MANUAL TÉCNICO

MANUAL TÉCNICO

2015 - 2016

JENNY PATRICIA MORALES MALDONADO

MANUAL TÉCNICO

SISTEMA DE CAPTACIÓN DE REQUERIMIENTOS DE DESARROLLO DE SOTWARE.

El siguiente manual está diseñado para el fácil mantenimiento del aplicativo.

El aplicativo se diseñó en el Lenguaje de programación Visual Studio 2012, con SQL Server 2012, Entity Framework 4.0 y Silverligth 5.0.

Tiene un servidor de aplicaciones y un servidor de base de datos independientes, que se encuentran en las instalaciones de la empresa Farmaenlace Cía. Ltda., bajo la disposición mantenimiento y organización del Dpto. de Redes.

Esta es una aplicación web disponible con la Intranet de Farmaenlace, con al URL: 192.168.238\RequerimientosSoftware

Los módulos del aplicativo son:

- Parametrizaciones
- Especificación
- Revisión
- Reportes

Los servicios ocupados para el aplicativo son:

- Guardar una solicitud creada
- Modificar los campos de la solicitud
- Guardar Observaciones designadas a cada solicitud
- Guarda y Enviar la solicitud creada al usuario revisor correspondiente
- Anular una solicitud creada por falta de coherencia en el contenido
- Buscar solicitudes creadas por usuario
- Revisión de solicitudes generadas en los tres niveles que cumple el ciclo
- Visualizar el Pdf de una especificación creada.
- Listado de solicitudes para modificación
- Listado de solicitudes con respectivos estados
- Carga de información para modificación

- Carga de información para revisión
- Listado de Centros de Costos
- Seguridades de Usuarios

Los Servicios antes mencionados se acceden desde la Url:

http://Sony/WcfReqSoft/Service1.svc

Implementación del Aplicativo

Dependiendo del tipo de solicitud o servicio se va añadiendo la página y los parámetros necesarios.

1. Menú Principal (MainPage.xaml, MainPage.xaml.cs).


```
/*****LOGIN DE LA APLICACION*****/


        LG_Controles.Login.w_login obj_login;
        bool cerrarlogin = false;

        private void UserControl_Loaded_1(object sender, RoutedEventArgs e)
        {

            LayoutRoot.Visibility = System.Windows.Visibility.Collapsed;
            obj_login = new LG_Controles.Login.w_login();
```

```
            obj_login.Show();
            obj_login.ClickAceptar += obj_login_ClickAceptar;
            obj_login.Closed += obj_login_Closed;
        }


        void obj_login_Closed(object sender, EventArgs e)
        {
            if (!App.Current.IsRunningOutOfBrowser)
            {
                if (!cerrarlogin)
                    HtmlPage.Window.Navigate(new Uri("Pag_Blanco",
UriKind.Relative));
            }
        }

        void obj_login_ClickAceptar(object sender, RoutedEventArgs e)
        {



            obj_login.IsEnabled = false;
            if (obj_login.Usuario != "" && obj_login.Clave != "")
            {

                Service_Login.WSEasyLoginSoapClient servicio_login = new
Service_Login.WSEasyLoginSoapClient();
                servicio_login.GetParametroAccessPointCompleted +=
servicio_login_GetParametroAccessPointCompleted;
                servicio_login.GetParametroAccessPointAsync("EASYSEGURIDADNET",
"ACTDIR");

            }
            else
            {

                LG_Controles.MessageBox.MessageBox_SL.show("Por favor ingrese
Usuario y Contraseña", "Advertencia!!",
LG_Controles.MessageBox.MessageBoxButton.ACEPTAR,
LG_Controles.MessageBox.MessageBoxIcon.INFORMACION);
                LG_Controles.MessageBox.MessageBox_SL.WinClose += new
LG_Controles.MessageBox.MessageBox_SL.WinCloseHandler(MessageBox_SL_WinClose);
            }
        }

        void servicio_login_GetParametroAccessPointCompleted(object sender,
Service_Login.GetParametroAccessPointCompletedEventArgs e)
        {

            String ACTDIR = e.Result;
            String AD_Path = ACTDIR.Split(',')[0];
            String AD_User = ACTDIR.Split(',')[1];
            String AD_Pass = ACTDIR.Split(',')[2];
            String mensaje = "";


            Service_Login.WSEasyLoginSoapClient servicio_login = new
Service_Login.WSEasyLoginSoapClient();
            servicio_login.AuthenticateCompleted +=
servicio_login_AuthenticateCompleted;
            servicio_login.AuthenticateAsync(obj_login.Usuario, obj_login.Clave,
mensaje, AD_Path, AD_User, AD_Pass);
```

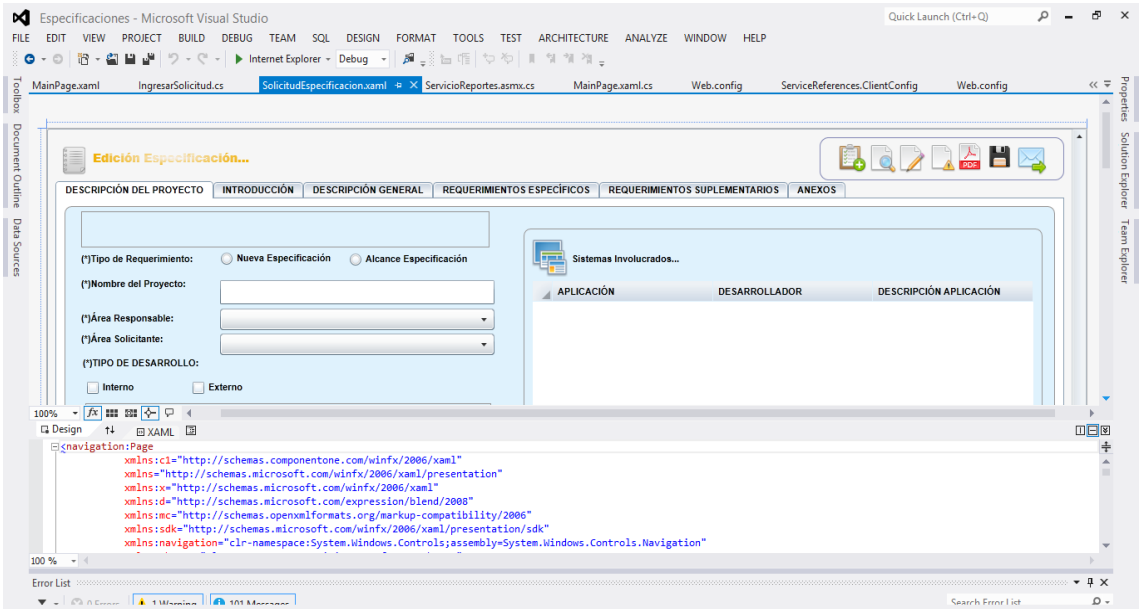
```
        }


        void servicio_login_AuthenticateCompleted(object sender,
Service_Login.AuthenticateCompletedEventArgs e)
        {
            Boolean res = e.Result;
            if (res)
            {

                LG_Controles.MessageBox.MessageBox_SL.showEspera("CARGANDO
PERFIL", "CARGANDO INFORMACIÓN");

                cerrarlogin = true;
                AsignarFunciones.user = obj_login.Usuario;
                AccordionItem d = (AccordionItem)amenu.Items[0];
                d.Header = AsignarFunciones.user.ToUpper();
                obj_login.Close();
                ServicioSOA.Service1Client servicio = new
ServicioSOA.Service1Client();
                servicio.GetAtribucionesCompleted +=
servicio_GetAtribucionesCompleted;
                servicio.GetAtribucionesAsync(AsignarFunciones.user, "CRDS",
AsignarFunciones.user);

                LayoutRoot.Visibility = System.Windows.Visibility.Visible;

            }
            else
            {

                LG_Controles.MessageBox.MessageBox_SL.show("Usuario o Clave
Incorrecto", "Error!!", LG_Controles.MessageBox.MessageBoxButton.ACEPTAR,
LG_Controles.MessageBox.MessageBoxIcon.INFORMACION);
                LG_Controles.MessageBox.MessageBox_SL.WinClose += new
LG_Controles.MessageBox.MessageBox_SL.WinCloseHandler(MessageBox_SL_WinClose);

            }
        }

        void servicio_GetAtribucionesCompleted(object sender,
ServicioSOA.GetAtribucionesCompletedEventArgs e)
        {
            if (e.Result != null)
            {
                List<ServicioSOA.Atribucione> Lts_Atribuciones = new
List<ServicioSOA.Atribucione>();
                Lts_Atribuciones = e.Result;
                foreach (var item in Lts_Atribuciones)
                {
                    if (item.Transaccion.Equals("aplicaciones_desarrolladas"))
                        aplicaciones_desarrolladas.Visibility =
System.Windows.Visibility.Visible;
                    if (item.Transaccion.Equals("funciones_responsables"))
                        funciones_responsable.Visibility =
System.Windows.Visibility.Visible;
                    if (item.Transaccion.Equals("requerimientos_no_funcionales"))
                        requerimientos_no_funcionales.Visibility =
System.Windows.Visibility.Visible;
                    if (item.Transaccion.Equals("solicitud_especificacion"))
                        solicitud_especificacion.Visibility =
System.Windows.Visibility.Visible;
```

```
                    if (item.Transaccion.Equals("revision_auditoria"))
                        revision_auditoria.Visibility =
System.Windows.Visibility.Visible;
                    if (item.Transaccion.Equals("revision_sistemas"))
                        revision_sistemas.Visibility =
System.Windows.Visibility.Visible;
                    if (item.Transaccion.Equals("revision_programador"))
                        revision_programador.Visibility =
System.Windows.Visibility.Visible;
                    if (item.Transaccion.Equals("listReportes"))
                        listReportes.Visibility =
System.Windows.Visibility.Visible;
                }


            }
            LG_Controles.MessageBox.MessageBox_SL.closeEspera();


        }
        void MessageBox_SL_WinClose(object sender,
LG_Controles.MessageBox.CloseEventArgs e)
        {
            obj_login.IsEnabled = true;
            obj_login.EstablecerFoco();
        }




        /*NUEVO MENUU*/


        private void Ocultar_MouseLeftButtonUp(object sender,
System.Windows.Input.MouseButtonEventArgs e)
        {


            titulo.Margin = new Thickness(3);//.Left = 0;
            imgabrir.Visibility = Visibility.Visible;
            lb_titulo.Visibility = Visibility.Collapsed;


        }

        private void imgabrir_MouseLeftButtonUp(object sender,
System.Windows.Input.MouseButtonEventArgs e)
        {
            // TODO: Add event handler implementation here.

            titulo.Margin = new Thickness(181, 7, 8, 8);
            imgabrir.Visibility = Visibility.Collapsed;
            lb_titulo.Visibility = Visibility.Visible;

        }

        private void aplicaciones_desarrolladas_MouseLeftButtonUp(object sender,
MouseButtonEventArgs e)
        {

            lb_titulo.Visibility = Visibility.Collapsed;
```

```
            imgabrir.Visibility = Visibility.Collapsed;
            border2.Visibility = Visibility.Collapsed;
            frame1.Navigate(new Uri(string.Format(CultureInfo.CurrentCulture,
"/Aplicaciones.xaml?x={0}", Uri.EscapeDataString(AsignarFunciones.user)),
UriKind.Relative));

        }

        private void funciones_responsable_MouseLeftButtonUp(object sender,
MouseButtonEventArgs e)
        {
            lb_titulo.Visibility = Visibility.Collapsed;
            imgabrir.Visibility = Visibility.Collapsed;
            border2.Visibility = Visibility.Collapsed;
            frame1.Navigate(new Uri(string.Format(CultureInfo.CurrentCulture,
"/FuncionesResponsables.xaml?x={0}",
Uri.EscapeDataString(AsignarFunciones.user)), UriKind.Relative));

        }

        private void requerimientos_no_funcionales_MouseLeftButtonUp(object
sender, MouseButtonEventArgs e)
        {
            lb_titulo.Visibility = Visibility.Collapsed;
            imgabrir.Visibility = Visibility.Collapsed;
            border2.Visibility = Visibility.Collapsed;
            frame1.Navigate(new Uri(string.Format(CultureInfo.CurrentCulture,
"/RequerimientosNoFuncionales.xaml?x={0}",
Uri.EscapeDataString(AsignarFunciones.user)), UriKind.Relative));

        }

        private void solicitud_especificacion_MouseLeftButtonUp(object sender,
MouseButtonEventArgs e)
        {
            AsignarFunciones.bandera = 1;
            LG_Controles.MessageBox.MessageBox_SL.showEspera("CARGANDO
INFORMACIÓN...", "CARGANDO");
            lb_titulo.Visibility = Visibility.Collapsed;
            imgabrir.Visibility = Visibility.Collapsed;
            border2.Visibility = Visibility.Collapsed;
            frame1.Navigate(new Uri(string.Format(CultureInfo.CurrentCulture,
"/SolicitudEspecificacion.xaml?x={0}",
Uri.EscapeDataString(AsignarFunciones.user)), UriKind.Relative));


        }

        private void revision_auditoria_MouseLeftButtonUp(object sender,
MouseButtonEventArgs e)
        {
            AsignarFunciones.bandera =2;
            lb_titulo.Visibility = Visibility.Collapsed;
            imgabrir.Visibility = Visibility.Collapsed;
            border2.Visibility = Visibility.Collapsed;
            frame1.Navigate(new Uri(string.Format(CultureInfo.CurrentCulture,
"/SolicitudEspecificacion.xaml?x={0}",
Uri.EscapeDataString(AsignarFunciones.user)), UriKind.Relative));


        }
```

```
        private void revision_sistemas_MouseLeftButtonUp(object sender,
MouseButtonEventArgs e)
        {
            AsignarFunciones.bandera = 3;
            lb_titulo.Visibility = Visibility.Collapsed;
            imgabrir.Visibility = Visibility.Collapsed;
            border2.Visibility = Visibility.Collapsed;
            frame1.Navigate(new Uri(string.Format(CultureInfo.CurrentCulture,
"/SolicitudEspecificacion.xaml?x={0}",
Uri.EscapeDataString(AsignarFunciones.user)), UriKind.Relative));

        }

        private void revision_programador_MouseLeftButtonUp(object sender,
MouseButtonEventArgs e)
        {
            AsignarFunciones.bandera = 4;
            lb_titulo.Visibility = Visibility.Collapsed;
            imgabrir.Visibility = Visibility.Collapsed;
            border2.Visibility = Visibility.Collapsed;
            frame1.Navigate(new Uri(string.Format(CultureInfo.CurrentCulture,
"/SolicitudEspecificacion.xaml?x={0}",
Uri.EscapeDataString(AsignarFunciones.user)), UriKind.Relative));

        }



        private void listReportes_MouseLeftButtonUp(object sender,
MouseButtonEventArgs e)
        {
            lb_titulo.Visibility = Visibility.Collapsed;
            imgabrir.Visibility = Visibility.Collapsed;
            border2.Visibility = Visibility.Collapsed;
            frame1.Navigate(new Uri(string.Format(CultureInfo.CurrentCulture,
"/Reporte_Especificaciones.xaml?x={0}",
Uri.EscapeDataString(AsignarFunciones.user)), UriKind.Relative));

        }
```

2. Creación de una solicitud de Especificación Funcional



MÉTODOS DEL MODULO SOLICITUD.

Nueva Solicitud.

```
   public long NuevaSolicitud(CRDS_Solicitudes solicitud)
        {
            sol.Inicializar();
            try
            {
                long codigo = -1;

                if (sol.Count != 0)
                {
                    CRDS_Solicitudes soli_guardar = new CRDS_Solicitudes();
                    soli_guardar = sol.Find(t => t.sol_id_solicitud ==
solicitud.sol_id_solicitud);
                    if (soli_guardar == null)
                    {
                        codigo = sol.Max(t => t.sol_id_solicitud) + 1;
                    }
                    else
                    {
                        codigo = 0;
                        return codigo;

                    }
                }
                else
                {
                    codigo = 1;
                }
                CRDS_Solicitudes soli = new CRDS_Solicitudes();
                soli.sol_id_solicitud = codigo;
                soli.sol_nombre_proyecto = solicitud.sol_nombre_proyecto;
                soli.sol_area_responsable = solicitud.sol_area_responsable;
                soli.sol_area_solicitante = solicitud.sol_area_solicitante;
                soli.sol_tipo_requerimiento = solicitud.sol_tipo_requerimiento;
```

```
                soli.sol_tipo_desarrollo = solicitud.sol_tipo_desarrollo;
                soli.sol_descripcion_actual = solicitud.sol_descripcion_actual;
                soli.sol_problema = solicitud.sol_problema;
                soli.sol_objetivo = solicitud.sol_objetivo;
                soli.sol_funcionalidad = solicitud.sol_funcionalidad;
                soli.sol_prospectiva = solicitud.sol_prospectiva;
                soli.sol_perspectiva = solicitud.sol_perspectiva;
                soli.sol_observacion_auditoria =
solicitud.sol_observacion_auditoria;
                soli.sol_observacion_sistemas =
solicitud.sol_observacion_sistemas;
                soli.sol_observacion_programador =
solicitud.sol_observacion_programador;
                soli.sol_estado_proceso = solicitud.sol_estado_proceso;
                soli.sol_fecha_aprovada_auditoria =
solicitud.sol_fecha_aprovada_auditoria;
                soli.sol_fecha_aprovada_sistemas =
solicitud.sol_fecha_aprovada_sistemas;
                soli.sol_fecha_aprovada_programador =
solicitud.sol_fecha_aprovada_programador;
                soli.sol_fecha_creacion = solicitud.sol_fecha_creacion;
                soli.sol_fecha_modificacion = solicitud.sol_fecha_modificacion;
                soli.sol_fecha_terminada = solicitud.sol_fecha_terminada;
                soli.sol_usuario_creacion = solicitud.sol_usuario_creacion;
                soli.sol_usuario_auditoria = solicitud.sol_usuario_auditoria;
                soli.sol_usuario_sistemas = solicitud.sol_usuario_sistemas;
                soli.sol_usuario_programador = solicitud.sol_usuario_programador;
                soli.sol_usuario_modificacion =
solicitud.sol_usuario_modificacion;
                soli.sol_fecha_modificacion = solicitud.sol_fecha_modificacion;
                soli.CRDS_AbreviaturasAcronimos =
solicitud.CRDS_AbreviaturasAcronimos;
                soli.CRDS_RequerimientosFuncionales =
solicitud.CRDS_RequerimientosFuncionales;
                soli.CRDS_REL_SolicitudAplicacionDesarrolladas =
solicitud.CRDS_REL_SolicitudAplicacionDesarrolladas;
                soli.CRDS_REL_RequerimientosNoFuncionalesSolicitud =
solicitud.CRDS_REL_RequerimientosNoFuncionalesSolicitud;
                soli.CRDS_REL_UsuariosSolicitud =
solicitud.CRDS_REL_UsuariosSolicitud;
                soli.CRDS_Referencias = solicitud.CRDS_Referencias;
                soli.CRDS_Anexos = solicitud.CRDS_Anexos;
                soli.CRDS_Dependencias = solicitud.CRDS_Dependencias;


                sol.Create(soli);
                return codigo;
            }

            catch (Exception ex)
            {

                System.Diagnostics.EventLog objLog = new
System.Diagnostics.EventLog();
                if (!EventLog.SourceExists("REQUERIMIENTOS"))
                    EventLog.CreateEventSource("REQUERIMIENTOS", "Solicitud");
                objLog.Source = "RECEPCION";
                objLog.WriteEntry("Error al guardar la Solicitud " +
GetMessageError(ex), EventLogEntryType.Warning);

                return -1;
            }
```

```
        }
```

Nueva Solicitud Observaciones

```
public long NuevaSolicitud_Observaciones(CRDS_Solicitudes solicitud_obs)
        {

            sol.Inicializar();
            try
            {
                CRDS_Solicitudes soli = new CRDS_Solicitudes();

                soli = sol.Find(t => t.sol_id_solicitud ==
solicitud_obs.sol_id_solicitud);

                if (soli != null)
                {
                    soli.sol_id_solicitud = solicitud_obs.sol_id_solicitud;
                    soli.sol_observacion_auditoria =
solicitud_obs.sol_observacion_auditoria;
                    soli.sol_observacion_sistemas =
solicitud_obs.sol_observacion_sistemas;
                    soli.sol_observacion_programador =
solicitud_obs.sol_observacion_programador;
                    soli.sol_estado_proceso = solicitud_obs.sol_estado_proceso;


                    foreach (CRDS_Restricciones item in restricciones.Filter(t =>
t.sol_id_solicitud == soli.sol_id_solicitud))
                    {
                        restricciones.Delete(item);
                    }

                    soli.CRDS_RequerimientosFuncionales.Clear();
                    foreach (var item in
solicitud_obs.CRDS_RequerimientosFuncionales)
                    {
                        soli.CRDS_RequerimientosFuncionales.Add(item);

                    }
                    sol.Update(soli);
                    return (soli.sol_id_solicitud);

                }

                else
                {
                    return -1;
                }

            }
            catch (Exception ex)
            {

                System.Diagnostics.EventLog objLog = new
System.Diagnostics.EventLog();
                if (!EventLog.SourceExists("REQUERIMIENTOS"))
```

```
                    EventLog.CreateEventSource("REQUERIMIENTOS", "Solicitud");
                objLog.Source = "RECEPCION";
                objLog.WriteEntry("Error al modificar la Solicitud " +
GetMessageError(ex), EventLogEntryType.Warning);

                return -1;
            }



        }
```

Modificar Solicitud.

```
public long ModificarSolicitud(CRDS_Solicitudes solicitud)
        {

            sol.Inicializar();
            try
            {
                CRDS_Solicitudes soli = new CRDS_Solicitudes();

                soli = sol.Find(t => t.sol_id_solicitud ==
solicitud.sol_id_solicitud);

                if (soli != null)
                {
                    soli.sol_area_responsable = solicitud.sol_area_responsable;
                    soli.sol_funcionalidad = solicitud.sol_funcionalidad;
                    soli.sol_nombre_proyecto = solicitud.sol_nombre_proyecto;
                    soli.sol_perspectiva = solicitud.sol_perspectiva;
                    soli.sol_problema = solicitud.sol_problema;
                    soli.sol_prospectiva = solicitud.sol_prospectiva;
                    soli.sol_tipo_desarrollo = solicitud.sol_tipo_desarrollo;
                    soli.sol_tipo_requerimiento =
solicitud.sol_tipo_requerimiento;
                    soli.sol_usuario_creacion = solicitud.sol_usuario_creacion;
                    soli.sol_id_solicitud = solicitud.sol_id_solicitud;
                    soli.sol_area_solicitante = solicitud.sol_area_solicitante;
                    soli.sol_descripcion_actual =
solicitud.sol_descripcion_actual;
                    soli.sol_estado_proceso = solicitud.sol_estado_proceso;
                    soli.sol_fecha_creacion = solicitud.sol_fecha_creacion;
                    soli.sol_fecha_aprovada_auditoria =
solicitud.sol_fecha_aprovada_auditoria;
                    soli.sol_fecha_aprovada_sistemas =
solicitud.sol_fecha_aprovada_sistemas;
                    soli.sol_fecha_aprovada_programador =
solicitud.sol_fecha_aprovada_programador;
                    soli.sol_fecha_terminada = solicitud.sol_fecha_terminada;
                    soli.sol_usuario_auditoria = solicitud.sol_usuario_auditoria;
                    soli.sol_usuario_sistemas = solicitud.sol_usuario_sistemas;
                    soli.sol_usuario_programador =
solicitud.sol_usuario_programador;
                    soli.sol_objetivo = solicitud.sol_objetivo;
                    soli.sol_observacion_auditoria =
solicitud.sol_observacion_auditoria;
                    soli.sol_observacion_sistemas =
solicitud.sol_observacion_sistemas;
```

```
                    soli.sol_observacion_programador =
solicitud.sol_observacion_programador;
                    soli.sol_usuario_modificacion =
solicitud.sol_usuario_modificacion;
                    soli.sol_fecha_modificacion =
solicitud.sol_fecha_modificacion;

                    soli.CRDS_AbreviaturasAcronimos.Clear();

                    foreach (var item in solicitud.CRDS_AbreviaturasAcronimos)
                    {
                        soli.CRDS_AbreviaturasAcronimos.Add(item);
                    }

                    soli.CRDS_Anexos.Clear();

                    foreach (var item in solicitud.CRDS_Anexos)
                    {
                        soli.CRDS_Anexos.Add(item);
                    }
                    soli.CRDS_Dependencias.Clear();

                    foreach (var item in solicitud.CRDS_Dependencias)
                    {
                        soli.CRDS_Dependencias.Add(item);
                    }

                    soli.CRDS_Referencias.Clear();
                    foreach (var item in solicitud.CRDS_Referencias)
                    {
                        soli.CRDS_Referencias.Add(item);
                    }

                    soli.CRDS_REL_RequerimientosNoFuncionalesSolicitud.Clear();
                    foreach (var item in
solicitud.CRDS_REL_RequerimientosNoFuncionalesSolicitud)
                    {
                        item.CRDS_RequerimientosNoFuncionales = null;


soli.CRDS_REL_RequerimientosNoFuncionalesSolicitud.Add(item);
                    }

                    soli.CRDS_REL_SolicitudAplicacionDesarrolladas.Clear();
                    foreach (var item in
solicitud.CRDS_REL_SolicitudAplicacionDesarrolladas)
                    {
                        soli.CRDS_REL_SolicitudAplicacionDesarrolladas.Add(new
CRDS_REL_SolicitudAplicacionDesarrolladas()
                        {
                            apl_id_aplicacion = item.apl_id_aplicacion,
                            sol_id_solicitud = item.sol_id_solicitud

                        });
                    }



                    soli.CRDS_REL_UsuariosSolicitud.Clear();

                    foreach (var item in solicitud.CRDS_REL_UsuariosSolicitud)
                    {
```

```
                        if (item.CRDS_REL_FuncionUsuarios != null)
                        {
                            item.CRDS_REL_FuncionUsuarios.CRDS_FuncionResponsable
= null;

item.CRDS_REL_FuncionUsuarios.CRDS_UsuariosEspecificaciones = null;


                        }
                        item.CRDS_REL_FuncionUsuarios = null;

                        soli.CRDS_REL_UsuariosSolicitud.Add(item);
                    }



                    foreach (CRDS_Restricciones item in restricciones.Filter(t =>
t.sol_id_solicitud == soli.sol_id_solicitud))
                    {
                        restricciones.Delete(item);
                    }

                    soli.CRDS_RequerimientosFuncionales.Clear();
                    foreach (var item in
solicitud.CRDS_RequerimientosFuncionales)
                    {
                        soli.CRDS_RequerimientosFuncionales.Add(item);

                    }

                    sol.Update(soli);
                    return (soli.sol_id_solicitud);
                }
                else
                {
                    return -1;
                }

            }
            catch (Exception ex)
            {

                System.Diagnostics.EventLog objLog = new
System.Diagnostics.EventLog();
                if (!EventLog.SourceExists("REQUERIMIENTOS"))
                    EventLog.CreateEventSource("REQUERIMIENTOS", "Solicitud");
                objLog.Source = "RECEPCION";
                objLog.WriteEntry("Error al modificar la Solicitud " +
GetMessageError(ex), EventLogEntryType.Warning);

                return -1;
            }

        }}
```

Cargar Solicitud.

```
   public CRDS_Solicitudes CargarSolicitud(long id_sol)
        {

            sol.Inicializar();
            CRDS_Solicitudes solicitud = sol.Find(t => t.sol_id_solicitud ==
id_sol);
            if (solicitud != null)
            {
                CRDS_Solicitudes solic = new CRDS_Solicitudes();


                solic.sol_area_responsable = solicitud.sol_area_responsable;
                solic.sol_funcionalidad = solicitud.sol_funcionalidad;
                solic.sol_nombre_proyecto = solicitud.sol_nombre_proyecto;
                solic.sol_perspectiva = solicitud.sol_perspectiva;
                solic.sol_problema = solicitud.sol_problema;
                solic.sol_prospectiva = solicitud.sol_prospectiva;
                solic.sol_tipo_desarrollo = solicitud.sol_tipo_desarrollo;
                solic.sol_tipo_requerimiento = solicitud.sol_tipo_requerimiento;
                solic.sol_usuario_creacion = solicitud.sol_usuario_creacion;
                solic.sol_usuario_auditoria = solicitud.sol_usuario_auditoria;
                solic.sol_usuario_sistemas = solicitud.sol_usuario_sistemas;
                solic.sol_usuario_programador =
solicitud.sol_usuario_programador;
                solic.sol_id_solicitud = solicitud.sol_id_solicitud;
                solic.sol_area_solicitante = solicitud.sol_area_solicitante;
                solic.sol_descripcion_actual = solicitud.sol_descripcion_actual;
                solic.sol_estado_proceso = solicitud.sol_estado_proceso;
                solic.sol_fecha_creacion = solicitud.sol_fecha_creacion;
                solic.sol_fecha_aprovada_auditoria =
solicitud.sol_fecha_aprovada_auditoria;
                solic.sol_fecha_aprovada_sistemas =
solicitud.sol_fecha_aprovada_sistemas;
                solic.sol_fecha_aprovada_programador =
solicitud.sol_fecha_aprovada_programador;
                solic.sol_fecha_terminada = solicitud.sol_fecha_terminada;
                solic.sol_objetivo = solicitud.sol_objetivo;
                solic.sol_usuario_modificacion =
solicitud.sol_usuario_modificacion;
                solic.sol_fecha_modificacion = solicitud.sol_fecha_modificacion;
                solic.sol_observacion_auditoria =
solicitud.sol_observacion_auditoria;
                solic.sol_observacion_sistemas =
solicitud.sol_observacion_sistemas;
                solic.sol_observacion_programador =
solicitud.sol_observacion_programador;



                solic.CRDS_AbreviaturasAcronimos = (from t in
solicitud.CRDS_AbreviaturasAcronimos.ToList()
                                                    select new
CRDS_AbreviaturasAcronimos()
                                                    {
                                                        abr_descripcion =
t.abr_descripcion,
                                                        abr_id_abreviatura =
t.abr_id_abreviatura,
                                                        abr_nombre =
t.abr_nombre,
```

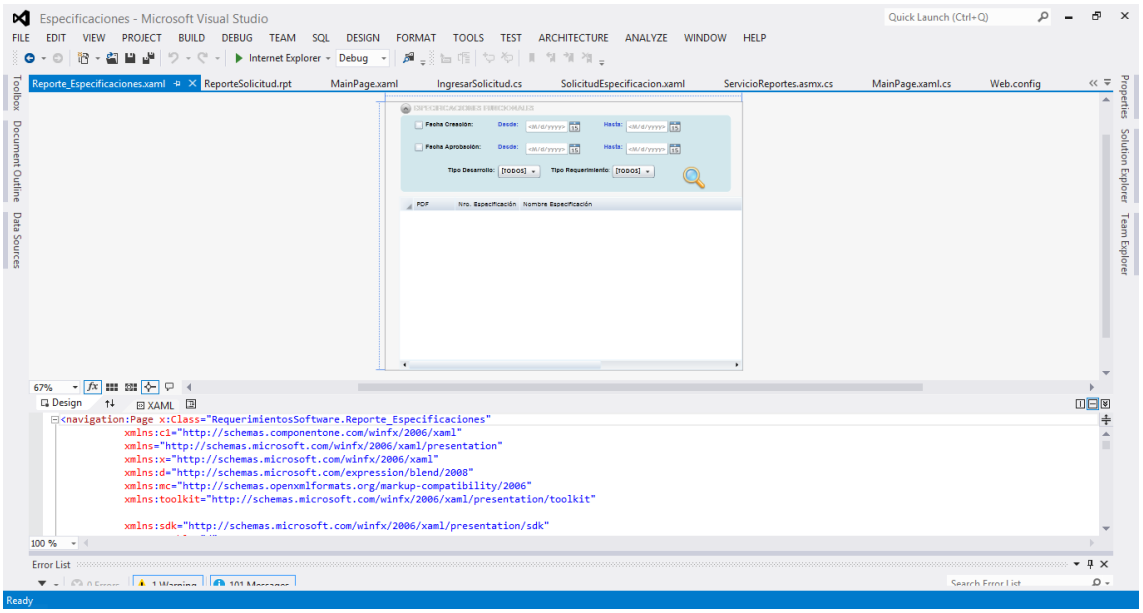
```
                                                    abr_posicion_final =
t.abr_posicion_final,
                                                    abr_posicion_inicial =
t.abr_posicion_inicial,
                                                    abr_tamanio_caracter =
t.abr_tamanio_caracter,
                                                    abr_tipo = t.abr_tipo
                                                }).ToList();

                solic.CRDS_Anexos = (from a in solicitud.CRDS_Anexos.ToList()
                                     select new CRDS_Anexos()
                                     {
                                         anx_archivo_cadena =
a.anx_archivo_cadena,
                                         anx_descripcion = a.anx_descripcion,
                                         anx_fecha_creacion =
a.anx_fecha_creacion,
                                         anx_id_anexo = a.anx_id_anexo,
                                         anx_nombre = a.anx_nombre,
                                         anx_tipo = a.anx_tipo,
                                         anx_ubicacion = a.anx_ubicacion,
                                         anx_usuario_creacion =
a.anx_usuario_creacion
                                     }).ToList();


                solic.CRDS_Dependencias = (from d in
solicitud.CRDS_Dependencias.ToList()
                                           select new CRDS_Dependencias()
                                           {
                                               dep_descripcion =
d.dep_descripcion,
                                               sol_id_solicitud =
d.sol_id_solicitud,
                                           }).ToList();


                solic.CRDS_Referencias = (from r in
solicitud.CRDS_Referencias.ToList()
                                          select new CRDS_Referencias()
                                          {
                                              ref_descripcion =
r.ref_descripcion,
                                              ref_fecha_creacion =
r.ref_fecha_creacion,
                                              ref_id_referencia =
r.ref_id_referencia,
                                              ref_tipo = r.ref_tipo,
                                              ref_usuario_creacion =
r.ref_usuario_creacion,

                                          }).ToList();

                solic.CRDS_REL_RequerimientosNoFuncionalesSolicitud = (from rnf
in solicitud.CRDS_REL_RequerimientosNoFuncionalesSolicitud.ToList()
                                                                       select new
CRDS_REL_RequerimientosNoFuncionalesSolicitud()
                                                                       {

rqnf_id_no_funcionales = rnf.rqnf_id_no_funcionales,

sol_id_solicitud = rnf.sol_id_solicitud,
```

```
CRDS_RequerimientosNoFuncionales = new CRDS_RequerimientosNoFuncionales()
                                                                    {

rqnf_id_no_funcionales =
rnf.CRDS_RequerimientosNoFuncionales.rqnf_id_no_funcionales,

rqnf_descripcion = rnf.CRDS_RequerimientosNoFuncionales.rqnf_descripcion,

rqnf_usuario_creacion =
rnf.CRDS_RequerimientosNoFuncionales.rqnf_usuario_creacion,

rqnf_usuario_modificacion =
rnf.CRDS_RequerimientosNoFuncionales.rqnf_usuario_modificacion,
                                                                    }


}).ToList();
                solic.CRDS_REL_SolicitudAplicacionDesarrolladas = (from sa in
solicitud.CRDS_REL_SolicitudAplicacionDesarrolladas.ToList()
                                                                     select new
CRDS_REL_SolicitudAplicacionDesarrolladas()
                                                                    {

apl_id_aplicacion = sa.apl_id_aplicacion,

sol_id_solicitud = sa.sol_id_solicitud,

CRDS_AplicacionesDesarrolladas = new CRDS_AplicacionesDesarrolladas()
                                                                        {

apl_id_aplicacion = sa.CRDS_AplicacionesDesarrolladas.apl_id_aplicacion,

apl_nombre = sa.CRDS_AplicacionesDesarrolladas.apl_nombre,

apl_descripcion=sa.CRDS_AplicacionesDesarrolladas.apl_descripcion,

apl_programador=sa.CRDS_AplicacionesDesarrolladas.apl_programador,

                                                                        }


                                                                    }).ToList();
                solic.CRDS_REL_UsuariosSolicitud = (from us in
solicitud.CRDS_REL_UsuariosSolicitud.ToList()
                                                     select new
CRDS_REL_UsuariosSolicitud()
                                                     {
                                                         fun_id_funcion =

us.fun_id_funcion,

                                                         sol_id_solicitud =

us.sol_id_solicitud,

                                                         usu_nombre_corto =

us.usu_nombre_corto,

                                                         CRDS_REL_FuncionUsuarios

= new CRDS_REL_FuncionUsuarios()

                                                         {
                                                             fun_id_funcion =

us.CRDS_REL_FuncionUsuarios.fun_id_funcion,

                                                             usu_nombre_corto =

us.CRDS_REL_FuncionUsuarios.usu_nombre_corto,
```

```
CRDS_FuncionResponsable = new CRDS_FuncionResponsable()
                                                                {
                                                                    fun_nombre =
us.CRDS_REL_FuncionUsuarios.CRDS_FuncionResponsable.fun_nombre,
                                                                },

CRDS_UsuariosEspecificaciones = new CRDS_UsuariosEspecificaciones()
                                                                {
                                                                    usu_apellido =
us.CRDS_REL_FuncionUsuarios.CRDS_UsuariosEspecificaciones.usu_apellido,
                                                                    usu_nombre =
us.CRDS_REL_FuncionUsuarios.CRDS_UsuariosEspecificaciones.usu_nombre,
                                                                    usu_email =
us.CRDS_REL_FuncionUsuarios.CRDS_UsuariosEspecificaciones.usu_email,

                                                                },


                                                            }


                                                        }).ToList();

                solic.CRDS_RequerimientosFuncionales = (from rf in
solicitud.CRDS_RequerimientosFuncionales.ToList()
                                                         select new
CRDS_RequerimientosFuncionales()
                                                         {
                                                             rqf_descripcion =
rf.rqf_descripcion,
                                                             rqf_estado =
rf.rqf_estado,
                                                             rqf_id_funcionales =
rf.rqf_id_funcionales,
                                                             rqf_imagen =
rf.rqf_imagen,
                                                             rqf_nombre =
rf.rqf_nombre,
                                                             rqf_objetivo =
rf.rqf_objetivo,
rqf_observacion_auditoria = rf.rqf_observacion_auditoria,
rqf_observacion_sistemas = rf.rqf_observacion_sistemas,
rqf_observacion_programador = rf.rqf_observacion_programador,
                                                             CRDS_Restricciones =
(from res in rf.CRDS_Restricciones
select new CRDS_Restricciones()
{
res_descripcion = res.res_descripcion,
res_estado = res.res_estado,
res_id_restriccion = res.res_id_restriccion,
res_tipo = res.res_tipo
```

```
}).ToList()
                                                        }).ToList();

                return solic;
            }
            else
                return null;


        }
```

METODOS REPORTES



Exportar Pdf.

```
public byte[] ExportarPdf(string codigosolicitud)
        {
            try
            {
                string PathReporte = Server.MapPath("ReporteSolicitud.rpt");
                CrystalDecisions.CrystalReports.Engine.ReportDocument rptDocument
= new CrystalDecisions.CrystalReports.Engine.ReportDocument();
                rptDocument.Load(PathReporte);

                //generador.GeneraReporte();
                string nombre = "Solicitud_" + codigosolicitud + ".pdf";
                //nombre = nombre.Replace('/', '_');
                //string url = "\\Reportes";

                //string ruta = "\\CargaArchivos\\Exportados\\" + nombre;
                string ruta = "\\Pdf\\" + nombre;
                string apppath = Server.MapPath("~");
```

```
                string nombre_archivo = apppath + ruta;
                //File.WriteAllText("C://compartido/errores.txt","Ruta: "+
nombre_archivo);

                rptDocument.DataSourceConnections[0].SetConnection("(local)",
"RequerimientosSoftware", false);
                rptDocument.SetDatabaseLogon("sa", "sql");
                rptDocument.SetParameterValue(0, codigosolicitud);
                rptDocument.SetParameterValue(1, codigosolicitud);
                rptDocument.SetParameterValue(2, codigosolicitud);
                rptDocument.SetParameterValue(3, codigosolicitud);
                rptDocument.SetParameterValue(4, codigosolicitud);
                rptDocument.SetParameterValue(5, codigosolicitud);
                rptDocument.SetParameterValue(6, codigosolicitud);
                //rptDocument.SetParameterValue("solicitud", codigosolicitud);
                //rptDocument.Subreports[0].SetParameterValue("solicitud",
codigosolicitud);
                //rptDocument.Subreports[1].SetParameterValue("solicitud",
codigosolicitud);
                //rptDocument.Subreports[2].SetParameterValue("solicitud",
codigosolicitud);
                //rptDocument.Subreports[3].SetParameterValue("solicitud",
codigosolicitud);
                //rptDocument.Subreports[4].SetParameterValue("solicitud",
codigosolicitud);
                //rptDocument.Subreports[5].SetParameterValue("solicitud",
codigosolicitud);
                //rptDocument.Subreports[6].SetParameterValue("solicitud",
codigosolicitud);



                //CrystalDecisions.CrystalReports.Engine.ReportDocument rdoc =
new ReportDocument();
                //rdoc = generador.Reporte;

rptDocument.ExportToDisk(CrystalDecisions.Shared.ExportFormatType.PortableDocForm
at, nombre_archivo);
                rptDocument.Dispose();
                rptDocument.Close();

                //FileStream fil=
                byte[] f = File.ReadAllBytes(nombre_archivo);
                return f;
            }
            catch (Exception ex)
            {


                System.Diagnostics.EventLog objLog = new
System.Diagnostics.EventLog();
                if (!EventLog.SourceExists("REQUERIMIENTOS"))
                    EventLog.CreateEventSource("REQUERIMIENTOS", "Solicitud");
                objLog.Source = "RECEPCION";
                objLog.WriteEntry("Error al modificar la Solicitud " +
GetMessageError(ex), EventLogEntryType.Warning);
                return null;
            }


        }
```

Enviar Correo

```
        [WebMethod]
        public string EnviarCorreo(string correo_origen, string correo_destino,
string correo_copia, string saludo, string mensaje, string usuario_creacion,
string ruta)
        {


            GeneraGuiaEasyTini.ADEnvioCorreo envio_correo = new ADEnvioCorreo();

            string respuesta;


            try
            {


                ServicioSOA.Service1Client servicio = new
ServicioSOA.Service1Client();
                List<ServicioSOA.Usuarios> lts_usuario_externo =
servicio.GetNombreUsuario(usuario_creacion);

                foreach (var item in lts_usuario_externo)
                {
                    if (correo_copia == "")
                        correo_copia += item.Email;
                    else

                        correo_copia += ";" + item.Email;
                }

                string nombre = "\\Pdf\\" + ruta;
                string apppath = Server.MapPath("~");
                string nombre_archivo = apppath + nombre;
                ruta = nombre_archivo;

                if (correo_destino == "")
                    correo_destino = correo_copia;
                if (correo_destino != "")
                {

                    respuesta = envio_correo.EnviarCorreo(correo_origen,
correo_destino, correo_copia, saludo, mensaje, nombre_archivo);
                }
                else
                    respuesta = envio_correo.EnviarCorreo(correo_origen,
correo_copia, correo_copia, saludo, mensaje, nombre_archivo);



                System.Diagnostics.EventLog objLog = new
System.Diagnostics.EventLog();
                if (!EventLog.SourceExists("REQUERIMIENTOS"))
                    EventLog.CreateEventSource("REQUERIMIENTOS", "Solicitud");
                objLog.Source = "RECEPCION";
                objLog.WriteEntry("OK", EventLogEntryType.Warning);


            }
            catch (Exception ex)
```

```
            {


                System.Diagnostics.EventLog objLog = new
System.Diagnostics.EventLog();
                if (!EventLog.SourceExists("REQUERIMIENTOS"))
                    EventLog.CreateEventSource("REQUERIMIENTOS", "Solicitud");
                objLog.Source = "RECEPCION";
                objLog.WriteEntry("Error al modificar la Solicitud " +
GetMessageError(ex), EventLogEntryType.Warning);

                respuesta = ex.Message + " " + ex.StackTrace;

            }
            return respuesta;

        }
```