UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERIA

“Módulo generador de reportes utilizando MVC .NET y Reporting Services para una dependencia de gobierno.”

INFORME DE TRABAJO PROFESIONAL

PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A:

Julio Cesar Rodriguez Garcia

ASESOR: M. I. Jorge Valeriano Assem

MÉXICO D.F., Febrero 2015

Índice

CAPÍTULO 1
INTRODUCCIÓN

1.1 Supuestos y aclaraciones

El siguiente informe presenta conceptos técnicos y no está dirigido a un público en general, por lo que se espera que el lector este familiarizado con conceptos de arquitectura de software, programación, bases de datos y el IDE Visual Studio. También se recomienda tener nociones sobre Reporting Services.

Más adelante se hara una descripción de las tecnologías utilizadas en mi participación en el proyecto pero no de manera profunda, ya que sale del propósito del informe.

Este informe contiene las actividades realizadas por un profesionista recién egresado de la carrera de Ingeniería en Computación en el ámbito profesional, laborando en un equipo de trabajo multidisciplinario de la UNAM y desarrollando en 4 Sistemas Web para una institución gubernamental.

Las imágenes presentadas con segmentos de código y configuraciones de Reporting Services harán siempre referencia al módulo de reportes en el cual colaboré.

Como fines prácticos de este informe sólo serán incluidas las imágenes de código y configuración más relevantes.

Se describirá a grandes rasgos el proyecto en general, el cual consta de cuatro Sistemas Web, pero se profundizará en el módulo de reportes.

1.2 Objetivo

Crear herramientas web de calidad, que permitan generar diferentes tipos de reportes de manera sistemática y ágil, para monitorear la actividad de una dependencia de gobierno, principalmente la prestación de servicios. Estos reportes concentraran información sobre la operación diaria, desde los anexos técnicos y vehículos montados hasta la validación y seguimiento del cobro a los clientes, todo esto permitirá a la dependencia establecer tramos de control en todas las áreas involucradas, sentando así las bases del Business Intelligence (en la fase de análisis de información) las cuales propiciaran una correcta toma de decisiones para mejorar el rumbo de la institución.

1.3 Justificación del proyecto en general

Hoy en día el cobro por los servicios prestados por parte de la dependencia del gobierno es un proceso con un bajo grado de automatización, el cual consume considerable tiempo, derivado de múltiples validaciones e interacciones con el cliente y el área sustantiva de la Institución, generando retrasos importantes y diversos errores en la consolidación y validación de la información.

El proyecto consta de cuatro Sitios Web, cada uno con características que cubren necesidades específicas del cliente, en cuanto al trámite de prestación de servicios. Así mismo cada sitio permite generar reportes (Módulo de reportes) de acuerdo a su alcance funcional. Para el monitoreo de la actividad diaria de la institución

Necesidades identificadas

- Automatización del proceso de cobranza por los servicios prestados.
- Validación de asistencia diaria de los integrantes de la institución.
- Estandarización en procesos de control y validación de asistencia de los Despliegues Operativos de la Institución.
- Reducción en tiempos de cobro y precisión en control de ingresos.

1.2 Estructura General del Informe

El informe busca ser los más claro, concreto y conciso posible, esquematizando mi participación en el proyecto, por lo cual me enfocaré exclusivamente al Módulo de Reportes.

El informe técnico está dividido en 3 grandes fases.

1. Introducción al equipo de trabajo y a las tecnologías usadas en el proyecto, mismas que son heredadas al módulo de Reportes. Profundizaré en Reporting Services.
2. Las características del Framework Reporting Services a las necesidades del cliente, y se explicará como se realizó la integración y comunicación entre los Sitios Web, Reporting Services y la Base de Datos de manera detallada. Esta será la parte más extensa.
3. Se dividirán los reportes por funcionalidad y se presentaran los resultados más relevantes en cuanto al diseño, configuración, código fuente, peculiaridades y utilidad de cada reporte.
4. Posteriormente se presentaran las conclusiones pertinentes.

CAPÍTULO 2
TECNOLOGÍAS UTILIZADAS

Dentro del equipo de trabajo de la UNAM, mi puesto es de Desarrollador .Net

El enfoque principal de esta área es el de analizar, construir y desarrollar una solución informática de calidad, con el objetivo de mejorar los procesos que se realizan a diario en el interior de la dependencia de gobierno y satisfacer de manera óptima sus requerimientos.

Así como también, es deber permanente de esta unidad analizar las diversas tecnologías en el ámbito del desarrollo de aplicaciones que puedan resultar útiles a los procesos llevados a cabo por el sistema.

Entre las tecnologías utilizadas se incluyen:

Arquitectura basada en la programación en capas, en este caso MVC, la cual se implementará utilizando el framework MVC .NET 4.0

Un lenguaje de programación robusto y capaz de atender a las necesidades del usuario. En este caso se optó por utilizar C# .NET.

Un modelo para mapear objetos de la Base de Datos (BD), que permita asegurar la integridad y seguridad de los mismos. Para esta tarea se utilizó el Patrón repositorio y unidad de trabajo. También para facilitar el acceso a la información en la BD se usó Entity Framework.

Servicios Web, implementados mediante WCF (Windows Communication Fundation) para el intercambio de información entre máquinas.

Una parte fundamental del sistema, es la generación de reportes, la cual permite tener una visión muy amplia sobre la situación actual de la institución, para ello se utilizó el framework Reporting Services.

Para la interfaz de usuario se utilizó JQuery, Bootstrap y Javascript, además de HTML Razor.

Se necesitó un IDE capaz de soportar todas estas tecnologías, Visual Studio 2010.

Mi participación en el sistema fue la creación de reportes. Lo cual involucra todas las tecnologías antes mencionadas, pero principalmente la última.

A continuación se describiran brevemente cada una de las tecnologías antes mencionadas.

2.1 MVC .NET

El modelo Vista-Controlador es un patrón de diseño que permite al programador estructurar de una manera organizada el proyecto. Lo que se busca es separar el acceso a la base de datos, la lógica de programación y la presentación al usuario. En la figura 1 se ejemplifica el proceso.

- El **Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.
- El **Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta de 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.
- La **Vista**: Presenta el 'modelo' (información y *lógica de negocio*) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

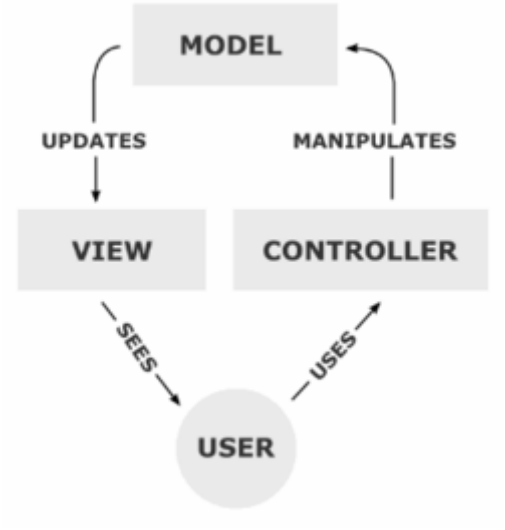


Figura 1. Esquema del funcionamiento de MVC

MVC .NET es la implementación que hizo Microsoft de esta arquitectura, puede ser usado a través del IDE Visual Studio. Y permite la separación en diferentes carpetas, del Controlador, el Modelo y la Vista, tal y como se muestra en la Figura 2.

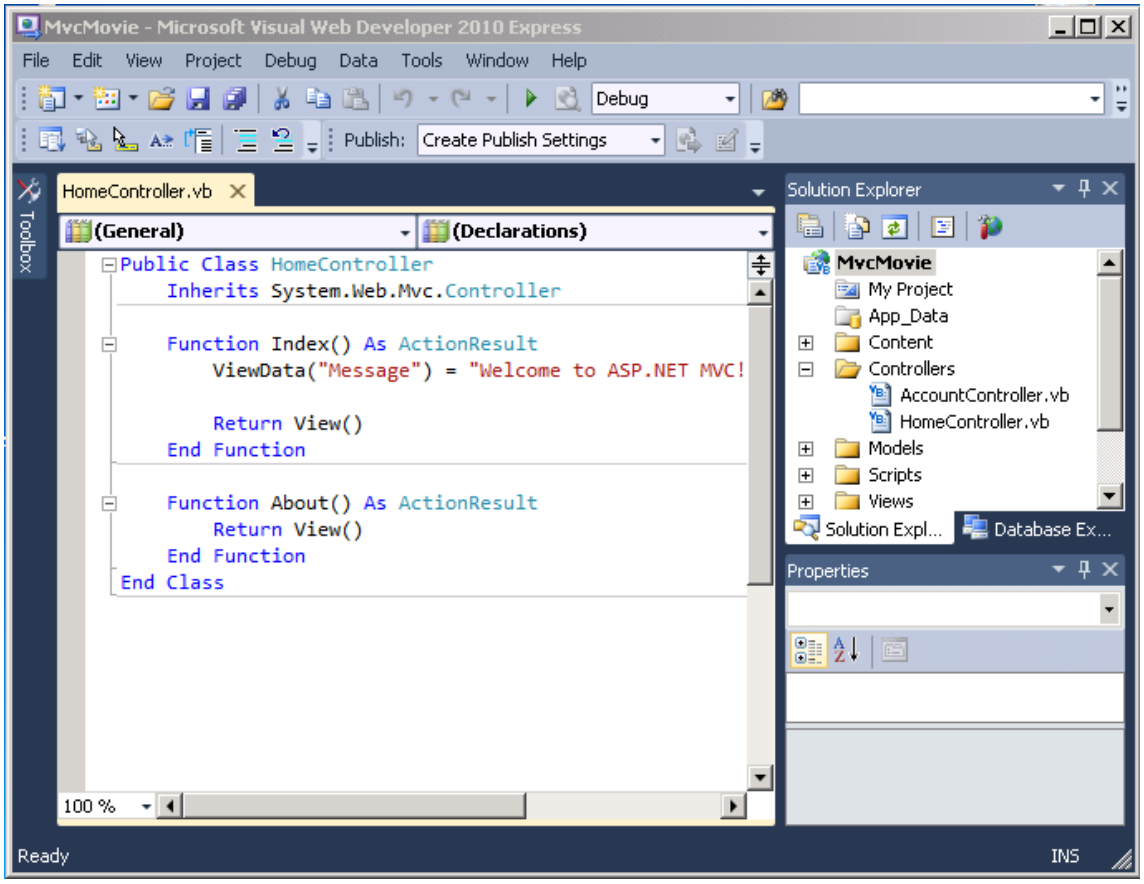


Figura 2. Estructura de MVC en Visual Studio 2010

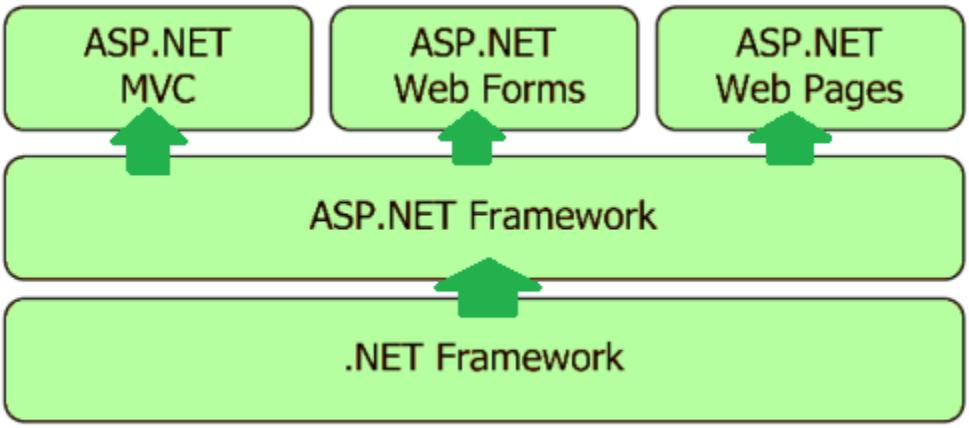


Figura 3. Jerarquía del framework .NET

Como se ve en la figura 3, **ASP.NET MVC** fue construido sobre **Microsoft ASP.NET Framework**, la cual está a su vez montada sobre Microsoft .NET Framework.

ASP.NET Web Forms fue concebido como un reemplazo y mejora para de Microsoft Active Server Pages (ASP) y pronto se convirtió en la metodología favorita para los desarrolladores a la hora de plantear una aplicación web. Sin embargo, a pesar que ASP.NET MVC salió a la luz un poco más adelante que ASP.NET Web Forms esto no significa que es un reemplazo para ella, ya que son dos tecnologías que emplean patrones totalmente diferentes.

Por lo cual ASP.NET MVC debe considerarse sólo una forma alternativa de hacer un sitio web de ASP.NET.

Como herramienta inicial para el proyecto se utilizó el entorno de desarrollo de Microsoft Visual Studio 2010 y el framework **ASP.NET MVC 4** que implementa el patrón modelo-vista-controlador.

Características de MVC 4 .NET

Entre las novedades que incluye MVC 4 en relación a su predecesor tenemos:

· **Entity Framework 4.3,** cuya principal ventaja es la migración de bases de datos usando únicamente código, preservando la información de la base de datos.

· Las aplicaciones se cargan de forma más rápida debido a que se disminuye el tiempo y el tamaño de peticiones web.

· ***Cache Busting*** que permite un mejor manejo de archivos.

· ***"Wep API's"***: Se incluye soporte para su creación, permitiendo crear servicios HTTP y API's que pueden ser llamados directamente desde el código desde una gran cantidad de clientes. Ideal para crear Servicios REST.

· **Web para móviles**: ASP.NET MVC 4 incluye soporte para construir aplicaciones y sitios web para móviles, permitiendo crear de forma mucho más sencilla experiencias dirigidas a teléfonos y tabletas.

· **Mejoras en Razor**: ASP.NET MVC 4 incluye la versión 2 del motor Razor, éste incluye mejoras en la creación de plantillas, más limpias y consistentes, soporte en la resolución de referencias URL, y en la muestra selectiva de atributos HTML.

2.2 C# .NET

Es un lenguaje orientado a objetos desarrollado por Microsoft, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método Main que es el punto de entrada de la aplicación, se encapsulan dentro de definiciones de clase. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. Los métodos que reemplazan a los métodos virtuales en una clase primaria requieren la palabra clave override como medio para evitar redefiniciones accidentales.

Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de software a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen las siguientes:

Firmas de métodos encapsulados denominadas delegados, que habilitan notificaciones de eventos con seguridad de tipos.

Propiedades, que actúan como descriptores de acceso para variables miembro privadas.

Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.

Comentarios en línea de documentación XML.

2.3 Patrón Repositorio y Unidad de trabajo

Los patrones repositorio y unidad de trabajo proporcionan una forma limpia para acceder a los datos mediante un ORM (Object-Relational mapping), mantener toda la lógica de acceso a datos en una ubicación central y al mismo tiempo mantener la aplicación fácil de probar.

La unidad de trabajo tiene como objetivo tratar como una unidad todos aquellos objetos nuevos, modificados o eliminados con respecto de una fuente de datos. Este patrón es el encargado de hacer el seguimiento de todos aquellos objetos que son nuevos, y que por lo tanto deben guardarse en la base de datos, de todos los objetos que han sido modificados y que deben actualizarse en la base de datos y de todos los que han sido borrados y deben quitarse de la base de datos.

El patrón repositorio está íntimamente relacionado con el acceso a datos y nos permite tener una abstracción de la implementación de acceso a datos en nuestras aplicaciones, de modo que nuestra lógica de negocio no conozca ni esté acoplada a la fuente de datos. En pocas palabras, esto quiere

decir que el repositorio actúa como un intermediario entre la lógica de negocio y la lógica de acceso a datos. En la figura 4 se ejemplifica este modelo.

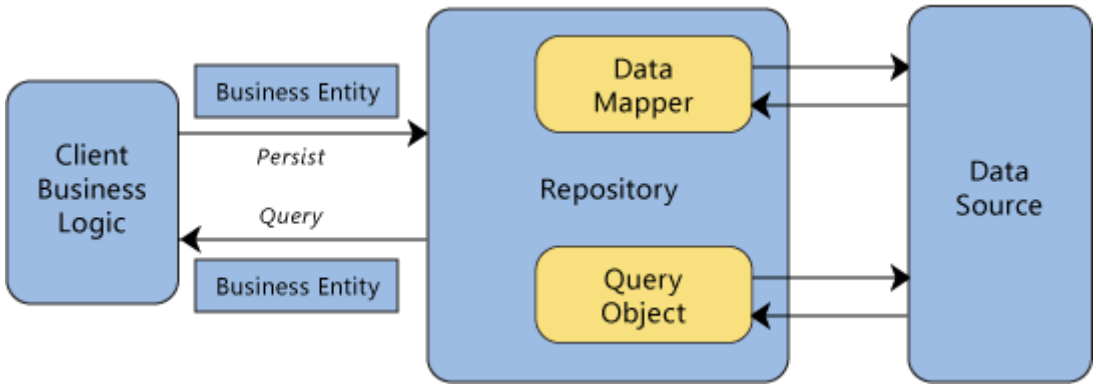


Figura 4. Patrón repositorio y unidad de trabajo

2.4 Web Services.

El término Web Services describe una **forma estandarizada de integrar** aplicaciones WEB mediante el uso de XML, SOAP, WSDL y UDDI sobre los protocolos de la Internet. XML es usado para describir los datos, SOAP se ocupa para la transferencia de los datos, WSDL se emplea para describir los servicios disponibles y UDDI se ocupa para conocer cuáles son los servicios disponibles. EL uso de estos Web Services nos permitió crear interacciones entre el cliente y el servidor mucho más seguras ya que el usuario nunca interactúa directamente con la información en la base de datos, tal como se muestra en la Figura 5.

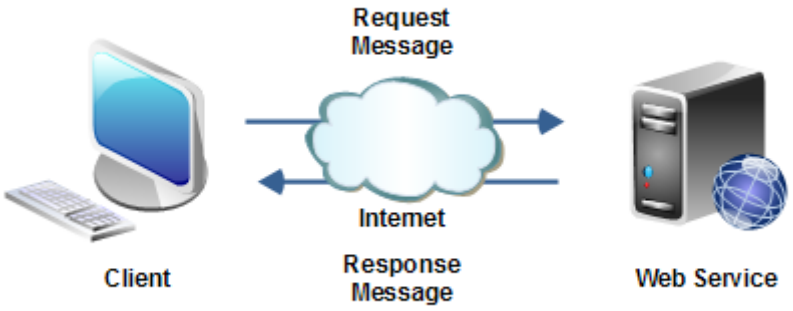


Figura 5. Arquitectura cliente-servidor

A diferencia de los modelos Cliente/Servidor, tales como un servidor de páginas Web, los Web Services no proveen al usuario una interfaz gráfica (GUI). En vez de ello, los Web Services comparten la **lógica del negocio**, **los datos y los procesos,** en la figura 6 se esquematiza esta interacción.

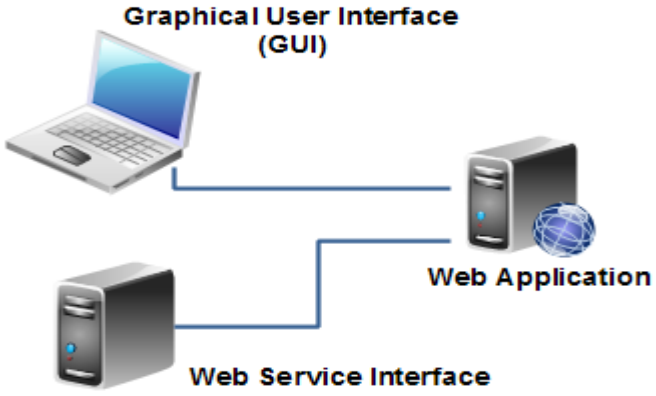


Figura 6. Arquitectura de Web Services

Los Web Services permiten a distintas aplicaciones, de diferentes orígenes, comunicarse entre ellos **sin necesidad de escribir programas costosos**, esto porque la comunicación se hace con XML. Los Web Services no están ligados a ningún sistema operativo o lenguaje de programación. Por ejemplo, un programa escrito en Java puede conversar con otro escrito en Pearl; Las aplicaciones Windows pueden conversar con aplicaciones Unix. Por otra parte, los Web Services no necesitan usar browsers (Explorer) ni el lenguaje de especificación HTML.

3.5 Reporting Services

Reporting Services es una plataforma que permite definir, administrar y distribuir distintos formatos de reportes dentro de una organización o a través de múltiples organizaciones.

Este servicio es una extensión a las capacidades Business Intelligence que nos provee de herramientas para almacenar información (Report Server), herramientas para crear reportes (Report Designer) y herramientas para administrar reportes (Report Manager).

Reporting Services es una plataforma de reportes basada en servidores, la misma que puede ser empleada para crear y administrar reportes tabulares, de matrices, gráficos y de libre formato.

Siempre que se piensa en elaborar reportes, tenemos que distinguir claramente algunos aspectos:

1. Definición del reporte, momento en que el autor del reporte define los datos y la manera de presentación de estos. En esta etapa normalmente hay que definir conexiones a los distintos orígenes de datos para ver de dónde obtener los resultados que debe reflejar el reporte.

2. Administración del reporte, está referido al hecho que en las organizaciones actuales tenemos distintas categorías de usuarios, como por ejemplo, los gerentes, los usuarios de servicio al cliente, etc. Por lo tanto, es importante definir quiénes serán los usuarios del reporte, para ello hay que publicar los reportes.
3. Entrega del reporte, es muy común en las organizaciones que muchos reportes sean requeridos de manera periódica, por ejemplo el reporte de ventas diarias debe estar en la oficina del Gerente de Ventas todas las tardes a las 5 pm., o un reporte de inventario todos los fines de semana, podríamos entonces aprovechar distintos servicios como el de mensajería para que estos reportes lleguen a los usuarios requeridos.

Las tres acciones, mencionadas anteriormente, conforman lo que se denomina "El Ciclo de Vida de un reporte", SQL Server Reporting Services nos otorga todas las facilidades necesarias para que podamos cubrir cada una de los aspectos asociados a la creación, administración y distribución de los reportes.

Ventajas de Reporting Services:

· Una de las principales, es que se cuenta con una interface Web para lo que es la administración de los reportes, desde esta interface se puede determinar en qué formato debe llegar el reporte, es decir, podemos decidir que el reporte llegue a una de las gerencias en formato PDF. Esto es particularmente útil para el cliente.

· Cuenta con un lenguaje de especificación estándar denominado Report Definition Language o simplemente RDL, el cual es un lenguaje de formato XML, que se encarga de definir el reporte. Esto nos da flexibilidad para adaptarnos a las necesidades del usuario.

· Otra de las grandes características de Reporting Services, es que puede distribuir el reporte en distintos formatos, como hojas de excel, documentos pdf, texto, XML, etc.

· La arquitectura de Reporting Services, permite a los desarrolladores preparar aplicaciones personalizadas que accedan a los reportes a través de una API que está expuesta como un Web Service. En la figura 7 observamos esta arquitectura.

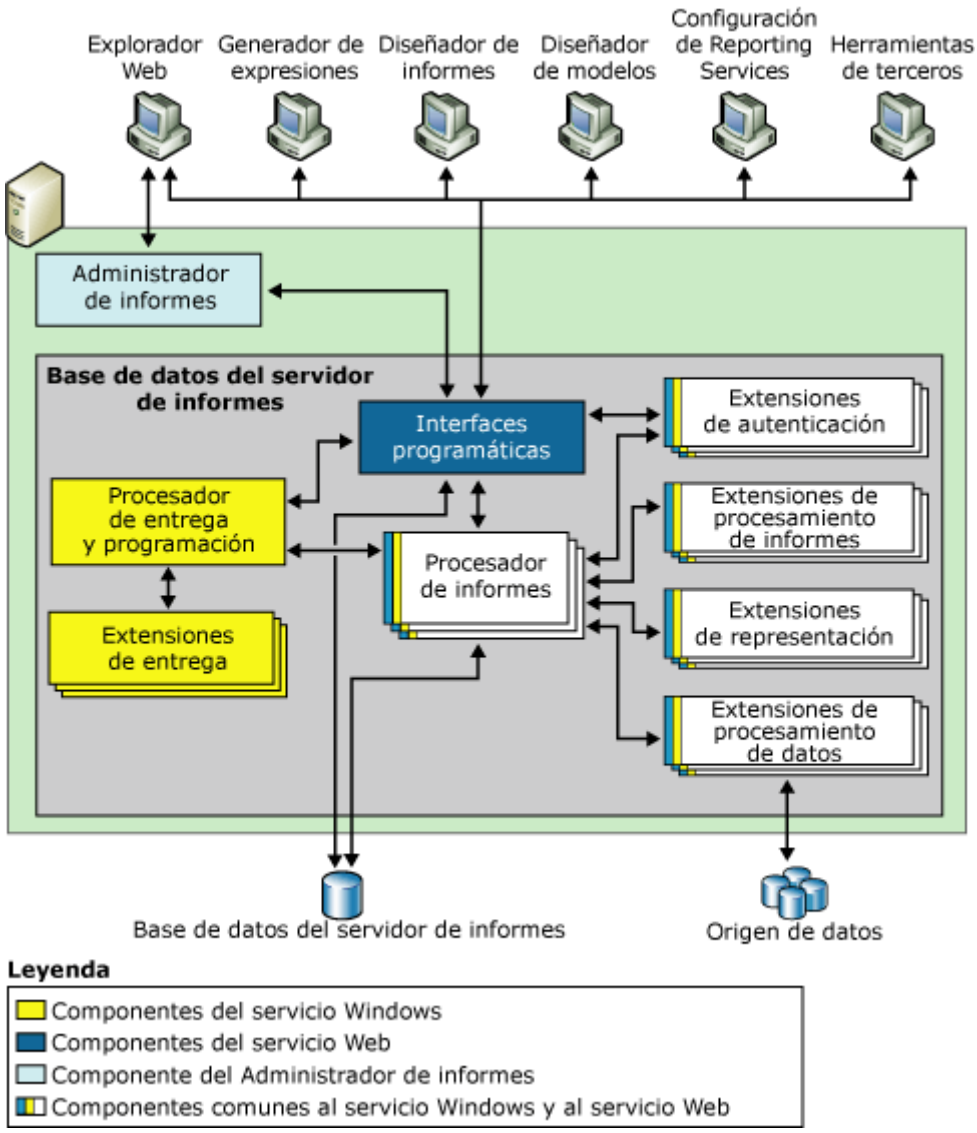


Figura 7. Arquitectura de Reporting Services

Reporting Services está divido en:

- Servidor de informes. Es el componente principal de RS, El servidor de informes se implementa como un servicio de Microsoft Windows y como un servicio Web. Se encarga de generar los reportes a través de los servicios Web y de la seguridad de los mismos.
- Administrador de informes. Es una herramienta de administración de los informes. Se accede a esta a través del explorador Web (IE), que lleva por debajo un portal Share Point o desde el MSSS Management Studio. A través de este podemos asignar permisos, crear carpetas, ver informes, crear nuevos informes con Report Builder, crear suscripciones(o instantáneas) etc.
- Base de datos del servidor de informes. Es una base de datos SQL Server 2005 donde almacena toda la información relacionada con los informes, con la seguridad, suscripciones, instantáneas y demás extensiones del mismo.

- Herramienta de configuración de Reporting Services. Esta es la encargada de configurar el servidor de informes, entre ellos, el estado del servidor, el directorio virtual del servidor de informes y administrador de informes, identidad del servicio de windows, identidad del servicio web, instalación de base de datos, claves de cifrado, inicialización, configuración de correo electrónico y cuenta de ejecución.

Creación de un reporte

Para crear un informe podremos hacerlo de diferentes formas:

Con Visual Studio .NET 2008. Con Visual Studio añadiendo un nuevo proyecto de servidor de informes. A este le añadimos una fuente de datos o DataSet y un informe nuevo.

Report Builder (RB). Para crear un informe con RB deberemos tener creados alguna fuente de datos o DataSet para trabajar con él. Lo destacable de la herramienta, que es para usuarios finales, donde nosotros creamos los roles, le damos los DataSets y ellos realizan el informe final.

Report Definition Lenguage.

Todos los reportes se generan en XML, lo cual da muchas posibilidades a la hora de trabajar con él, ya que podemos programarlo a mano desde código o pasarlo a través de firewalls al ser texto plano.

El Report Definition Lenguage (RDL) es un estándar para la generación del código XML del reporte, está constituido por una serie de etiquetas que podemos clasificar de la siguiente forma:

- Cabecera. En la cabecera podemos encontrar la versión de XML, la codificación.
- Report. Contiene la dirección de la definición del RDL y todo el contenido del informe va entre sus etiquetas.
    - DataSources. Aquí muestra la fuente de datos, en ella podemos ver, el proveedor de datos, la cadena de conexión entre otros.
    - ReportParameters. Aquí se define el nombre del parámetro, el tipo y el texto que mostrará al usuario.
    - PageHeader. Esta es la cabecera del informe.

- Body. Este es el cuerpo del Informe, aquí es donde van los elementos del informe, como las especificaciones de diseño y los ReportItems que contiene las tablas, textbox y demás ítems del informe.
- PageFooter. Esta es el pie del informe.
- Propiedades de Página. Ancho, alto, márgenes, idioma y todo lo relacionado con el Report en general.
- DataSets. Contiene el/los DataSet/s utilizados para el informe. En estos se pueden encontrar: los campos y sus tipos y la consulta empleada entre otros.

3.6 Interfaz de usuario

Se utilizó HTML (Hyper Text Markup Lenguage***) (***«lenguaje de marcas de hipertexto»), para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, entre otros.

Para hacer el sistema más agradable a la vista, se utilizó CSS (Cascade Style Sheet), es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML, en la cual podemos definir aspectos visuales de nuestra página web, como el tamaño y tipo de letra, posicionamiento de componentes de la página web y animaciones.

Razor es un motor de vistas, implementado desde MVC 3 con el que podemos generar vistas, más claras, más sencillas e intuitivas incluyendo en su estructura código de programación de alto nivel. Dicho de otra forma, mediante el uso de ***Razor*** pudimos insertar código en nuestras vistas.

Minimiza el número de caracteres presentes en un código fuente, permitiendo un flujo de trabajo rápido y elegante. Por otro lado, no es necesario interrumpir su codificación de la vista para indicar explícitamente los bloques de servidor dentro de su HTML, es decir, se puede mezclar, código servidor y código cliente, y siempre quedarán identificadas ambas partes ya que el analizador es capaz de deducir esto. Esto permite una sintaxis muy compacta y expresiva, además de páginas menos pesadas, lo que influye en el tiempo de carga de la aplicación.

CAPÍTULO 3
MÓDULO DE REPORTES

A continuación se muestra un caso de uso del módulo de reportes.

Descripción

El usuario podrá generar distintos reportes relacionados con la función de los clientes dentro del Sistema.

Flujo Básico

No.	Flujo Principal	Flujo Alterno
1	El usuario ingresa a SINCO	
2	El usuario selecciona la opción de reportes	
3	El sistema presenta el menú de reportes disponibles para DGO	
4	El usuario selecciona una opción de reportes	
5	El sistema despliega la pantalla con los filtros disponibles para el reporte seleccionado	
6	El usuario ingresa los datos para los filtros del reporte	
7	El usuario selecciona generar el reporte	El usuario selecciona nuevo reporte con lo que se limpian todos los datos de los filtros
8	El sistema genera el reporte	
9	El sistema despliega la información y opciones de exportación válidas	
10	El usuario selecciona cerrar	
11	El sistema cierra la pantalla de reporte	

Precondiciones

Se tiene una cuenta para ingreso al sitio de DGO, con permiso para generar los reportes

Postcondiciones

Se obtienen los reportes con la información solicitada

El módulo de reportes está conformado por dos importantes secciones, la aplicación web y Reporting Services, las cuales se comunican entre sí para lograr el funcionamiento correcto del módulo. En la figura 8 se muestra como la aplicación interactúa con los módulos, en la figura 9 se muestra el menú del módulo de reportes.

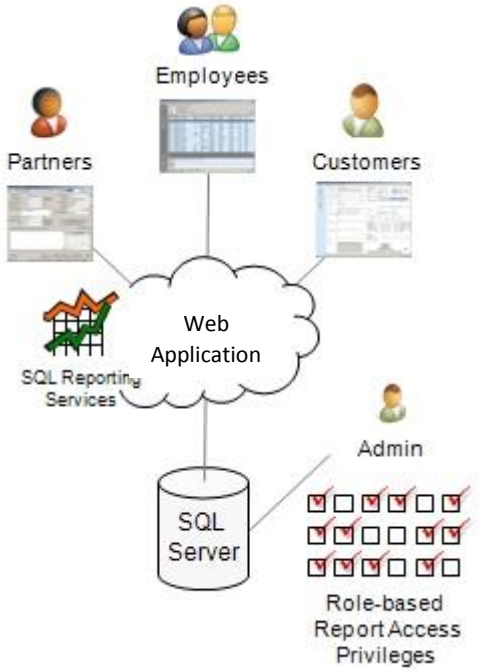


Figura 8. Diagrama de interacción en la aplicación web

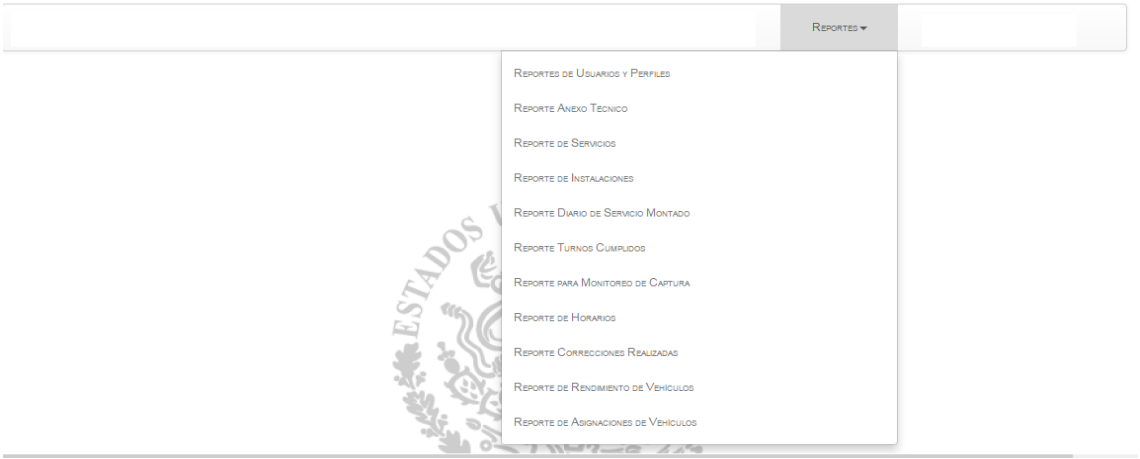


Figura 9. Menú de reportes en la aplicación

3.1 Aplicación web (Parte I)

Como se había comentado anteriormente, el proyecto utiliza una arquitectura MVC, la cual se logra a través del Framework de Microsoft MVC .NET. El módulo de reportes en su parte "Aplicación Web" contiene:

- Vistas (Una para cada reporte)

- Modelos (Uno para cada reporte)
- Controlador
- RestfulModel
- Repositorio de Reportes (Realiza la comunicación con la base de datos)
- Reporte Genérico

El objetivo de la aplicación web es mostrar al usuario una serie de parámetros mediante los cuales puede filtrar el reporte, estos parámetros muestran en forma de DropDownList y se generan de manera dinámica de acuerdo a la base de datos. Posteriormente recolecta los parámetros y los envía a Reporting Services.

3.1.1 Modelos

Lo primero que se hizo para implementar MVC en el modulo de reportes, fue generar los modelos de los mismos, aquí se definieron todas aquellas variables que serían necesarias para el reporte.

Aun cuando cada reporte necesita de parámetros diferentes, los tipos son prácticamente los mismos.

```
[DisplayName("Zona:")]
public int IdZona { get; set; }
public SelectList ListaZonas { get; set; }
```

Figura 10. Dropdownlist utilizando C#

En la Figua 10 podemos observar, que se declaró un objeto de tipo SelectList y un int, que será el identificador de nuestra Lista Desplegable. Tenemos múltiples Listas que permitirán al usuario filtrar el reporte, de acuerdo a sus necesidades

```
public class Customer
{
        [Required(ErrorMessage="Customer code is required")]
        public string CustomerCode
        {
            set;
            get;
        }
}
```

Figura 11. Ejemplo de atributo Required en C#

En la Figura 11 se observa que también se marcaron como obligatorios los campos que el cliente requirió y además se estableció el rango de valores que las listas desplegables aceptan.

Otra cuestión importante que el cliente requirió fue la validación de fechas en ciertos reportes. La fecha inicial no podía ser mayor a la actual ni menor a 1/1/1901, además la fecha final no podía ser menor a la fecha inicial.

```
[DisplayName("Vigencia Fecha Inicio:")]
[DisplayFormat(DataFormatString = "{0:dd/mm/yyyy}", ApplyFormatInEditMode = true)]
[ValidarFechaInicioAttribute(ErrorMessage = "ERROR: La fecha inicial no puede ser mayor a la actual ni menor a 1/1/1901.")]
public DateTime? FechaInicio { get; set; }

[DisplayName("Fecha Fin:")]
[DisplayFormat(DataFormatString = "{0:dd/mm/yyyy}", ApplyFormatInEditMode = true)]
[ValidarFechaFinAttribute(ErrorMessage = "ERROR: La fecha final no puede ser menor a la fecha inicial.")]
public DateTime? FechaFin { get; set; }
```

Figura 12. Declaración de fechas en

Como podemos observar en la Figura 12 eso se logró a través de ValidarFechaInicioAttribute y ValidarFechaFinAttribute, además mostrará un mensaje de error al usuario en caso de que la validación no sea exitosa.

```
public class ValidarFechaInicioAttribute : ValidationAttribute, IClientValidatable
{
    protected override ValidationResult IsValid(Object value, ValidationContext validationContext)
    {
        if (value == null)
        {
            return ValidationResult.Success;
        }

        try
        {
            DateTime fechaSeleccionada = (DateTime)value;
            if (new DateTime(1901, 1, 1) < fechaSeleccionada && fechaSeleccionada <= DateTime.Now)
            {
                return ValidationResult.Success;
            }
            else
            {
                return new ValidationResult(this.FormatErrorMessage(validationContext.DisplayName));
            }
        }
        catch (Exception)
        {
            return new ValidationResult(this.FormatErrorMessage(validationContext.DisplayName));
        }
    }

    public IEnumerable<ModelClientValidationRule> GetClientValidationRules(ModelMetadata metadata, ControllerContext context)
    {
        var rule = new ModelClientValidationRule();
        rule.ErrorMessage = FormatErrorMessage(metadata.GetDisplayName());
        rule.ValidationType = "exclude";
        yield return rule;
    }
}
```

Figura 13. Validaciones de fecha en C#

La clase que se hizo hereda de ValidationAttribute, la cual es una clase abstracta, parte del API de .NET, nos permite crear atributos personalizados los cuales podremos implementar y validar en cualquier variable de nuestro modelo, en la figura 13 observamos la implementación de esta clase.

En este caso se creó el atributo FechaInicio y FechaFin, al ser una clase abstracta tuve que sobrecargar el método IsValid, en el cual se realizan las condiciones necesarias, para que nuestra clase determine si el atributo personalizado se validó correctamente.

En el reporte anterior, el cliente no requirió que las fechas fueran obligatorias, por lo cual, en caso de que el usuario no las seleccione, nuestra clase de validación no mostrara error.

La información es algo vital para nuestro cliente, tener registro de todas las actividades le permite controlar eficazmente lo que se hace y como se hace, por lo se nos pidió además, que se guardará un registro en la base de datos con todas las veces que se generaba un reporte, y los parámetros que eran enviados a éste.

Esto se logró haciendo una sobrecarga del método ToString(), ya que de esa manera se centraliza la recolección de información en un solo método, tal como se observa en la figura 14.

```
public override string ToString()
{
    return base.ToString() + " ,idTipoServicioProteccion=" + this.idTipoServicioProteccion + ",IdContrato=" + this.IdContrato
                           + " ,IdZona=" + this.IdZona + ",IdServicio=" + this.IdServicio + ",IdInstalacion=" + this.IdInstalacion
                           + " ,FechaInicio=" + (this.FechaInicio == null ? new DateTime(1901, 1, 1) : this.FechaInicio)
                           + " ,FechaFin=" + (this.FechaFin == null ? new DateTime(1901, 1, 1) : this.FechaFin)
                           + " ,IdEstatus=" + this.IdEstatus;
}
```

Figura 14. Sobre escritura del método ToString()

Los modelos de todos los reportes harán una sobrecarga del método ToString() dado que cada uno utiliza diferentes parámetros. Pero al final la recolección se hará invocando al mismo método.

El modelo es una parte esencial del MVC, sin embargo, es sólo el esqueleto, por tal razón es necesario darle funcionalidad a la aplicación web.

Controlador.

Aquí se encuentra toda la lógica del Módulo de Reportes.

Analizando un poco los requerimientos del usuario en cuanto al Módulo de Reportes me di cuenta que hay 2 puntos importantes en el ciclo de vida de la aplicación web.

Cuando el usuario:

- Filtra (selección de parámetros)

-Genera el reporte.

En ambos se hacen llamadas al servidor, sin embargo el propósito es diferente, en la primera se trae la información con la que se llenaran los dropdownlist y en la segunda se hace un submit de nuestro form. Por lo cual, es necesario controlar estas dos acciones. Esto se hace implementando dos métodos para cada reporte, dentro del controlador. Un método GET y otro POST.

En ambos métodos se crea una instancia de nuestro modelo y se inicializa, como se muestra en la Figura 15.

```
var ri = new ReporteDiario()
{
    ActionName = "Diario",
    ControllerName = "Reportes",
    ListaZonas = new SelectList(new List<String>() { }),
    ListaEstaciones = new SelectList(new List<String>() { }),
    ListaServicios = new SelectList(new List<String>() { }),
    ListaConceptos = new SelectList(new List<String>() { }),
    ListaContratos = new SelectList(new List<String>() { }),
    ListaInstalaciones = new SelectList(new List<String>() { }),

    Fecha = null,
    ReporteGenerado = 0
};
```

Figura 15. Instancia del modelo de un reporte

Una variable importante es la de ReporteGenerado la cual es una bandera para la Vista, que se activa al presionar el botón de “Generar”, de esta manera la Vista sabe cuándo renderizar el reporte.

Posteriormente, se realiza el llenado de todos los SelectList del modelo mediante el Restfulmodel, el cual contiene todos los métodos de los reportes, que serán usados por la aplicación, tal como se muestra en la Figura 16.

```
ZonaPermisos zone = new ZonaPermisos() { idZona = 0, zonDescripcion = "SELECCIONAR" };
List<ZonaPermisos> lz = new RestfulModel().GetZonas(User.GetSPFUser().NombreDeUsuario, User.GetSPFUser().Contraseña);
lz.Insert(0, zone);
ri.ListaZonas = new SelectList(lz, "idZona", "zonDescripcion", ri.IdZona);
```

Figura 16. Llenado de los Dropdownlist con información de la BD

El RestfulModel posteriormente se comunica con el repositorio y ejecuta el StoredProcedure necesario para recabar información de la base de datos.

Es importante mencionar que todos los SelectList tienen un elemento “Seleccionar”, el cual será un valor por default en todos. Es por eso que en el código referido en la Figura 16 se agregó.

La única diferencia entre el método GET y POST es que el segundo necesita mantener en pantalla y seleccionados los parámetros que el usuario escogió en la primera fase, después de realizar el submit. El método GET, muestra los filtros sin selección.

Vista

Dado que los requerimientos del usuario en cuanto a la simplicidad y fluidez en la interfaz gráfica eran muchos, se utilizó Razor, el cual al ser nativo del framework .NET la conversión a HTML es prácticamente inmediata.

Con Razor, se pueden crear interfaces de manera rápida y eficiente pero sobre todo ordenadas, crear vistas con HTML plano puede resultar en un caos si la complejidad de estas es grande.

Lo primero que se hizó fue ligar el modelo con la vista, esto se realizó con el identificador "@model" seguido de la ruta del modelo, como se muestra en la Figura 17.

```
@model SPFDGO.Models.Reportes.ReporteDiario
@{
    ViewBag.Title = "Reporte Diario de Servicio Montado";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Figura 17. Uso del identificador @model

Posteriormente se declaró el formulario, como se observa en la Figura 18.

```
<div class="panel-body">
@using (Html.BeginForm(
Model.ActionName,
Model.ControllerName,
FormMethod.Post,
new { @class = "form-horizontal", role = "form" }))
{

    <div style="visibility:hidden">
    @Html.RadioButtonFor(m => m.ReporteGenerado, 0, new { @id = "Nuevo" })
    @Html.RadioButtonFor(m => m.ReporteGenerado, 1, new { @id = "Generado" })
 </div>
```

Figura 18. Declaración del formulario web con Razor

La notación pareciera compleja sin embargo es sumamente simple, el HTML.BeginForm es un helper de Razor, los HTMLHelpers permiten crear estructuras simples o complejas de HTML, en unas pocas líneas de código.

@Html.RadioButtonFor es equivalente a un input de tipo radio en HTML.

Una cosa más que pueden notar en el código es que se utilizan referencias al modelo de dos formas, mediante la palabra Model y con notación lambda, siendo esta última necesaria para hacer referencia a partes del modelo dentro de nuestro HTMLHelper.

La vista sabe perfectamente que la palabra reservada model estará ligada a un Modelo en particular, que indicamos al inicio del archivo.

Posteriormente se declaran los dropdownlists, mediante notación lambda, indicando con que variable de nuestro modelo se llenaran. Todos los DropDownLists se mapearon con variables de tipo SelectList. Las cuales fueron llenadas con información de la base de datos, mediante el controlador. En la Figura 19 se muestra un ejemplo del llenado de un Dropdownlist.


```
<body>
    <div>
        @using (Html.BeginForm())
        {
            <div style="margin:30px;">
                Color :  @Html.DropDownListFor(m => m.FavouriteColor)
                <br />
                Name : @Html.EditorFor(x => x.Name)
                <br />
                <input type="submit" value="Submit" />
            </div>
        }
    </div>
</body>
```

Figura 19. Dropdownlist en Razor con HTML

Se declararon todos los DropDownList que requería el reporte, tal como se muestra en la Figura 20.

Figura 20. Dropdownlist visualizados desde la aplicación web

Así como en el método ToString(), en las vistas se hace una recolección de parámetros, pero el propósito es distinto.

```
@if (Model.ReporteGenerado == 1)
{
     <div class="row" id="#contenedorReporte">
      <div class="panel panel-default" id="ReporteD">
          <div class="panel-heading">
              <h3 class="panel-title"><strong>Reporte Diario de servicio montado</strong></h3>
          </div>
          <div class="panel-body">
              <br />
              <div class="row">
                  <div class="col-xl-12">
                         <iframe id="MyReport" src="../../Reports/ReporteGenerico.aspx?idReporte=5
                         &IdZona=@Model.IdZona
                         &IdServicio=@Model.IdServicio
                         &IdInstalacion=@Model.IdInstalacion
                         &IdEstacion=@Model.IdEstacion
                         &IdTipoServicio=@Model.IdConcepto
                         &IdContratoConvenio=@Model.IdContrato
                         &FechaFin=@Model.Fecha" width="100%" height="430" ></iframe>
                           <div> </div>
                           <div class="banner" id="barra" style="text-align:center; font-weight:bold">SERVICIO DE PROTECCIÓN FEDERAL</div>
                           <div> </div>
                  </div>
              </div>
          </div>
      </div>
  </div>
 }
</div>
<script src="../../Scripts/Reportes/RepDiarioValid.js" type="text/javascript"></script>
```

Figura 21. Envío de parámetros recolectados al reporte genérico

Como se puede observar en la Figura 21, primero se verifica el estado de la bandera "ReporteGenerado", después se hace un frame que contendrá el Reporte Genérico, al cual se le mandan los parámetros que seleccionó el usuario.

Para centralizar y estandarizar la comunicación con ReportingServices, se realizó un Reporte Genérico el cual renderizará los reportes en función del idReporte que se envía por URL, tal y como pueden apreciar en la imagen anterior.

Por último se referencia al script de JS que permitirá controlar aún más la aplicación, a través de Listeners de eventos, ya que hay reportes donde la selección de un DropDownList habilita otro.

Reporte Genérico

El reporte genérico, es el encargado de renderizar todos los reportes, su función es muy importante ya que actúa como intermediario entre la aplicación y ReportingServices desde la configuración hasta el envió de parámetros.

Está definido en una página aspx la que contiene un ReportViewer, la cual es un control predefinido de la librería Microsoft.Reporting.WebForms, que nos permite renderizar el Reporte.

El código en C# del archivo aspx, nos permite configurar la conexión, no mostraré imágenes sobre la configuración del cliente, debido a cuestiones de privacidad, sin embargo plantearé una idea general del proceso.

1. Declarar una variable que contendría la ruta donde se guardan los reportes.
2. Después establecer el procesamiento remoto, ya que los reportes se guardan en un servidor independiente.
3. Después crear un switch del idReporte, para saber que reporte se tiene que renderizar.
4. En cada escenario del switch, los cuales representan un reporte en particular, crear un array con los parámetros necesarios.
5. Hacer la llamada a Reporting services y enviar los parámetros.
6. Meter todo en un bloque try catch, por si algo en la comunicación falla.

Lo anterior se muestra en la Figura 22.

```
private void ShowReport()
{
    try
    {
        string urlReportServer = "http://sqlDBServer//Reportserver";
        rptViewer.ProcessingMode = ProcessingMode.Remote; // ProcessingMode will be Either Remote or
Local
        rptViewer.ServerReport.ReportServerUrl = new Uri(urlReportServer); //Set the ReportServer Url
        rptViewer.ServerReport.ReportPath = "/ReportName"; //Passing the Report Path

        //Creating an ArrayList for combine the Parameters which will be passed into SSRS Report
        ArrayList reportParam = new ArrayList();
        reportParam = ReportDefaultPatam();

        ReportParameter[] param = new ReportParameter[reportParam.Count];
        for (int k = 0; k < reportParam.Count; k++)
        {
            param[k] = (ReportParameter)reportParam[k];
        }
        // pass crendentitilas
        //rptViewer.ServerReport.ReportServerCredentials =
        //  new ReportServerCredentials("uName", "PassWORD", "doMain");

        //pass parmeters to report
        rptViewer.ServerReport.SetParameters(param); //Set Report Parameters
        rptViewer.ServerReport.Refresh();
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Figura 22. Archivo de configuración que comunica .NET con Reporting Services

BUSINESS INTELLIGENCE

Es una estrategia para la empresa que persigue incrementar el rendimiento de la misma o la competitividad del negocio, a través de la organización inteligente de sus datos históricos (transacciones u operaciones diarias), usualmente residiendo en Datawarehouse corporativos o Data Marts departamentales.

3.2 REPORTING SERVICES (PARTE II)

Todos los reportes fueron creados con Visual Studio 2008 – SQL Server Bussines Intelligence Development Studio, ya que esta herramienta nos permite hacer uso del framework Reporting Services.

Dado que esta herramienta interactúa directamente con la base de datos, no se mostrará información importante, ni datos específicos, por cuestiones de privacidad del cliente. Primero se abrió el IDE tal como se observa en la figura 23.

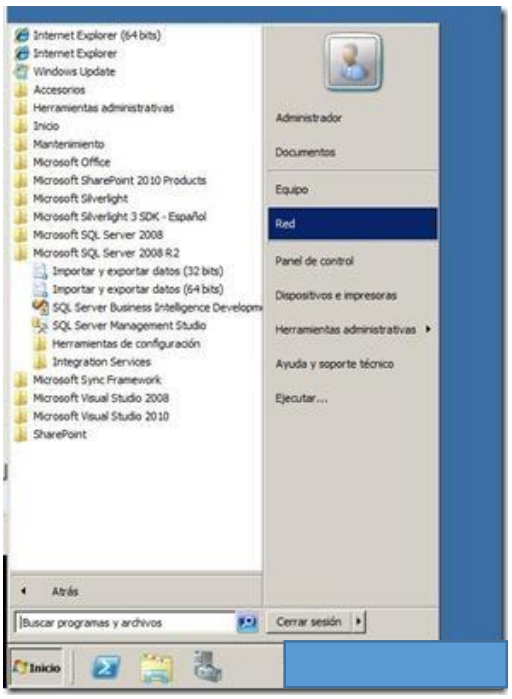


Figura 23. Abriendo SQL Server - Bussines Intelligence Development Studio

Lo primero que se hizo fue crear un nuevo proyecto, como se observa en la Figura 24, la interfaz de esta herramienta es muy similar al IDE de programación Visual Studio 2010, por lo cual familiarizarse con la interfaz es una tarea sencilla.

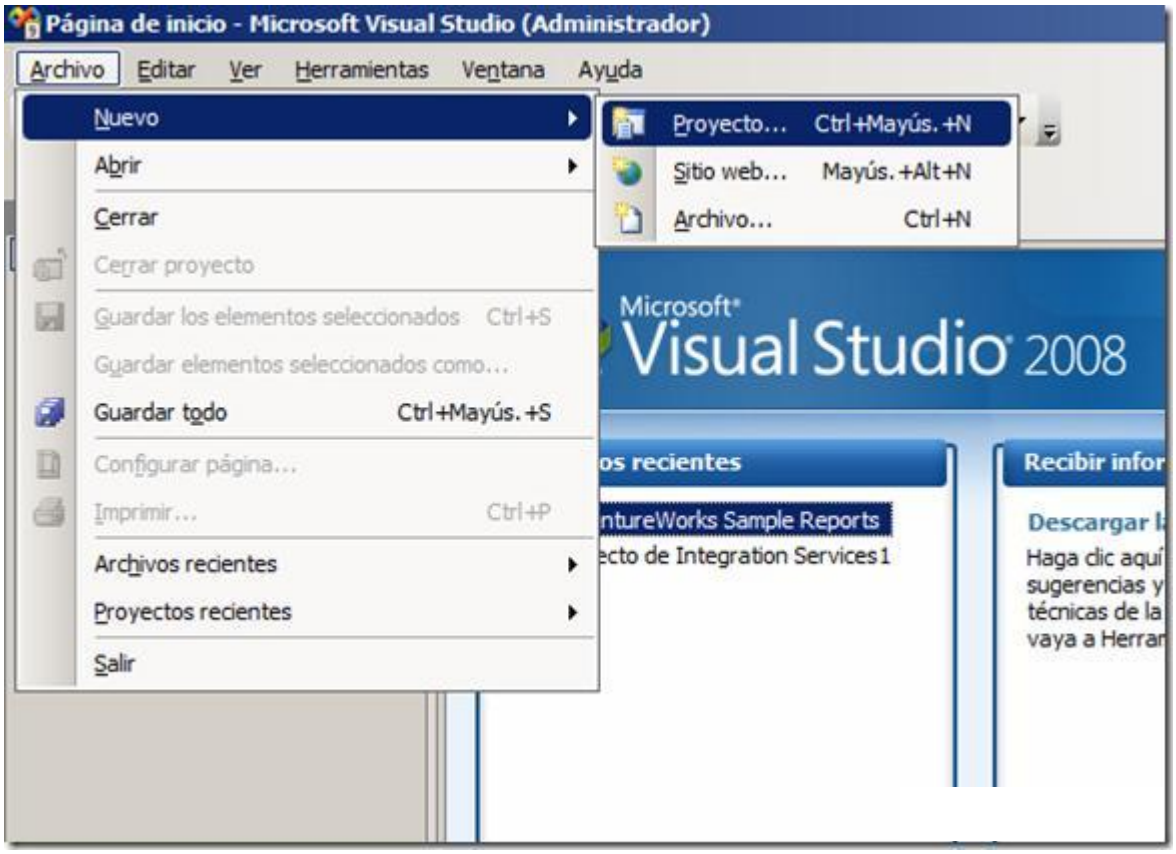


Figura 24. Creando un nuevo proyecto de Reporting Services

Luego elegir la plantilla (template) Proyecto de servidor de informes (Report Sever Project) y escribir el nombre del reporte así como la ruta en la que se guardará, tal como se observa en la Figura 25.

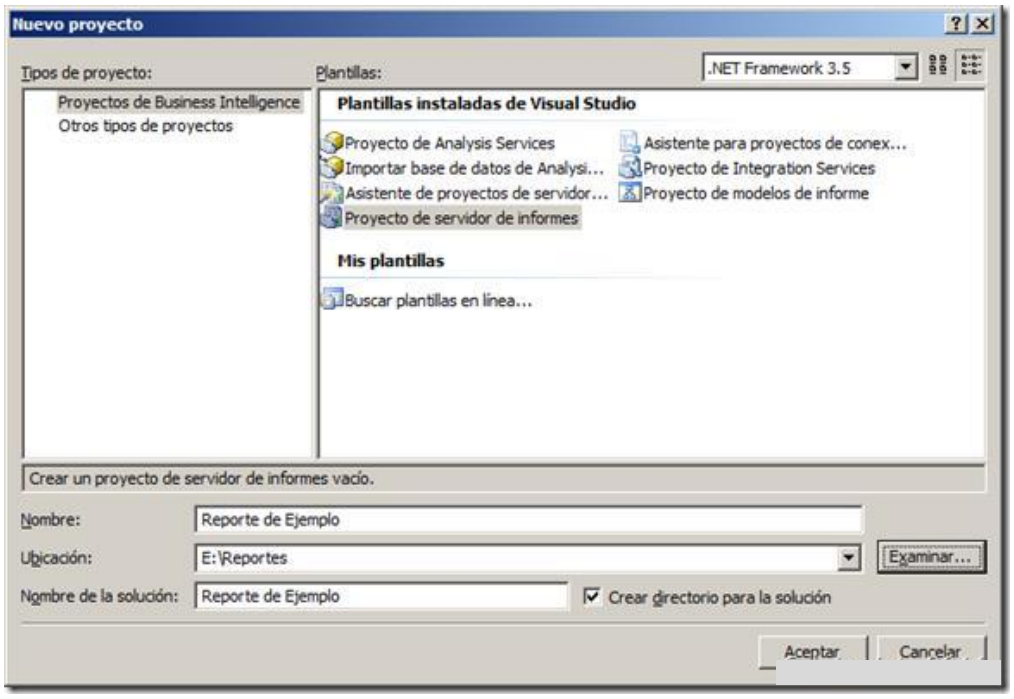


Figura 25. Seleccionando una plantilla

Cabe mencionar que se creó una solución para cada sitio web (cuatro en total) dado de que estos reportes se alojan en carpetas del servidor del cliente, es importante un orden. En la figura 26 se muestra una solución creada.

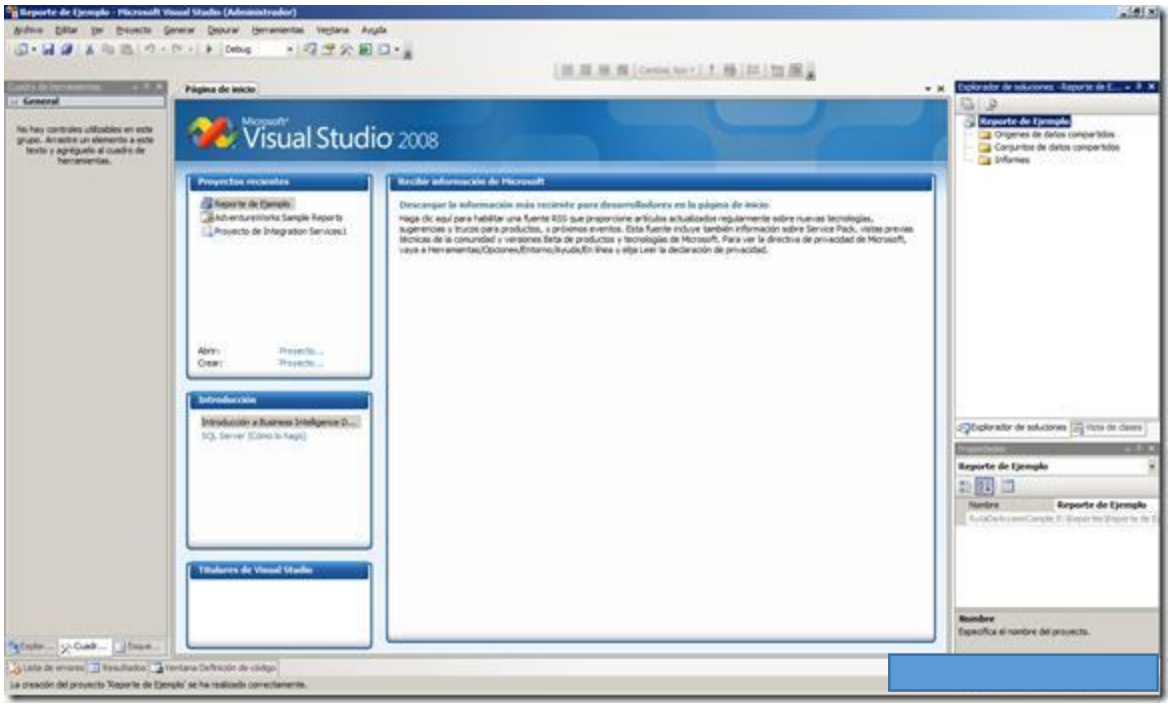


Figura 26. Estructura de un proyecto de Reporting

En la Figura 27, se observa el Explorador de Soluciones (Solution Explorer) y podemos ver la creación de tres carpetas:

1. Orígenes de datos
2. Conjunto de datos compartidos
3. Informes

Figura 27. Explorador de soluciones de Reporting Services

Para crear una conexión a la base de datos y posteriormente a un procedimiento almacenado hay que crear un Origen de datos compartidos, tal como se observa en Figura 28. Las conexiones pueden hacerse en cada reporte por separado, pero es preferible tener una para todos.

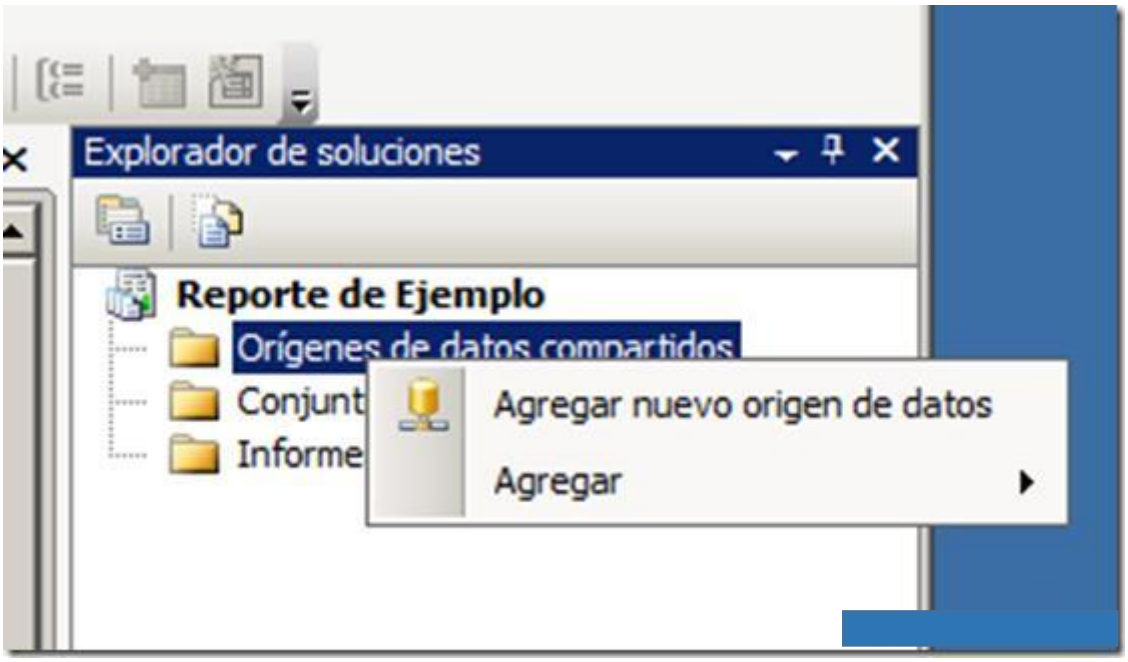


Figura 28. Agregando un nuevo DataSource

Una vez seleccionada la opción "Agregar nuevo origen de datos" tendremos una caja de diálogo como la de la Figura 29:

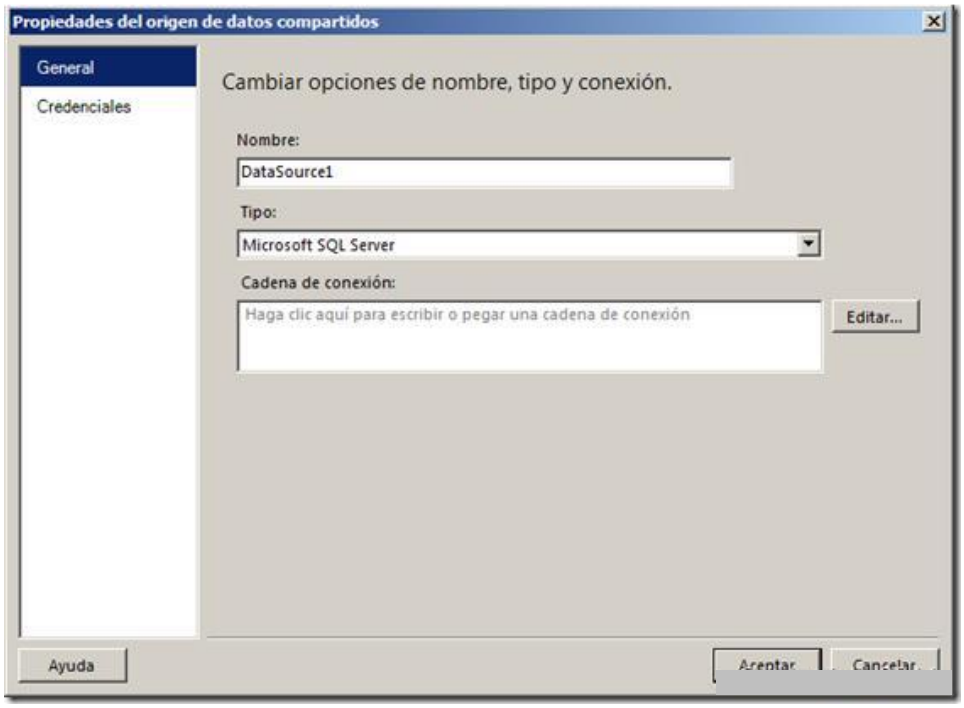


Figura 29. Seleccionando el nombre del DataSource

En la ventana anterior se pone la cadena de conexión, que contiene la IP del servidor donde se aloja la base de datos.

Para describir el proceso, se hará uso de un ejemplo.

En la Figura 30 se crea la conexión AdventureWorks2008

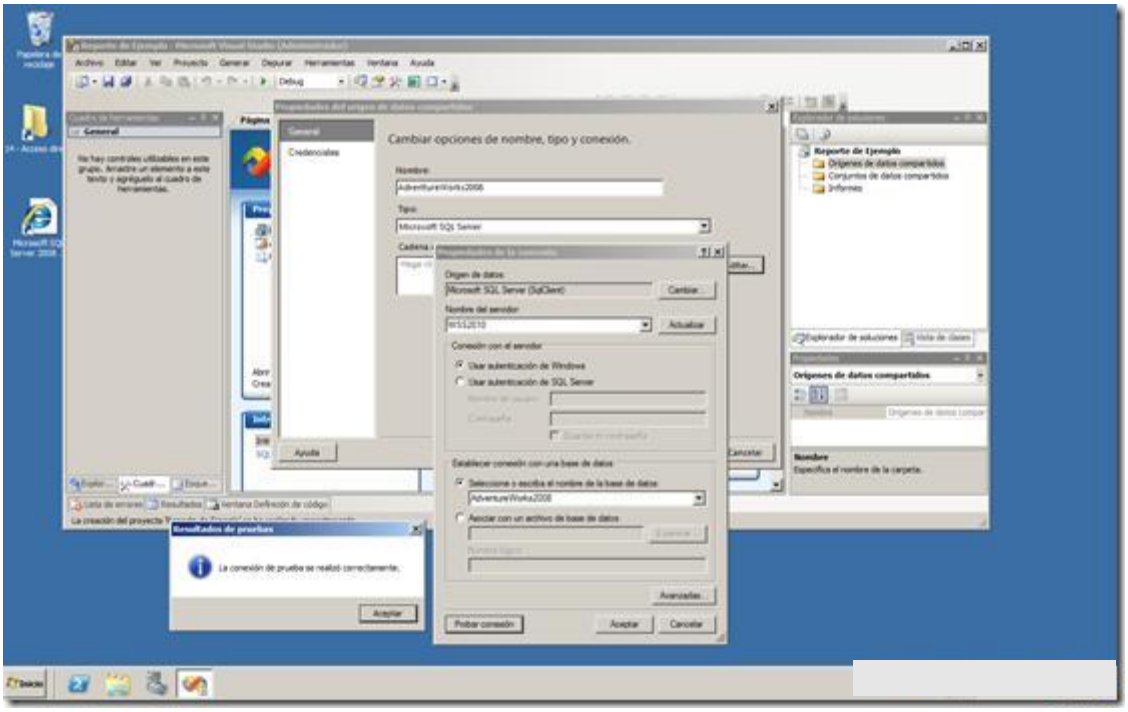


Figura 30. Eligiendo la base de datos y el tipo de autenticación

Seleccionamos o escribimos el nombre del servidor, luego seleccionamos la base de datos y por último probamos conexión y aceptamos para volver a la caja de diálogo anterior, tal como observamos en la Figura 31:

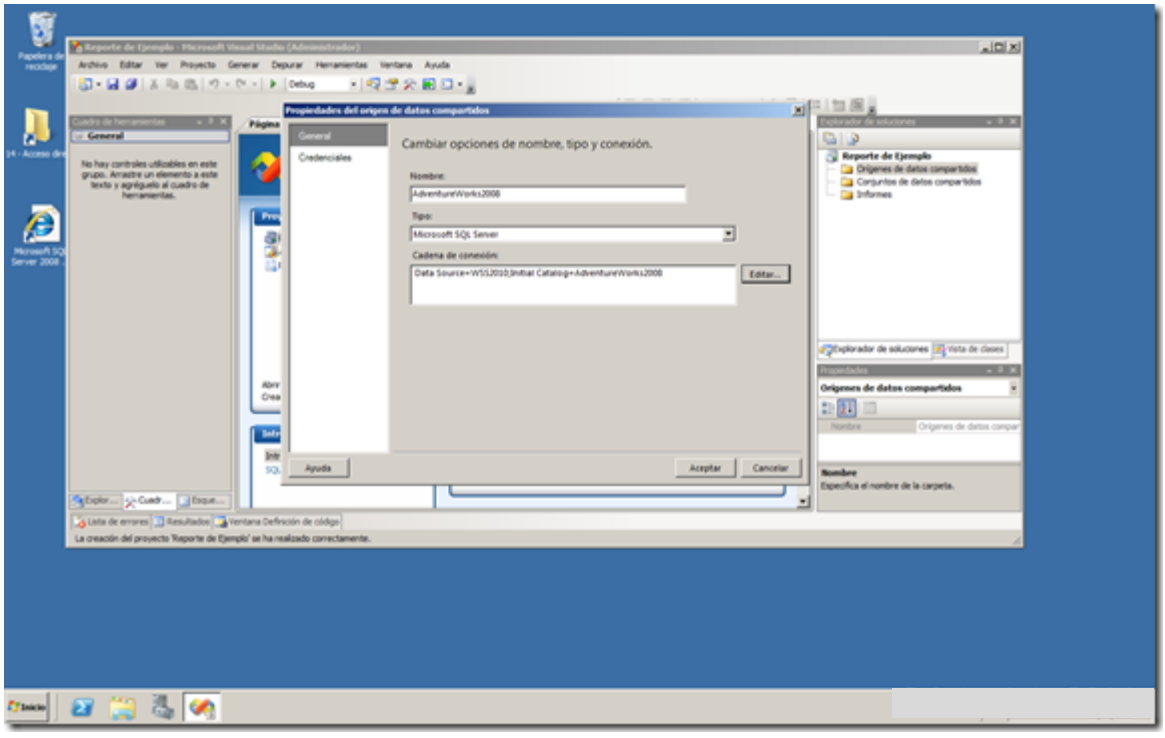


Figura 31. Observamos que la cadena de conexión se crea automáticamente

Lo siguiente es crear un reporte, tal como en la Figura 32.

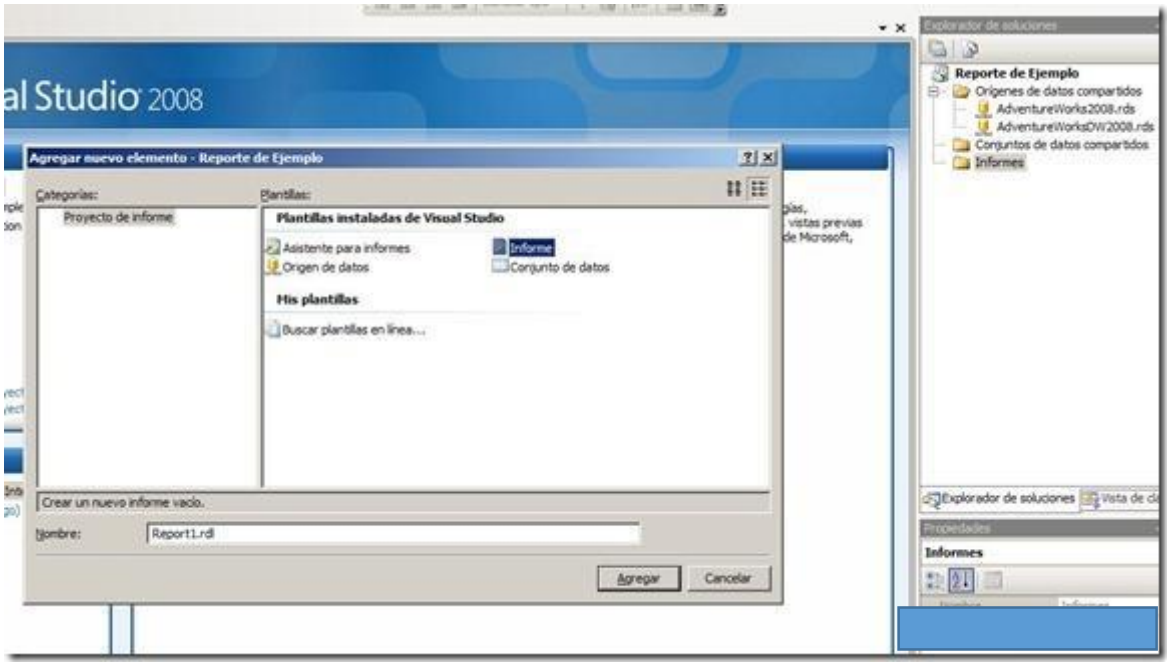


Figura 32. Creando un nuevo reporte

Luego de poner un nombre a nuestro reporte aparece el Diseñador de Reportes (Report Designer) con 2 pestañas: Diseño (Design) y Vista Previa (Preview). En las versiones anteriores aparecía una pestaña adicional con los datos (Data), ahora esta pestaña aparece como "Datos del Informe".

Lo siguiente fue especificar que elemento consumiría el reporte, un reporte puede consumir múltiples elementos de la base de datos, para ello se creó un DataSet, tal como se observa en la Figura 33.

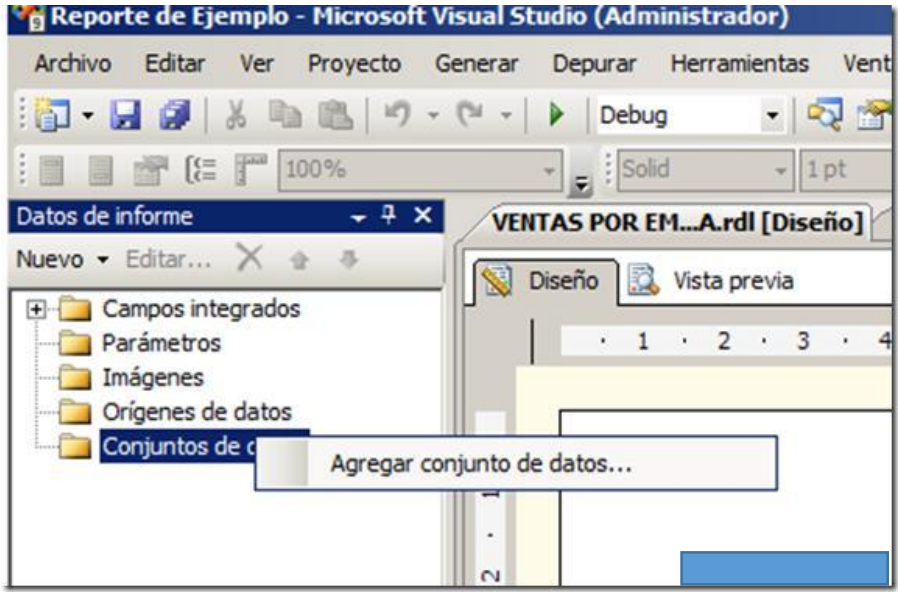


Figura 33. Agregando un DataSet

Es importante especificar un nombre que identifique correctamente la información que está consumiendo. Se puede embeber un DataSet al reporte o crearlo por separado y hacer referencia a él

desde el reporte. En el proyecto se optó por la primera opción ya que resulta más fácil administrar, tal como se observa en la Figura 34.

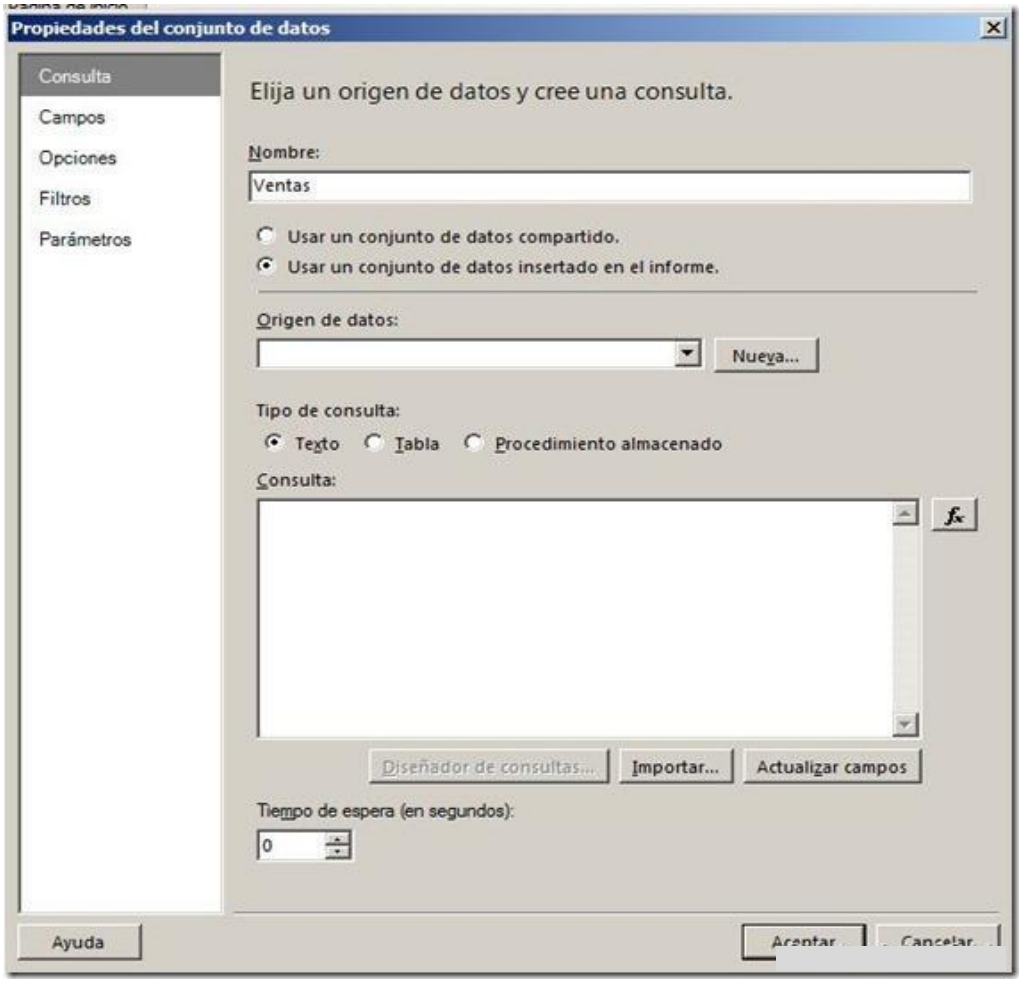


Figura 34. Seleccionando el DataSet que se usara

Luego seleccionar el origen de datos compartido, el cual establecerá que cadena de conexión a la base de datos usaremos, tal como se observa en la Figura 35.

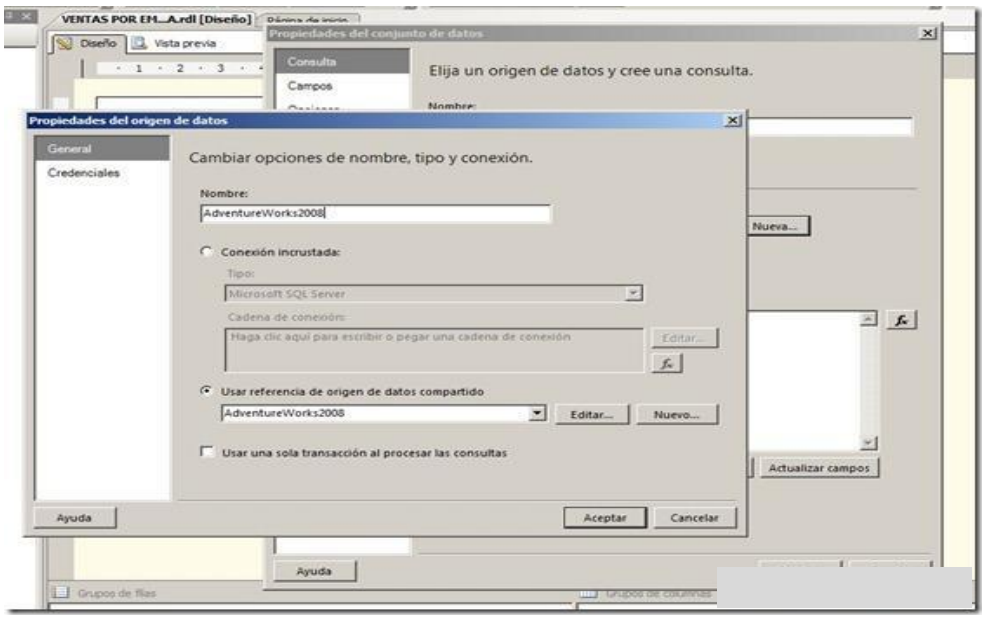


Figura 35. Creando una referencia al origen de datos previamente creado

Una vez hecho esto se selecciona el tipo de consulta, puede ser que consumamos una Vista o un Procedimiento almacenado, en el caso de la vista, se hace a través de una consulta a la base de datos tal como se observa en la Figura 36, con el SP es más fácil, ya que sólo hay que elegir el nombre.

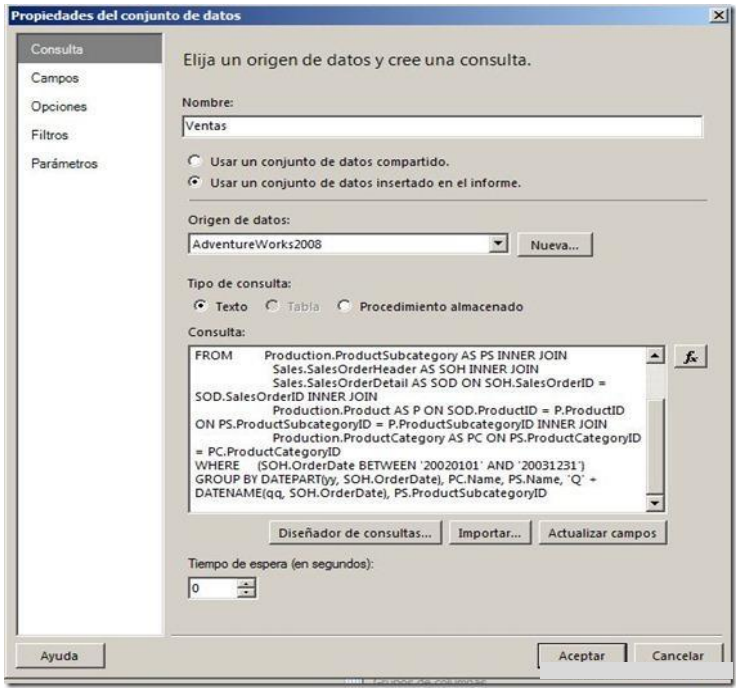


Figura 36. Seleccionando una consulta de tipo texto

En el diseñador de consultas se puede testear una consulta, o incluso ejecutar un procedimiento almacenado directamente con los parámetros deseados.

Lo siguiente fué crear el reporte, para ello Reporting Services nos brinda una Caja de Herramientas (Toolbox) donde aparecen los componentes gráficos que podemos utilizar en el informe y en la pestaña Datos del Informe (Data) estarán a su vez los Conjuntos de datos (DataSet) que podemos utilizar en ellos. Para este ejemplo, primero vamos a la Caja de Herramientas (Toolbox) y elegimos el elemento Matriz y lo arrastramos hacia el Diseño, tal como se observa en la Figura 37.

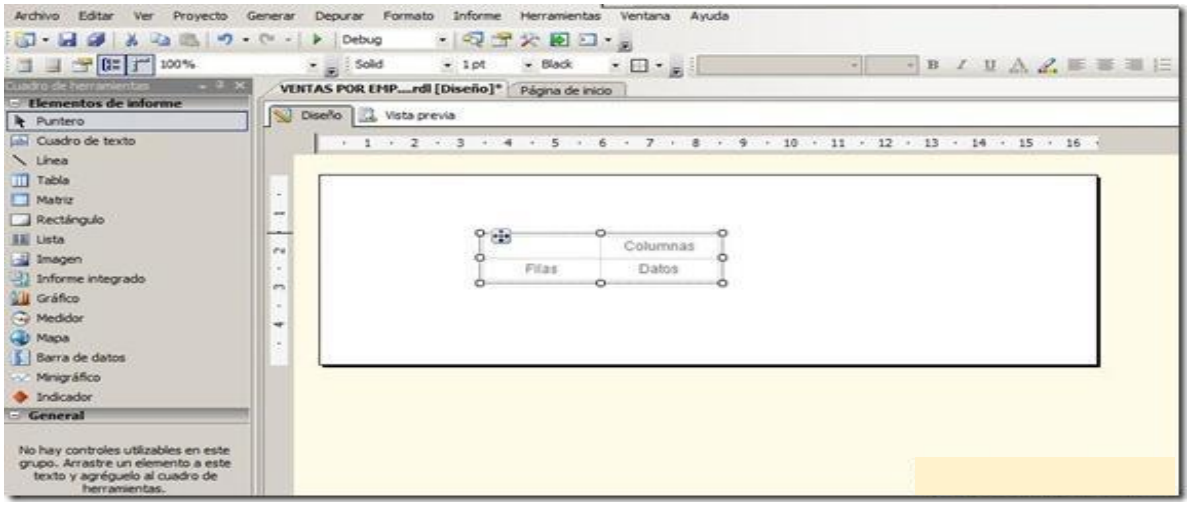


Figura 37. Creando una matriz a través del toolbox

A través del submenú de la izquierda podemos elegir cualquier dato que nuestra consulta o SP haya regresado y mapearlo a un control de la caja de herramientas. La herramienta más potente es la Matriz ya que genera las columnas y los renglones de manera dinámica de acuerdo a la información de la base de datos. En la Figura 38 se observa el mapeo de los datos.

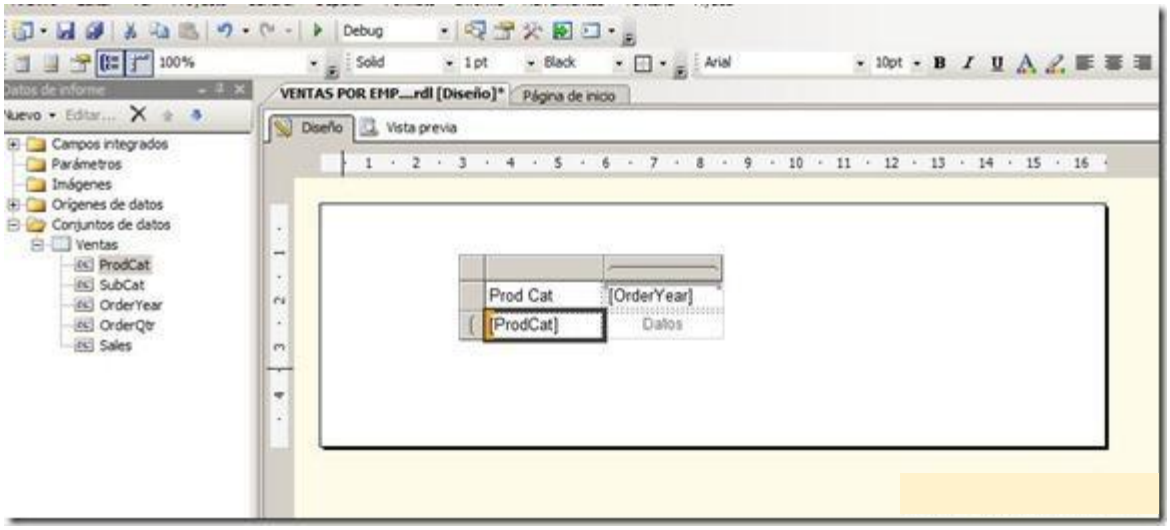


Figura 38. Mapeando datos a la matriz

Mucho de los reportes tenían que estar agrupados de acuerdo a la zona e instalación de la dependencia de gobierno o mostrar información alfabéticamente ordenada, esto sería una tarea sencilla si fueran datos estáticos. En este caso, se hace uso de expresiones, con las cuales se puede manipular la información en tiempo de ejecución. Más adelante se verán algunos ejemplos de expresiones que fueron usadas en el módulo de reportes.

Para agregar un grupo primero nos ubicamos en la matriz, en la fila ProdCat y ahí hacemos clic derecho y elegimos Agregar Grupo (Insert Group) y seleccionamos Grupo Secundario, tal como se muestra en la Figura 39.

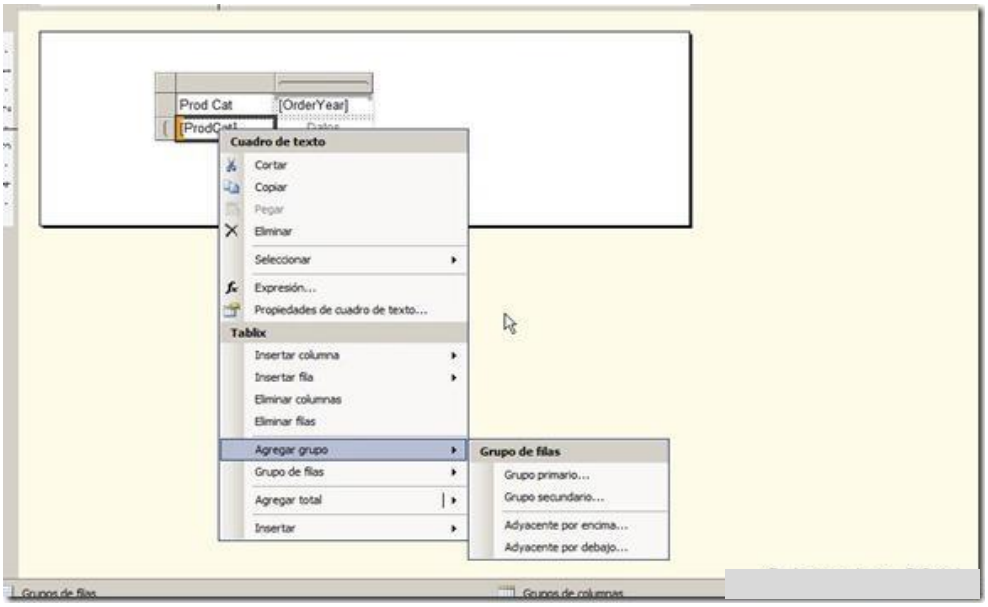


Figura 39. Agregando un grupo de filas a la matriz

Posteriormente se elige el campo por el cual se agrupara la fila, es decir cómo se mostrará la información por bloques, tal como se observa en la Figura 40.

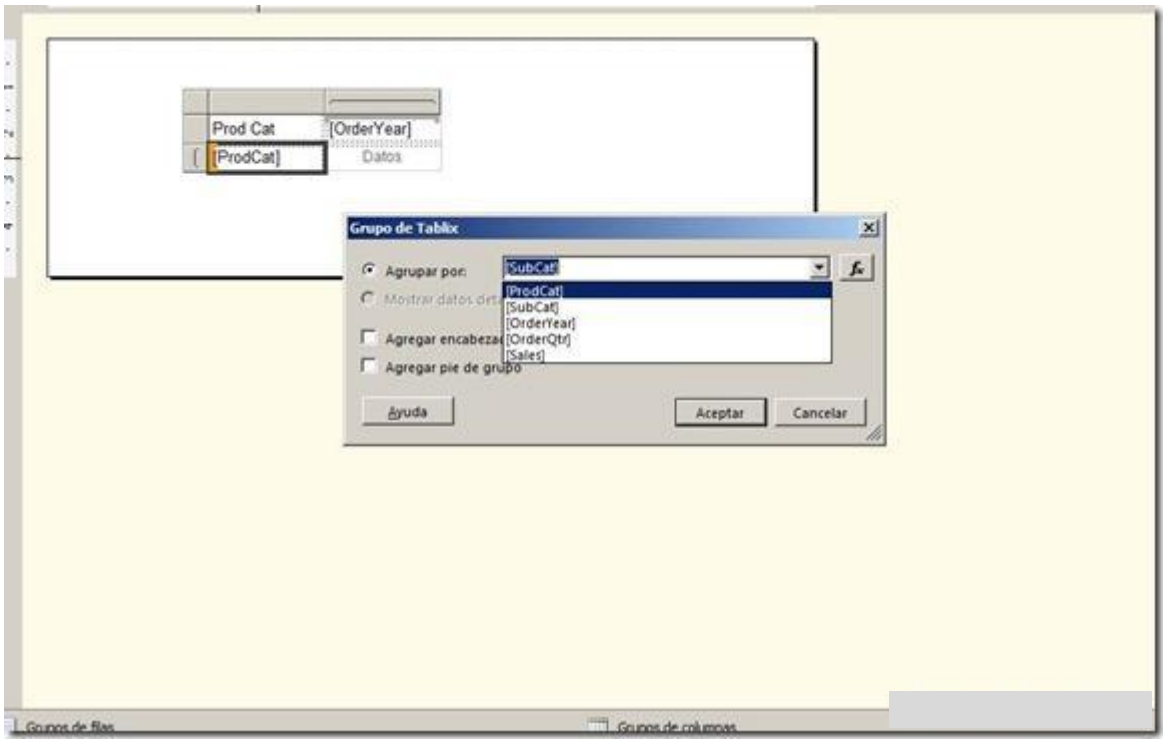


Figura 40. Seleccionando el dato por el cual se agrupara la fila

En la Figura 41, se muestra como quedan los grupos de filas.

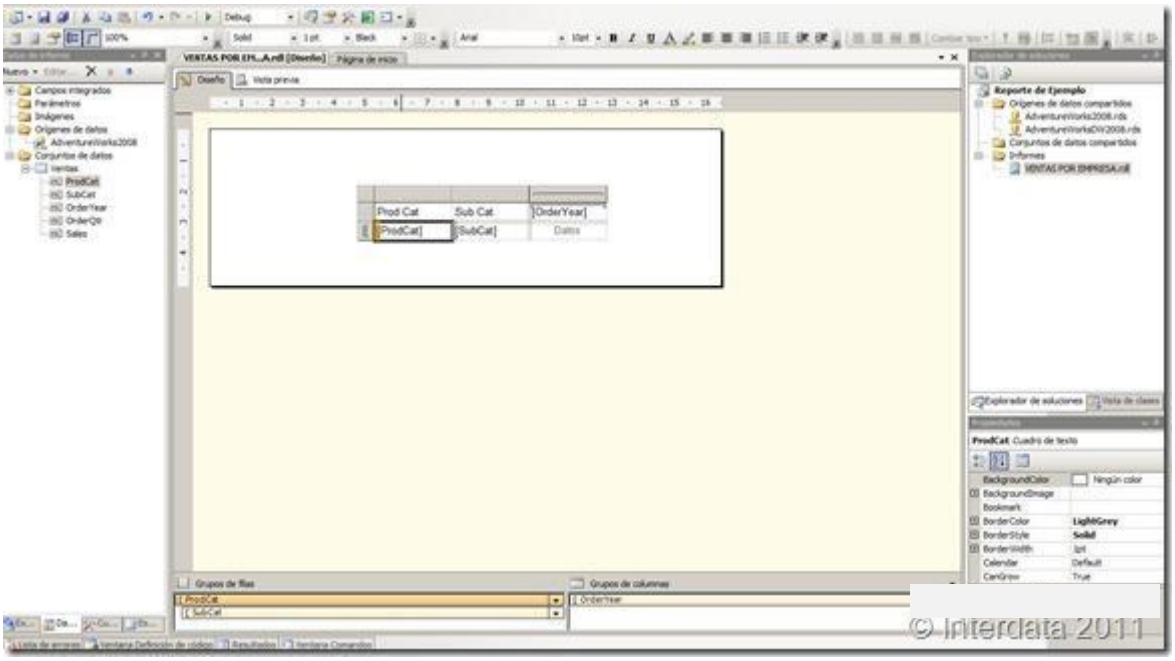


Figura 41. Grupo de fila creado

Al hacer un vista previa, nos damos cuenta como se agrupo por Accesories, Bikes y Clothing. Si en algún momento agregaran más categorías a la base de datos, se generarían automáticamente las agrupaciones correspondientes, tal como se observa en la Figura 42.

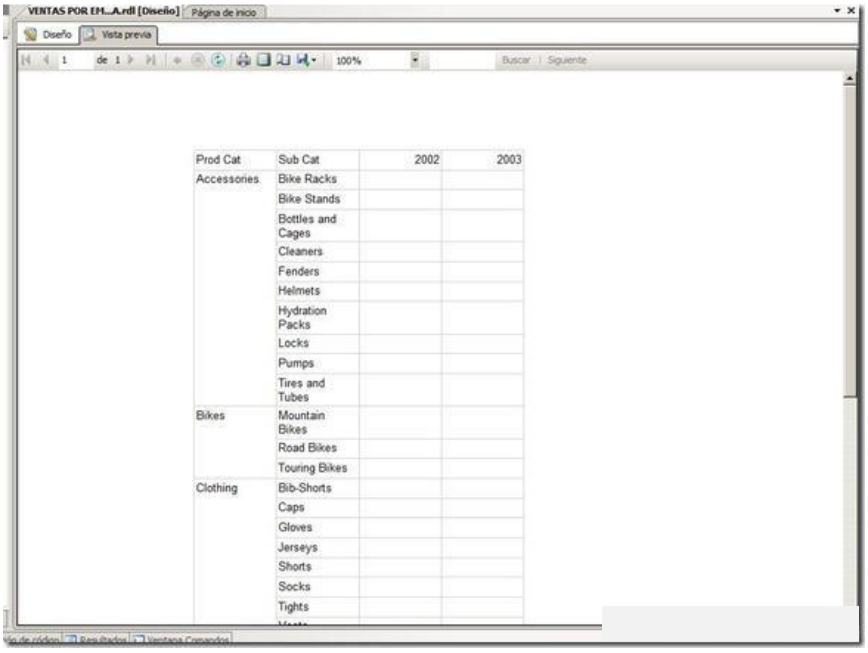


Figura 42. Vista previa de la agrupación creada

En muchos reportes se necesitaban combinar agrupaciones de columnas y renglones para lograr dinamismo en el renderizado de la informacion, poder realizar un Drill Down y Drill Up para estas agrupaciones es muy fácil y útil.

Los pasos que se siguieron fueron: Ocultar la columna sub categoría bajo una condición especial dentro de las propiedades del Grupo. En la figura 43 se observa cómo se accede a estas propiedades.

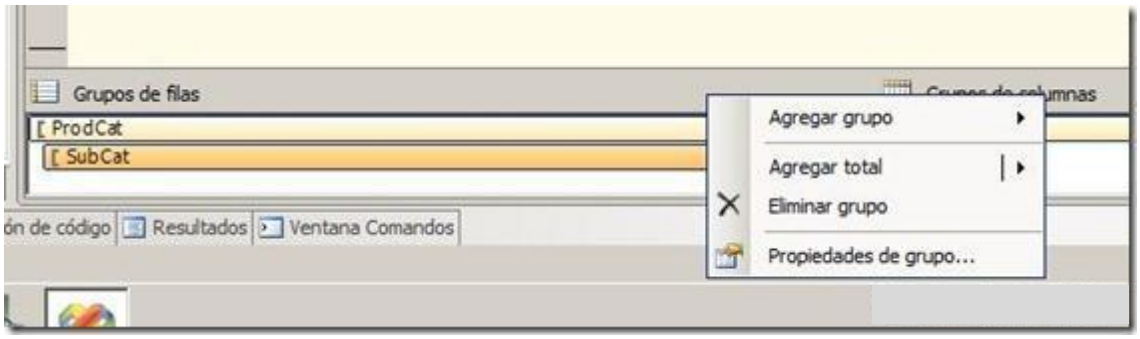


Figura 43. Menú de opciones para un grupo de filas

En la opción visibilidad, en el menú de la Caja de Diálogo seleccionamos si queremos mostrar u ocultar la columna, tal como observamos en la Figura 44:

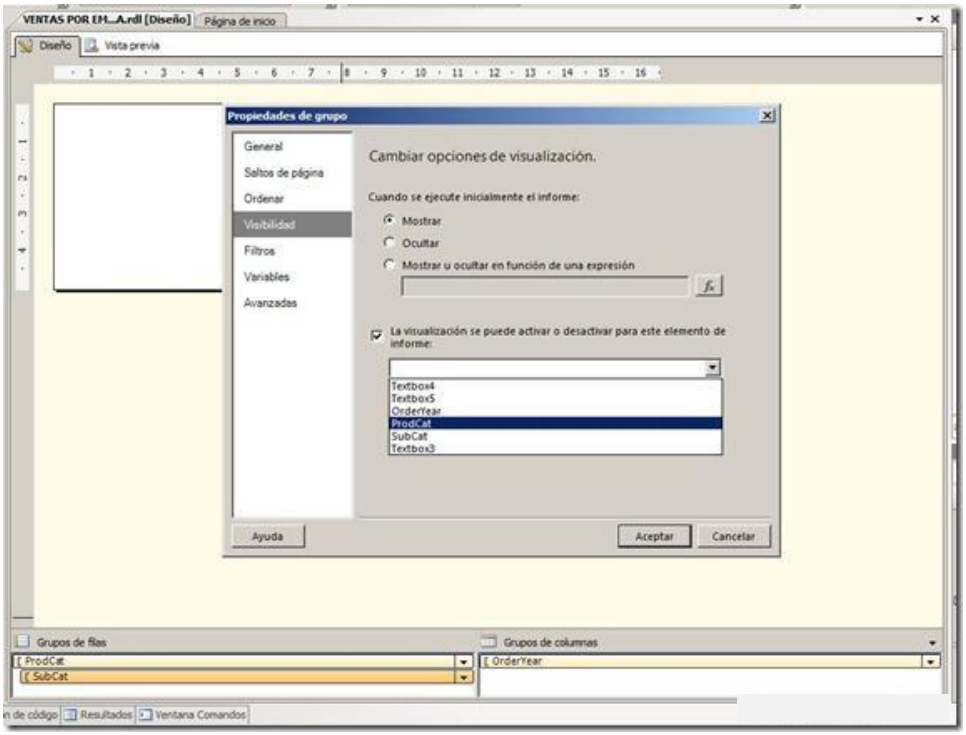


Figura 44. Seleccionando la opción visibilidad

Una vez realizado esto se puede desplegar y contraer información de un grupo en específico tal como se muestra en la Figura 45, esto fue especialmente útil en reportes con grandes cantidades de información.

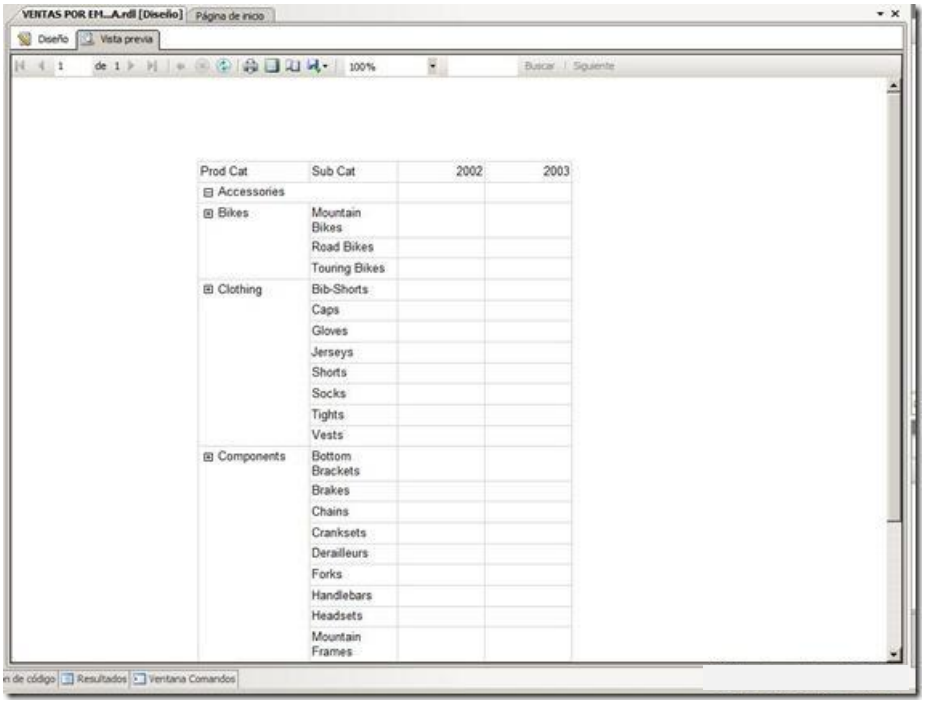


Figura 45. Ejemplo de grupos de filas que se contraen

Trabajar los conceptos anteriores a nivel de columnas, tal como se observa en la Figura 46, da los mismos efectos de visibilidad del Grupo y el Sub Grupo, de igual manera se puede contraer y expandir la información.

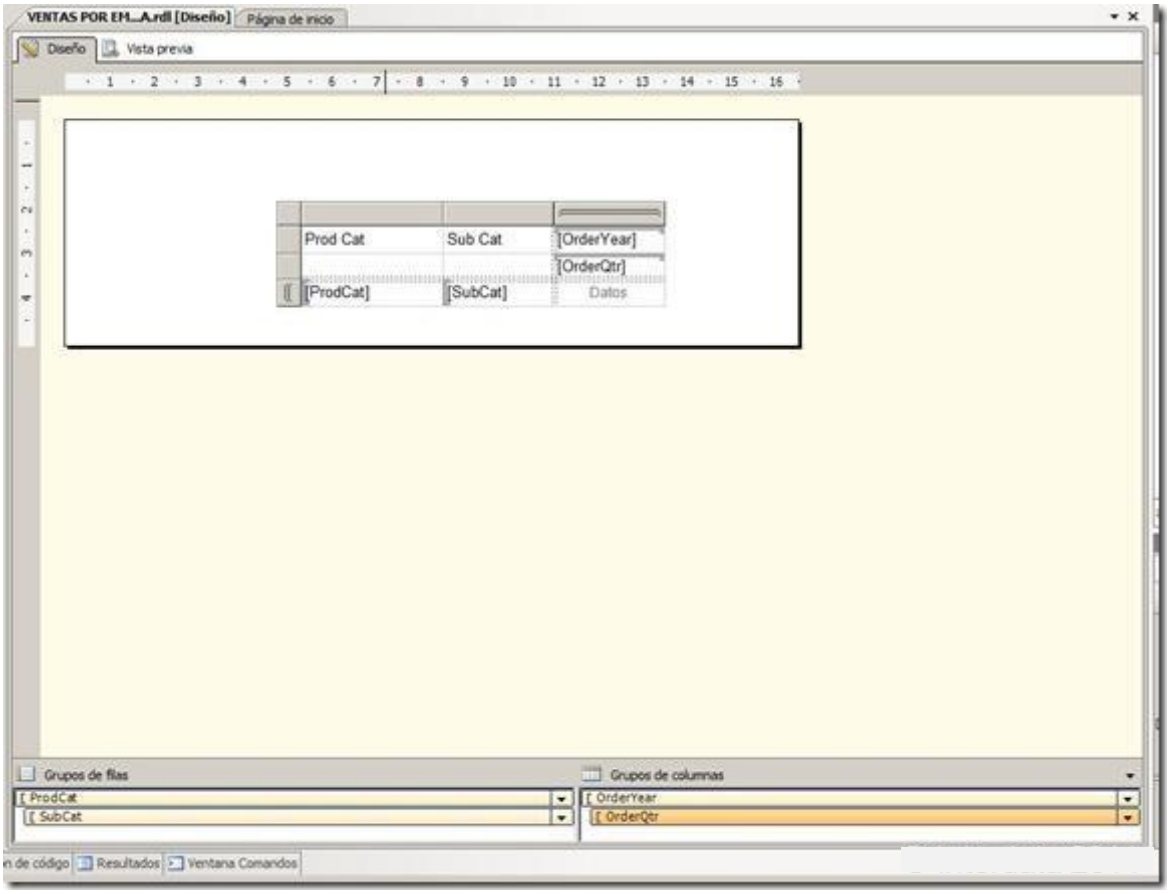


Figura 46. Agrupación por columnas

Podemos generar la vista previa de la misma forma como se mencionó anteriormente, en la Figura 47 observamos los resultados:

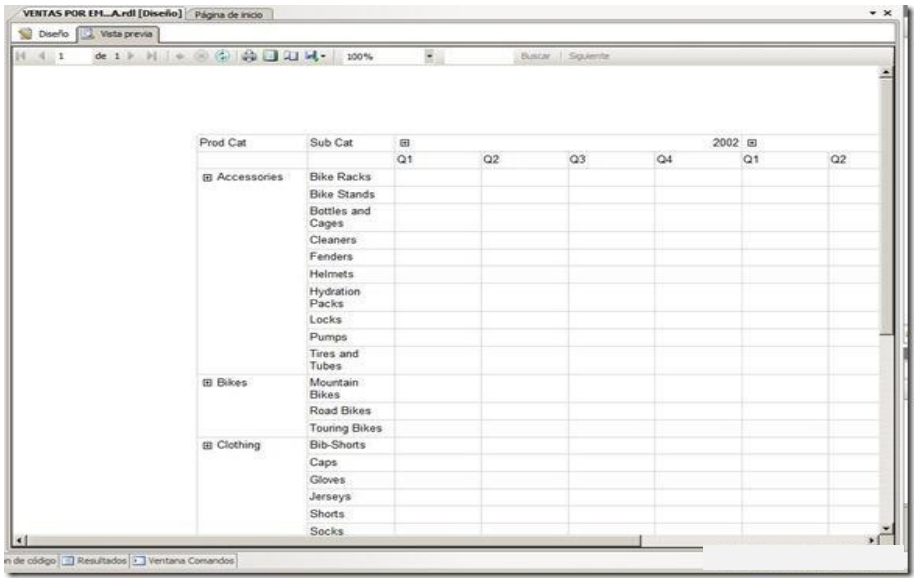


Figura 47. Ejemplo de columnas agrupadas que se contraen y expanden

Ahora se agregan los valores a visualizar en la matriz, para ello se arrastra desde los Orígenes de Datos el campo Sales a los datos de la Matriz, tal como se muestra en la Figura 48.

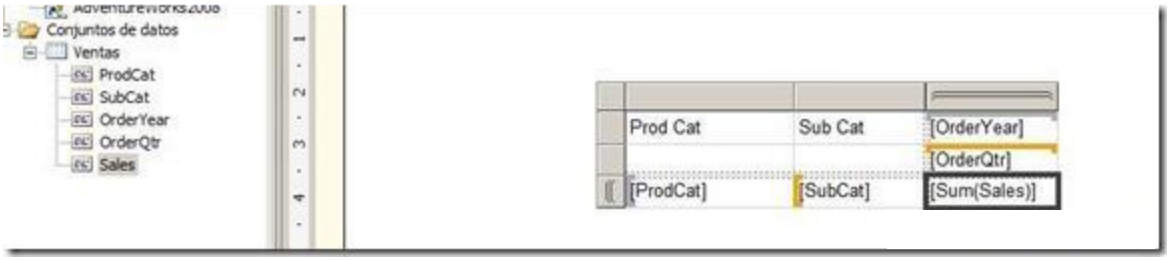
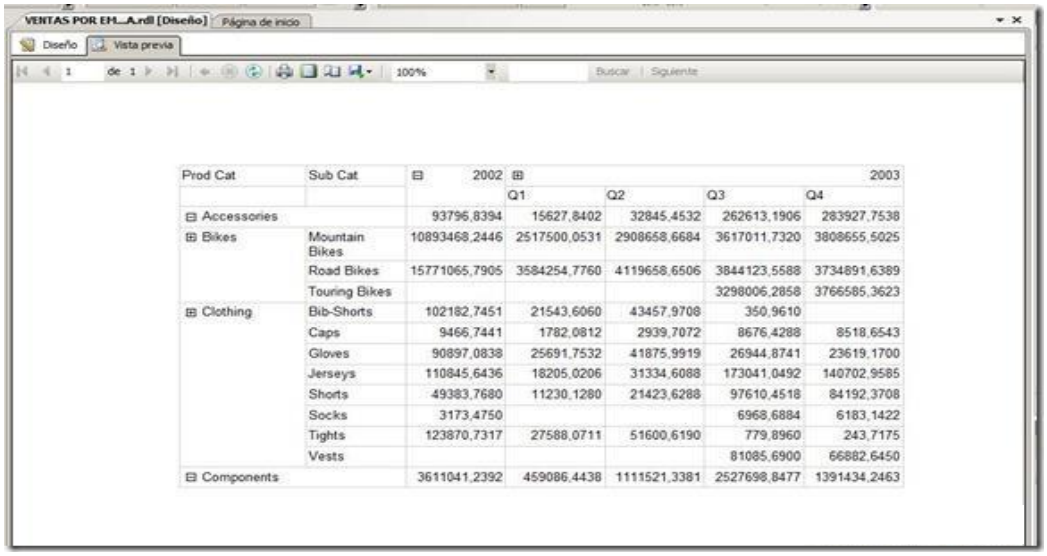


Figura 48. Agregando valores a la matriz

Y para verificar se vuelve a generar una Vista Previa.



Prod Cat	Sub Cat	2002	2003			
			Q1	Q2	Q3	Q4
Accessories		93796,8394	15627,8402	32845,4532	262613,1906	283927,7538
Bikes	Mountain Bikes	10893468,2446	2517500,0531	2908658,6684	3617011,7320	3808655,5025
	Road Bikes	15771065,7905	3584254,7760	4119658,6506	3844123,5588	3734891,6389
	Touring Bikes				3298006,2858	3766585,3623
Clothing	Bib-Shorts	102182,7451	21543,6060	43457,9708	350,9610	
	Caps	9466,7441	1782,0812	2939,7072	8676,4288	8518,6543
	Gloves	90897,0838	25691,7532	41875,9919	26944,8741	23619,1700
	Jerseys	110845,6436	18205,0206	31334,6088	173041,0492	140702,9585
	Shorts	49383,7680	11230,1280	21423,6288	97610,4518	84192,3708
	Socks	3173,4750			6968,6884	6183,1422
	Tights	123870,7317	27588,0711	51600,6190	779,8960	243,7175
	Vests				81085,6900	66882,6450
Components		3611041,2392	459086,4438	1111521,3381	2527698,8477	1391434,2463

Figura 49. Ejemplo de una agrupación por renglones y columnas

En el reporte de Monitoreo de captura (Figura 50) observamos como a partir de las agrupaciones se forman los renglones y columnas de manera dinámica.

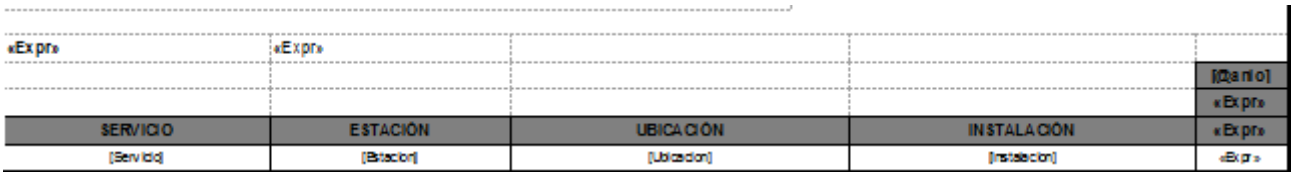


Figura 50. Agrupaciones del reporte monitoreo de captura.

Otra cosa a destacar de la figura 50 son las expresiones las cuales fueron fundamentales en la creación de reportes. Las expresiones se usan con frecuencia para controlar el contenido y la apariencia de los mismos.

Hay múltiples funciones que podemos usar en las expresiones, las más usadas en el proyecto fueron:

Funciones matemáticas

- La función Round es útil para redondear números al entero más cercano. La siguiente expresión redondea el valor 1,3 a 1:

  = Round(1.3)

Funciones de fecha

- La función Today proporciona la fecha actual. Esta expresión puede utilizarse en un cuadro de texto para mostrar la fecha en el reporte o puede utilizarse en un parámetro para filtrar los datos por la fecha actual.

  =Today()

- La función Year muestra el año correspondiente a una fecha determinada. Puede utilizarse para agrupar las fechas o para mostrar el año como etiqueta.

  =Year(Fields!OrderDate.Value)

Funciones de cadena

- La expresión siguiente devuelve dos campos, cada uno de ellos en una línea diferente del mismo cuadro de texto:

  =Fields!FirstName.Value & vbCrLf & Fields!LastName.Value

Funciones de decisión

- La función IIf devuelve un valor u otro en función de si la expresión es TRUE o no. En la expresión siguiente, se usa la función IIf para devolver el valor booleano True si el valor de LineTotal es mayor que 100. En caso contrario, devuelve False:

  =IIF(Fields!LineTotal.Value > 100, True, False)

- La función Switch resulta de gran utilidad cuando se necesita probar tres condiciones o más. La función Switch devuelve el valor asociado a la primera expresión en una serie que se evalúa como TRUE:

  =Switch(Fields!PctComplete.Value >= 10, "Green", Fields!PctComplete.Value >= 1, "Blue", Fields!PctComplete.Value = 1, "Yellow", Fields!PctComplete.Value <= 0, "Red",)

  Los valores mayores o iguales que 10 se muestran con un fondo verde, los valores entre 1 y 9 se muestran con un fondo azul, los valores iguales que 1 se muestran con un fondo amarillo, y los valores iguales o menores que 0 se muestran con un fondo rojo.

Funciones de informe

Sum

- La función Sum puede calcular el total de los valores de un grupo o de una región de datos. Esta función puede resultar útil en el encabezado o en el pie de página de un grupo. La expresión siguiente muestra la suma de los datos del grupo o de la región de datos Order:

  =Sum(Fields!LineTotal.Value, "Order")

RowNumber

- La función RowNumber, cuando se utiliza en un cuadro de texto de una región de datos, muestra el número de fila de cada instancia del cuadro de texto en que aparece la expresión. Esta función puede ser de utilidad para numerar las filas de una tabla. También puede resultar útil para tareas más complejas, como proporcionar saltos de página según el número de filas.

  =RowNumber(Nothing)

CAPÍTULO 4
RESULTADOS

Finalmente se muestran los resultados obtenidos en este módulo de reportes

4.1 Reporte Usuarios y Perfiles

Este reporte permite consultar todos los usuarios para el sistema con acceso a las opciones.

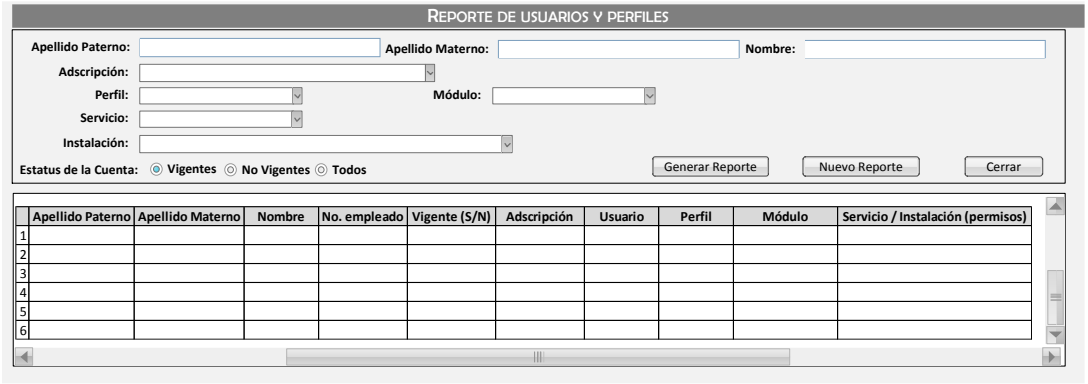


Figura 50. Reporte usuarios y perfiles

4.2 Reporte Anexo Técnico

El reporte presenta los datos de los anexos técnicos. (Figura 51)

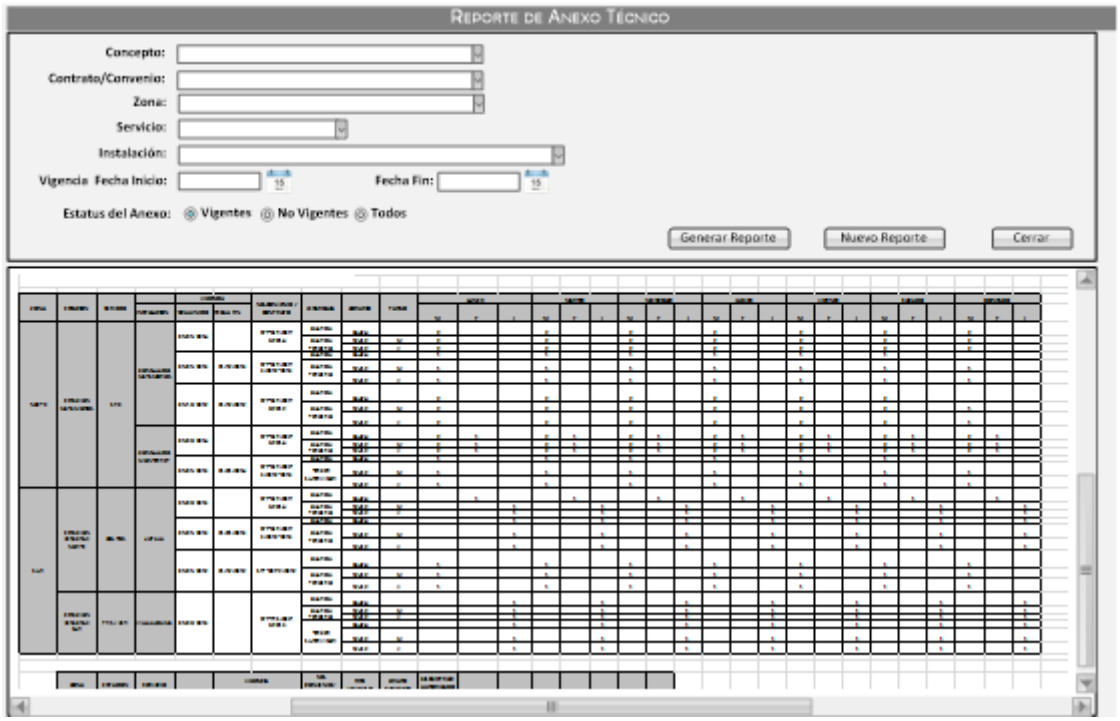


Figura 51. Reporte anexo técnico

4.3 Reporte Servicios

Reporte que presenta toda la información de los clientes o servicios que tiene contratados la dependencia de gobierno. (Figura 52)

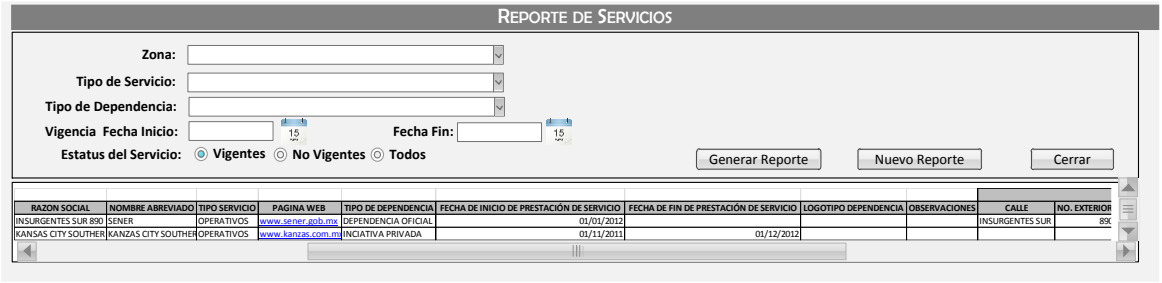


Figura 52. Reporte servicios

4.4 Reporte Instalaciones

Reporte que muestra los datos de las instalaciones las que se proporciona servicio. (Figura 53)

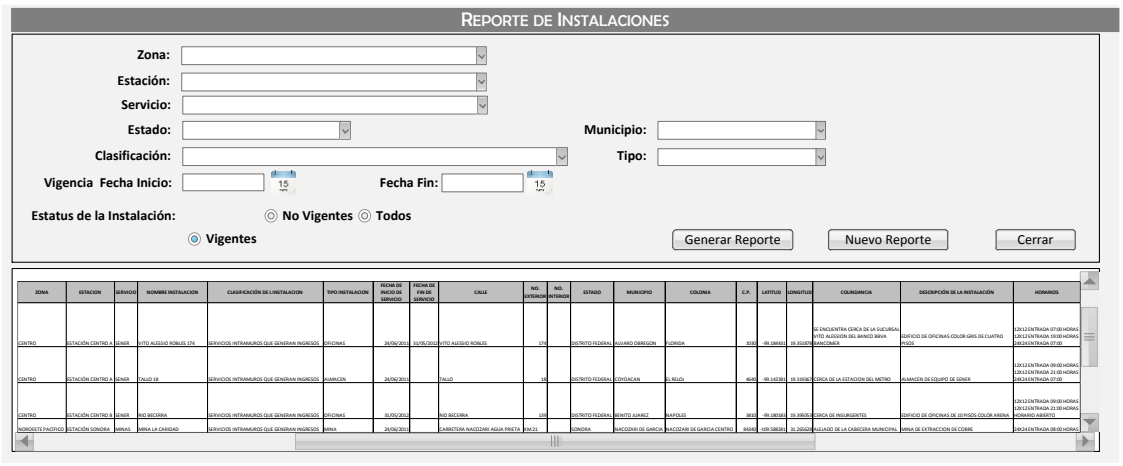


Figura 53. Reporte instalaciones

4.5 Reporte Diario de Servicio Montado

Reporte que despliega la información diaria referente al servicio montado en cada instalación.

Esta opción se presenta sólo si el usuario tiene los permisos necesarios para la ejecución del reporte. (Figura 54)

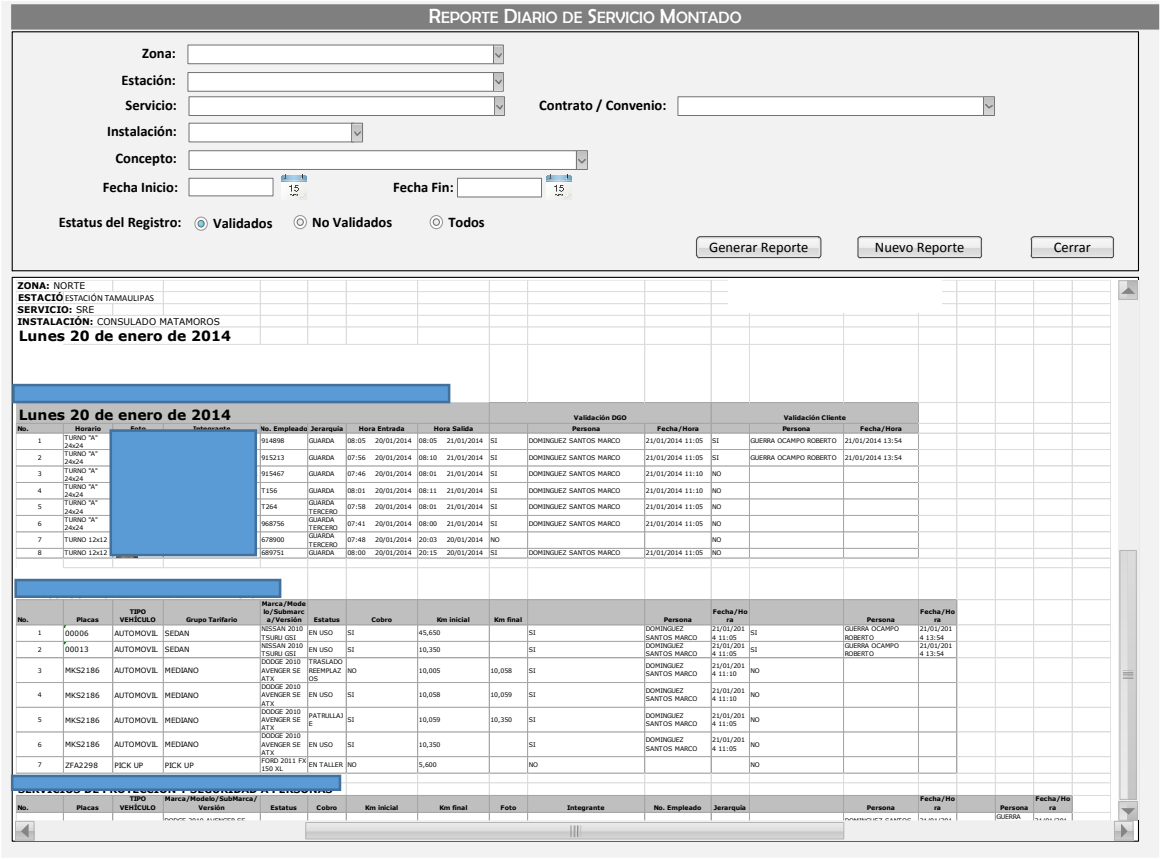


Figura 54. Reporte diario de servicio montado

4.6 Reporte Turnos Cumplidos

Reporte que muestra totales de los servicios montados de manera diaria. (Figura 55)

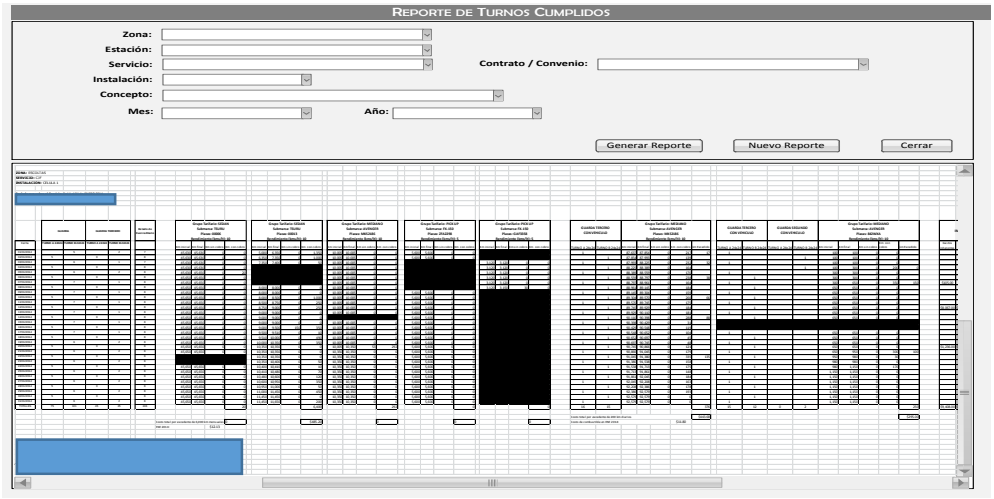


Figura 55. Reporte turnos cumplidos

4.7 Reporte para Monitoreo de Captura

Reporte que muestra si se realizó la captura de registro de presencia, y desde donde se realiza la captura. (Figura 56)

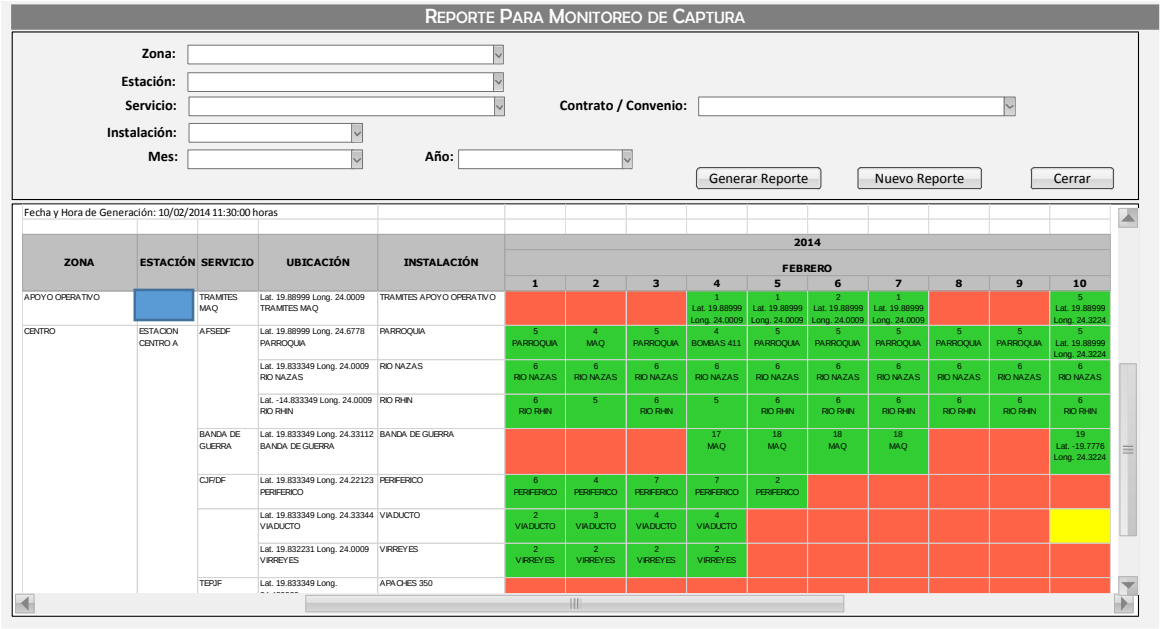


Figura 56. Reporte para monitoreo de captura

4.8 Reporte Horarios

Reporte que despliega los horarios registrados en la base de datos. (Figura 57)

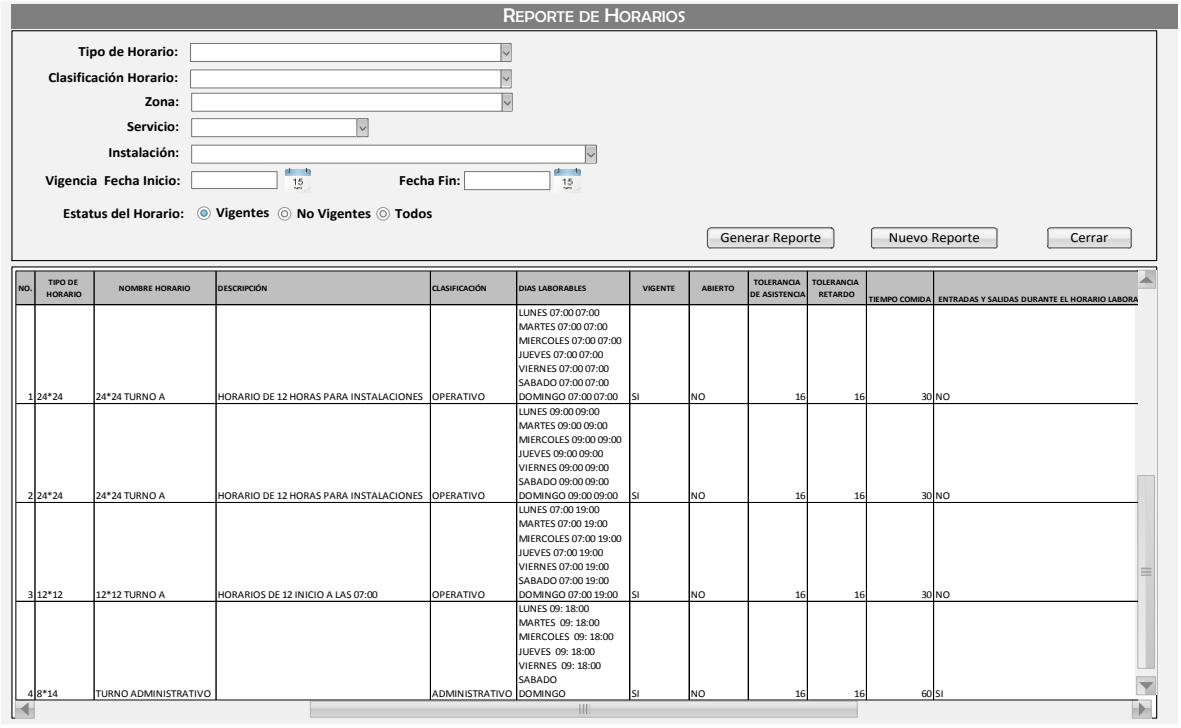


Figura 57. Reporte de horarios

4.3 Reporte Correcciones Realizadas

Reporte que despliega las correcciones realizadas por la dependencia a los registros de presencia. (Figura 58)

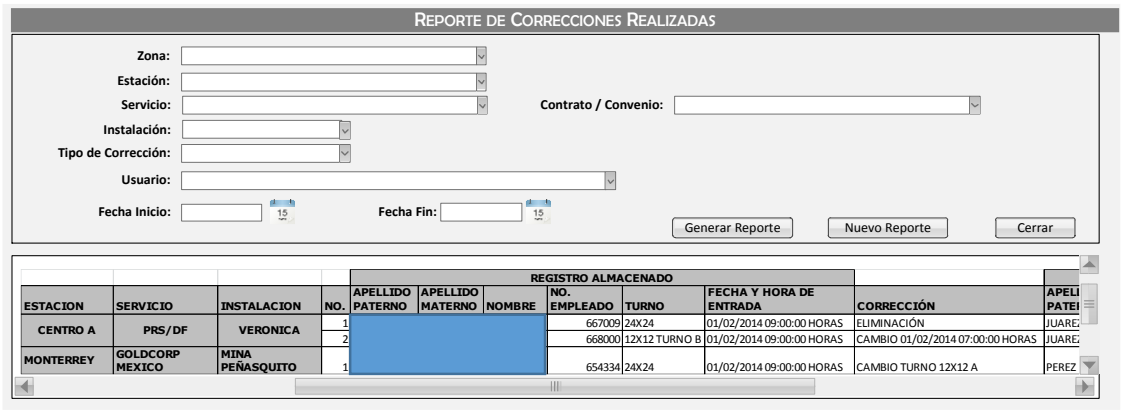


Figura 58. Reporte correcciones realizadas

4.10 Reporte de Rendimiento de Vehículos

Reporte que despliega el rendimiento de los vehículos. (Figura 59)

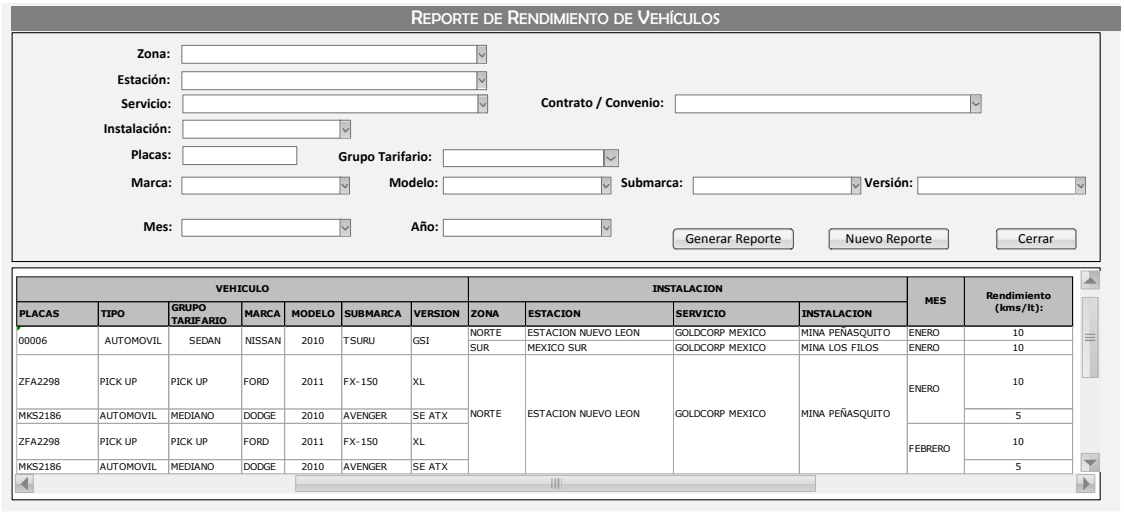


Figura 59. Reporte de rendimiento de vehículos

4.11 Reporte de Asignaciones de Vehículos

Reporte que despliega las asignaciones de vehículos. (Figura 60)

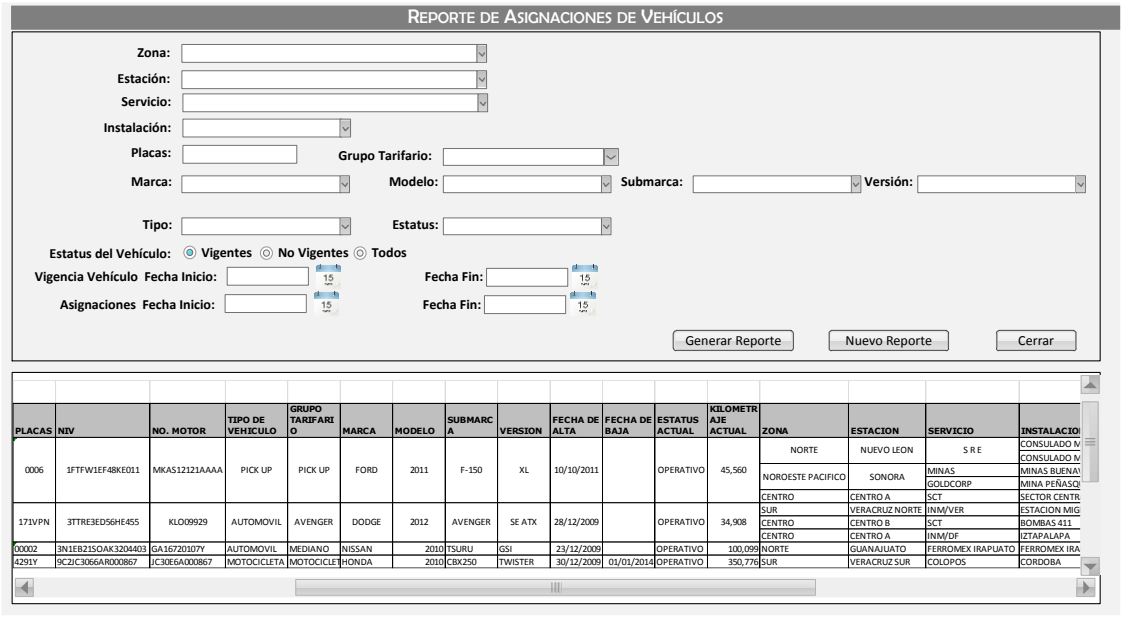


Figura 60. Reporte de asignaciones de vehículos

Conclusiones

La participación en el proyecto amplió mi visión respecto a las necesidades de las empresas, que aun cuando son diferentes, comparten ciertas áreas de oportunidad para los Ingenieros en Computación, también me permitió conocer un poco más sobre Business Intelligence (BI) el cual es de suma importancia en cualquier organización que desee mejorar sus procesos o tomar decisiones sobre el rumbo de la misma.

En este proyecto pude aplicar conocimientos adquiridos en materias tales como: bases de datos, Ingeniería de software, programación avanzada y negocios electrónicos.

Uno de los retos más grandes que tuve fué aprender a trabajar en equipo con las diferentes áreas que conformaban el proyecto. Tener que entregar diferentes resultados bajo tiempos estipulados me permitió darme cuenta de la importancia de la organización y buena comunicación con los compañeros de trabajo.

Quede satisfecho con el resultado del módulo el cual brinda la facilidad de crear *interfaces* en las que el cliente puede monitorear la actividad diaria de la dependencia.

El moduló puede presentar algunas mejoras tales como la utilización de un solo procedimiento almacenado para cada reporte. Los reportes dependen directamente de la base de datos por lo cual mientras más optimizadas sean las consultas, más rápido será el reporte.

También es importante la seguridad, por lo cual es importante que además de la autenticación por medio de la aplicación web, se provea de una interfaz que controle la visualización del reporte de acuerdo a las credenciales del usuario.

A lo largo de mi experiencia profesional he trabajado con Reporting Services, por lo cual además de lo comentado en este informe, podría proveer de graficas dinámicas a los reportes, que esquematicen de mejor manera la información, teniendo así un módulo mucho más intuitivo y fácil de usar.

La capacidad para tomar decisiones de negocio precisas y de forma rápida se ha convertido en una de las claves para que una empresa llegue al éxito. Sin embargo, los sistemas de información

tradicionales (como la mayoría de los programas de gestión, las aplicaciones a medida, e incluso los ERP más sofisticados), suelen presentar una estructura muy inflexible para este fin. Aunque su diseño se adapta con mayor o menor medida para manejar los datos de la empresa, no permite obtener la información de los mismos, y mucho menos extrapolar el conocimiento almacenado en el día a día de las bases de datos.

Para superar todas estas limitaciones, el Business Intelligence se apoya en un conjunto de herramientas que facilitan la extracción, la depuración, el análisis y el almacenamiento de los datos generados en una organización, con la velocidad adecuada para generar conocimiento y apoyar la toma de decisiones de los directivos y los usuarios oportunos.

No es que los productos de BI sean mejores que las aplicaciones actuales: se trata de sistemas con objetivos distintos, eficientes en sus respectivas ramas, pero que deben complementarse para optimizar el valor de los sistemas de información.

Es por eso que la creación de módulos que reporten la actividad diaria de la empresa es indispensable, ya que facilitan el proceso de análisis de la información, agilizando de esta manera la toma de decisiones y la implementación de Business Intelligence.

Referencias.

Microsoft. (2014). Reporting Services (SSRS). 02/02/2015, de Librería Microsoft Sitio web: https://msdn.microsoft.com/es-MX/library/ms159106.aspx

Microsoft. (2014). Accessing the Reporting Services Web Service Using Visual Basic or Visual C#. 04/02/2015, de Librería Microsoft Sitio web: https://technet.microsoft.com/en-us/library/aa237438%28v=sql.80%29.aspx

Microsoft. (2014). Tutoriales de Reporting Services. 06/02/2015, de Librería Microsoft Sitio web: https://msdn.microsoft.com/es-MX/library/bb522859.aspx

Rod Paddock. (2014). Integrating .NET Code and SQL Server Reporting Services. 06/02/2015, de Codemag Sitio web: http://www.codemag.com/Article/0701061

Glosario de terminos

Base de datos: conjunto de datos organizados de modo tal que resulte fácil acceder a ellos, gestionarlos y actualizarlos.

Cliente/servidor: este término define la relación entre dos programas de computación en el cual uno, el cliente, solicita un servicio al otro, el servidor, que satisface el pedido.

HTML: Hyper Text Mark-up Language. Lenguaje de programación para armar páginas web.

HTTP: Hypertext Transfer Protocol. Protocolode transferencia de hipertextos. Es un protocolo que permite transferir información en archivos de texto, gráficos, de video, de audio y otros recursos multimedia.

Página web: una de las páginas que componen un sitio de la WorldWideWeb. Un sitio web agrupa un conjunto de páginas afines. A la página de inicio se la llama "home page".

Servidor: computadora central de un sistema de red que provee servicios y programas a otras computadoras conectadas.

Software: término general que designa los diversos tipos de programas usados en computación.

SQL: Structured Query Language. Lenguaje de programación que se utiliza para recuperar y actualizar la información contenida en una base de datos. Fue desarrollado en los años 70 por IBM. Se ha convertido en un estándar
ISO y ANSI.

World Wide Web: red mundial; telaraña mundial. Es la parte multimedia de Internet. Es decir, los recursos creados en HTML y sus derivados. Sistema de información global desarrollado en 1990 por Robert Cailliau y Tim Berners-Lee en el CERN (Consejo Europeo para la Investigación Nuclear).
Con la incorporación de recursos gráficos e hipertextos, fue la base para la explosiva popularización de Internet a partir de 1993.

U*nix*: sistema operativo multiusuario, fue muy importante en el desarrollo de Internet.

Web Service: es un método de comunicación entre dos dispositivos electrónicos, generalmente con el fin de intercambiar información

Reporting Services: Es un software de generación de reportes que ofrece una gama completa de herramientas y servicios listos para usar que monitoreas la actividad diaria. Forma parte de la suite de Bussines Intelligence de Microsoft

ORM: Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

XML: Es un lenguaje de marcas utilizado para almacenar datos en forma legible. Deriva del lenguaje HTML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes

RDL: Es una representación XML de una definición de informe de SQL Server Reporting Services. Una definición de informe contiene información acerca de la recuperación y el diseño de los datos de un informe

Visual Studio: Es un entorno (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP; al igual que entornos de desarrollo web.

IDE: Es una aplicación de software, que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador.

CSS: Es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML.

Datawarehouse: Es una base de datos corporativa que se caracteriza por integrar y depurar información de una o más fuentes distintas, para luego procesarla permitiendo su análisis desde infinidad de perspectivas y con grandes velocidades de respuesta.

MVC: Es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

Framework: Define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Razor: Es un motor de vistas que proporciona una sintaxis de programación simple para escribir código en páginas web donde el código basado en servidor se incrusta en el formato HTML de las páginas web.

Entity Framework: Es un ORM que permite a los desarrolladores de .NET trabajar con datos relacionales usando objetos específicos del dominio. Elimina la necesidad de la mayor parte del código de acceso a datos que los desarrolladores suelen tener que escribir.

WCF: Es un marco de trabajo para la creación de aplicaciones orientadas a servicios. Con WCF, es posible enviar datos como mensajes asincrónicos de un extremo de servicio a otro. Un extremo de servicio puede formar parte de un servicio disponible continuamente hospedado por IIS, o puede ser un servicio hospedado en una aplicación.

Bootstrap: Es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS.