos en su interfaz correspondiente, la subconsulta es útil porque ayuda a que aunque el video no tenga serie este se mostrará en la interfaz, como es una relación de 1 a n no hay posibilidad de que devuelva más de uno.

selectid\_video, fecha as Fecha, v.nombre as Nombre, (selects.nombrefrom serie s wheres.id\_serie=v.id\_serie) as Serie FROM video v WHERE true

#Se selecciona el contenido, nombre y serie de los videos para cuando selecciona alguno y lo quiere ver o editar, estos datos seleccionados se arreglan para mostrar una interfaz con ellos, el id\_serie se usa para seleccionas la serie en un combo box de las series a las que se puede asociar.

select contenido, Nombre, id\_serie, DATE\_FORMAT(Fecha,'%Y-%m-%d') as Fecha FROM video WHERE id\_video = '6'

#Se seleccionan las series para mostrar en la sección de asignación en la edición o agregación de un video, el id se oculta y guarda en otra parte.

select id\_serie, nombre FROM Serie WHERE True

#Se hace un update para actualizar los datos del video a los elegidos por el usuario, el null es para quitarlo de una serie o su fecha de emisión.

update video set contenido = 'sid',nombre = 'no',ID\_Serie = null,fecha = null WHERE ID\_Video = '6'

#Se actualizan los datos de una serie desde su formulario de edición y al guardar se actualizan con lo ingresado en los campos

update Serie set fecha\_finalizacion = null,nombre = 'holid' WHERE ID\_Serie = '6'

#Se seleccionan las series con sus nombres y fechas para el menú de búsqueda

selectid\_serie, fecha\_finalizacion as Fecha, nombre as Nombre FROM serie WHERE true

#Se muestran los temas de las publicidades asociadas a una empresa en su menú de datos

selectID\_Publicidad, Tema FROM publicidad WHERE ID\_Empresa = '1'

#Se actualizan los datos de la empresa en su interfaz de edición

update Empresa set Nombre = 'Mil',Telefono = '092521203',Mail = 'dd@gmail.com' WHERE ID\_Empresa = '1'

#Se ingresa un video con los datos que el usuario desea

Insertinto video ( contenido,nombre,fecha ) values ('None','Voces de ...','2020-02-02' )

#Se ingresa una serie con los datos que el usuario desea

Insertinto serie ( ID\_Serie,nombre ) values ('10','Voces' )

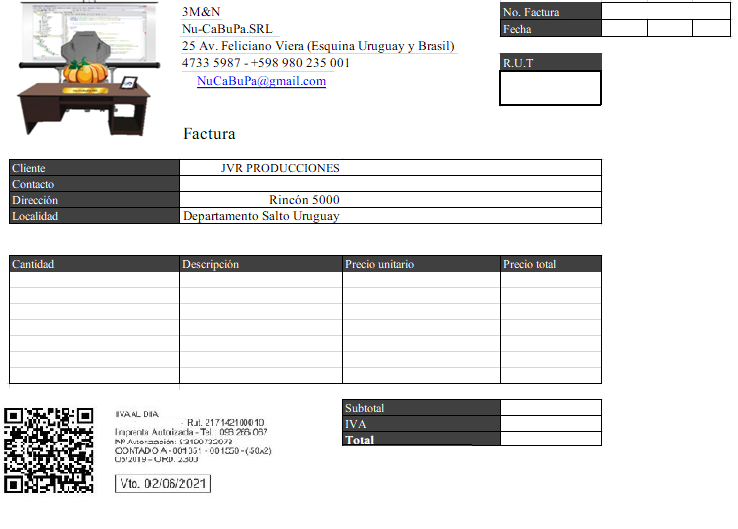
#Se ingresa un video con los datos que el usuario desea pero desde una interfaz de administración, dicha interfaz solo se puede acceder con permisos de administrador y permite ingresar datos en todas las tablas de forma más accesible que en consola

Insertinto video ( ID\_Video,nombre,ID\_Serie ) values ('15','None','10' )

#Se ingresa una empresa con los datos que el usuario desea

Insertinto empresa ( Nombre ) values ('NuCaBuPA' )

### Documento comercial



### Proyecto S.O

1. Crear un script que permita crear un árbol de directorios (con subdirectorios y archivos). En primer lugar se debe preguntar cuál será el directorio “base” del árbol, que deberá crearse dentro del directorio personal del usuario utilizado.

Posteriormente se deben mostrar opciones dando la posibilidad de crear un archivo o subdirectorio, y en ambos casos preguntar dentro de qué rama del árbol se lo desea crear. A medida que se van agregando archivos y subdirectorios, deberá desplegarse el árbol correspondiente para poder verificar que se hizo en forma correcta.

Implementarse **al menos 2 funciones** de modo de no repetir código innecesariamente: una para crear directorio y otra para crear archivo.

2. Utilizando el árbol creado anteriormente, crear otro script que permita renombrar una lista de archivos, cambiando alguno de sus caracteres. En primer lugar se deberá mostrar el árbol de directorios junto a archivos, preguntar en qué directorio se desea hacer la modificación, luego el carácter a ser cambiado y por cuál. Antes de proceder al cambio, mostrar qué se renombrará y pedir confirmación, en cuyo caso se procederá a realizarlo.

#### Script 1:

#!/bin/bash

Verificar(){

    Direcciones=($(find "$TD" -type d))

    Result=${#Direcciones[@]}

    if [ $Result -gt 1 ]; then

        ExisteDir="Si"

    else

        ExisteDir="No"

    fi

}

Crear () {

    printf "%s\n" "Desea crear un archivo o subdirectorio?" "1)Archivo" "2)Subdirectorio" "Otro)Salir"

    read OP

    Eleccion=""

    if [ $OP = "1" ]; then

        Eleccion="Archivo"

    elif [ $OP = "2" ]; then

        Eleccion="Subdirectorio"

    else

        exit

    fi

    clear

    if [ $ExisteDir = "Si" ]; then

        RecD "$TD"

    else

        DirTemp="$TD"

        Nombre

    fi

}

Nombre(){

    printf "%s\n" "Indique un nombre para el $Eleccion"

    read Nombre

    if [ $OP = "1" ]; then

        touch "$DirTemp/$Nombre"

    elif [ $OP = "2" ]; then

        mkdir "$DirTemp/$Nombre"

    else

        Error "Nombre"

    fi

    clear

    printf "%s\n" "$Eleccion creado exitosamente en $TD"

    tree "$TD"

    Verificar

    Crear

}

#Recorre el directorio

RecD () {

    printf "%s\n" "Donde desea crearlo?" "(Ruta o nombre. Ingrese un . para crearlo en el directorio $TD)"

    tree "$TD"

    read nombre

    DirTemp="$TD"

    if [ "$nombre" = "." ]; then

        Nombre

        exit

    elif [[ "${nombre:0:1}" = "/" ]]; then

        Buscar "wholename"

    elif [[ "$nombre" =~ "/" ]]; then

       nombre="$DirTemp/$nombre"

       Buscar "wholename"

    else

        Buscar "name"

    fi

}

Buscar(){

    Direcciones=($(find "$DirTemp" -type d -$1 "$nombre" | sed "s/ /'/g"))

    Result=${#Direcciones[@]}

    if [ $Result -eq 1 ] && [ "${Direcciones[0]}" != "''" ]; then

        DirTemp=$(echo ${Direcciones[$i]} | sed "s/'/ /g")

        printf "%s\n" "Encontrado: $DirTemp"

        Nombre

    elif [ $Result -gt 1 ]; then

        printf "%s\n" "A cual directorio se refiere?"

        Num=0

        iter=`expr $Result - 1`

        for i in `seq 0 $iter`; do

            printf "$Num) %s\n" "$(echo ${Direcciones[$i]} | sed "s/'/ /g")"

            Num=`expr $Num + 1`

        done

        read Num

        eleccion=${Direcciones[$Num]}

        echo $eleccion

        if [ -d "$eleccion" ]; then

            DirTemp=$eleccion

            Nombre

        else

            Error "RecD"

        fi

    else

        Error "RecD"

    fi

}

Error(){

        printf "%s\n" "No se encontró" "Ingrese cualquier caracter para continuar o 0 para salir"

        read Salir

        if [ $Salir = "0" ]; then

            exit

        else

            $1

        fi

}

TD=""

DirTemp=""

OP=""

ExisteDir=""

while [ "$TD" = "" ]

do

#Se pregunta cual es la dirección en la que se creará el arbol

printf "%s\n" "Bienvenid@ al generador de archivos/directorios"

printf "%s\n" "Indique el nombre del directorio a crear para el árbol" "(Si ya existe se utilizará ese)" "Ingrese un . para salir"

read TD

    if [ "$TD" = "." ]; then

        exit

    fi

done

#Verificamos que no sea camino absoluto y de que exista

TD=$(echo "$TD" | sed 's/^\///')

TD="$HOME/$TD"

if [ -d "$TD" ] ; then

    printf "%s\n\n" "Directorio encontrado : $TD"

    Verificar

else

    printf "%s\n\n" "Directorio creado"

    mkdir "$TD"

    ExisteDir="No"

fi

echo "$TD" > "$HOME/Direccion.txt"

Crear

#### Script 2:

#!/bin/bash

Verificar(){

    Direcciones=($(find "$TD" -type d))

    Result=${#Direcciones[@]}

    if [ $Result -gt 1 ]; then

        ExisteDir="Si"

    else

        ExisteDir="No"

    fi

}

Letra1(){

    printf "%s\n" "Indique el caracter a ser cambiado"

    read Letra

    Verificacion=$(echo $Letra | egrep "^[a-Z0-9]$")

    if [ Verificacion = "" ]; then

        Error "Letra1"

    fi

    Letra2

}

Letra2(){

    printf "%s\n" "Indique el caracter por el cual reemplazar"

    read LetraN

    Verificacion=$(echo "$LetraN" | egrep "^[a-Z0-9]$")

    if [ Verificacion = "" ]; then

        Error "Letra2"

    fi

    Cambiar

}

Cambiar(){

    Anterior=($(find "$DirTemp" -maxdepth 1 -type f | sed "s/^.\*\///" | egrep "$Letra" | sed "s/ /'/g"))

    Nuevo=($(find "$DirTemp" -maxdepth 1 -type f | sed "s/^.\*\///" | egrep "$Letra" | sed "s/$Letra/$LetraN/g" | sed "s/ /'/g"))

    Canti=${#Anterior[@]}

    if [ $Canti -eq 0 ]; then

        clear

        printf "%s\n" "Ningun nombre posee dicho caracter"

        Error "Letra1"

        exit

    fi

    can=`expr $Canti - 1`

    for i in `seq 0 $can`; do

        printf "%s %s %s\n" "'$(echo ${Anterior[$i]} | sed "s/'/ /g")'" "Pasará a ser:" "'$(echo ${Nuevo[$i]} | sed "s/'/ /g")'"

    done

    printf "%s\n" "Para aceptar inserte 'y'" "Para no renombrar 'n'" "Para salir inserte cualquier otro"

    read Salir

        if [ "$Salir" = "y" ]; then

            Renombrar

        elif [ "$Salir" = "n" ]; then

            clear

            Letra1

        else

            exit

        fi

}

Renombrar(){

    Archivos=$(find "$DirTemp" -maxdepth 1 -type f | egrep "$Letra" | sed "s/ /'/g")

    for o in $Archivos

    do

        i=$(echo "$o" | sed "s/'/ /g")

        OldName=$i

        NewName="$DirTemp/$(echo "$i" | sed "s/^.\*\///" | sed "s/$Letra/$LetraN/g")"

        if [ "$OldName" != "$NewName" ]; then

            mv "$OldName" "$NewName"

        fi

    done

    printf "%s\n" "Resultado: "

    tree "$TD"

    printf "\n%s\n\n" "Si quiere seguir modificando inserte 'y', para salir inserte cualquier otro"

    read Salir

        if [ "$Salir" = "y" ]; then

            clear

            RecD

        else

            exit

        fi

}

RecD () {

    printf "%s\n" "En que carpeta desea cambiar los nombres?" "(Ruta o nombre. Ingrese un . para usar en el directorio $TD)"

    tree "$TD"

    read nombre

    DirTemp="$TD"

    if [ "$nombre" = "." ]; then

        Letra1

        exit

    elif [[ "${nombre:0:1}" = "/" ]]; then

        Buscar "wholename"

    elif [[ "$nombre" =~ "/" ]]; then

       nombre="$DirTemp/$nombre"

       Buscar "wholename"

    else

        Buscar "name"

    fi

}

Buscar(){

    Direcciones=($(find "$DirTemp" -type d -$1 "$nombre" | sed "s/ /'/g"))

    Result=${#Direcciones[@]}

    if [ $Result -eq 1 ] && [ "${Direcciones[0]}" != "''" ]; then

        DirTemp=$(echo ${Direcciones[$i]} | sed "s/'/ /g")

        printf "%s\n" "Encontrado: $DirTemp"

        Letra1

    elif [ $Result -gt 1 ]; then

        printf "%s\n" "A cual directorio se refiere?"

        Num=0

        iter=`expr $Result - 1`

        for i in `seq 0 $iter`; do

            printf "$Num) %s\n" "$(echo ${Direcciones[$i]} | sed "s/'/ /g")"

            Num=`expr $Num + 1`

        done

        read Num

        eleccion=${Direcciones[$Num]}

        echo $eleccion

        if [ -d "$eleccion" ]; then

            DirTemp=$eleccion

            Letra1

        else

            Error "RecD"

        fi

    else

        Error "RecD"

    fi

}

Error(){

        printf "%s\n" "No se encontró" "Ingrese cualquier caracter para continuar o 0 para salir"

        read Salir

        if [ $Salir = "0" ]; then

            exit

        else

            $1

        fi

}

Inicio(){

TD=""

DirTemp=""

OP=""

while [ "$TD" = "" ]

do

    #Se pregunta cual es la dirección en la que se creará el arbol

    printf "%s\n" "Bienvenid@ al renombrador de archivos"

    if [ ! -f "$HOME/Direccion.txt" ]; then

        printf "%s\n" "Indique el nombre del arbol creado" "(O ruta relativa)" "Ingrese un . para salir"

        read TD

        if [ "$TD" = "." ]; then

            exit

        elif [ "$TD" <> "" ]; then

            #Verificamos que no sea camino absoluto y de que exista

            TD=$(echo "$TD" | sed 's/^\///')

            TD="$HOME/$TD"

        fi

    else

        TD=$(cat "$HOME/Direccion.txt")

    fi

done

if [ -d "$TD" ] ; then

    printf "%s\n\n" "Directorio encontrado : $TD"

    Verificar

else

    Error "Inicio"

fi

if [ $ExisteDir = "Si" ]; then

    RecD

else

    DirTemp="$TD"

    Letra1

fi

}

ExisteDir=""

Inicio