**Arquitectura orientada al procesamiento y aprovechamiento de datos abiertos**

En el marco de la tecnología para los datos abiertos es pertinente tener en cuenta los elementos y/o dispositivos que producen o generan datos. Con la aparición de tecnologías como Big Data y IoT, consideraremos una Arquitectura base que soporte la recolección y extracción de datos, tratamiento y análisis de los datos, y finalmente, la visualización de los datos de forma consolidada y agrupada para la toma de decisiones.

Esta Arquitectura se define en capas lógicas verticales y cada una de ellas realiza capacidades específicas dentro del esquema tecnológico que utiliza principalmente plataformas Cloud.

En el siguiente diagrama se muestra la Arquitectura base y sus componentes.



En el siguiente cuadro se describen los componentes propuestos.

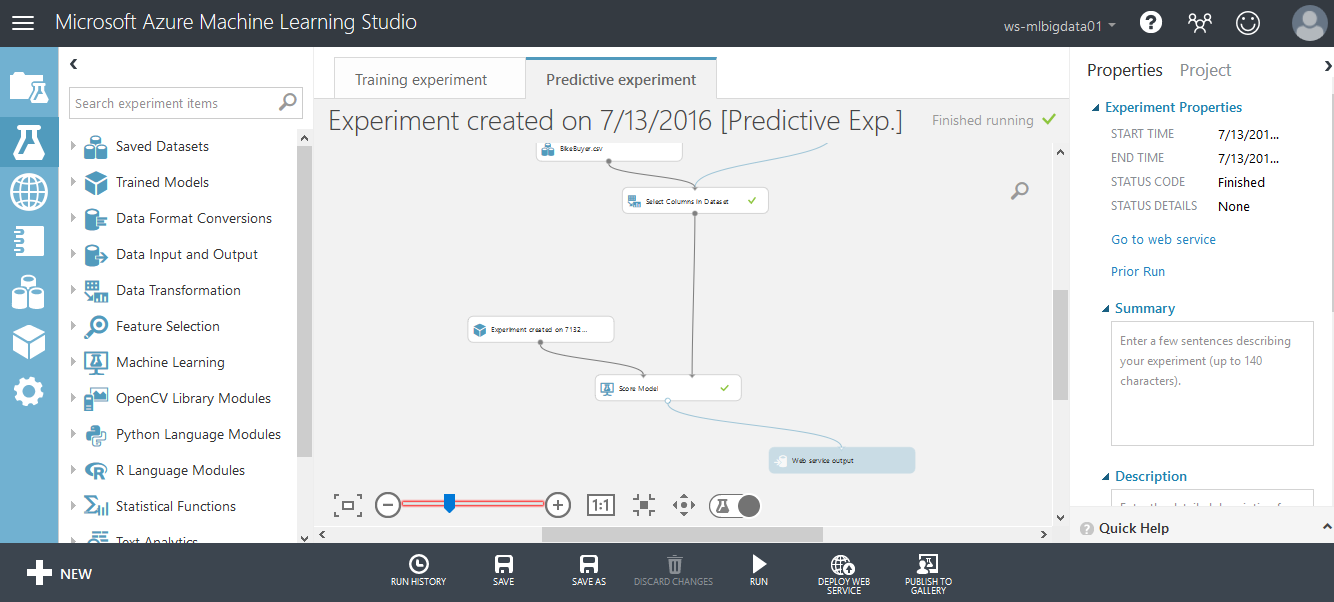
|  |  |  |
| --- | --- | --- |
| **Capa Lógica** | **Componente/Tecnología** | **Propósito** |
| **Productores de datos** | **API de datos** | Permiten acceder a servicios de datos/datasets a través de REST |
| **Servicios Web de datos** | Permiten exponer datasets en formato XML |
| **Sensores** | Monitorean sucesos del mundo real, capturan datos y permiten enviarlos a través de Internet |
| **Transporte** | **Cloud Gateway** | Permite realizar la recolección de los datos que se capturan a través de eventos (eventos/segundo) producidos en los dispositivos y aplicaciones |
| **Transformación** | **Stream Processors** | Este componente permite procesar eventos provenientes desde los productores en tiempo real |
| **Hadoop** | Este componente es la implementación que permite el procesamiento de grandes volúmenes de datos |
| **Análisis de datos** | **Machine Learning Model** | Este componente brinda capacidades de analítica de datos aplicando algoritmos y modelos matemáticos, además, puede publicar los resultados de los modelos a través de Web Services |
| **Almacenamiento** | **Contenedores/Repositorios** | Son los repositorios permanentes de almacenamiento de los datos estructurados y no estructurados |
| **Base de Datos** |
| **Visualización/Toma de decisiones** | **Web Applications** | Componente utilizado para presentar la información con acceso a dispositivos como PCs y Laptops. |
| **Mobile Applications** | Componente utilizado para presentar la información desde un dispositivo móvil |

Por qué Microsoft Azure

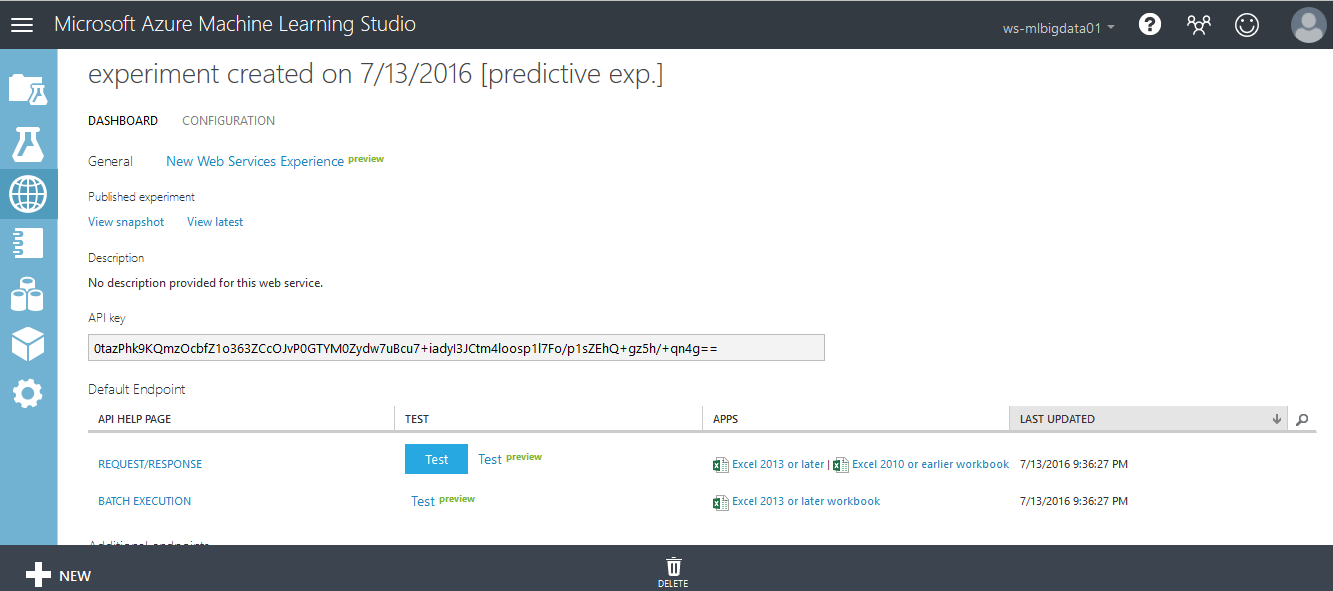
* Es la plataforma Cloud con mayor tiempo en el mercado
* Cuenta con el mayor número de servicios para ambientes Cloud
* Fácil acceso, uso y consumo de servicios incluyendo la diferentes suscripciones y opciones de pago
* Es una plataforma que permite una amplia flexibilidad y escalamiento de la infraestructura, arquitectura, aplicaciones y servicios
* Satisface los estándares internacionales en seguridad y protección de los datos
* Contiene herramientas integradas para el desarrollo y gestión de aplicaciones empresariales, Web, móviles, IoT, Big Data y analítica de datos.

Análisis de datos

El componente Machine Learning dentro de la arquitectura propuesta se puede implementar utilizando soluciones SaaS (Software as a Service) en plataformas Cloud. Para este caso, se utilizó Azure Machine Learning que tiene un entorno de desarrollo que permite crear modelos que son entrenados y ajustados con los algoritmos seleccionados con base a los criterios aplicados sobre los datos.

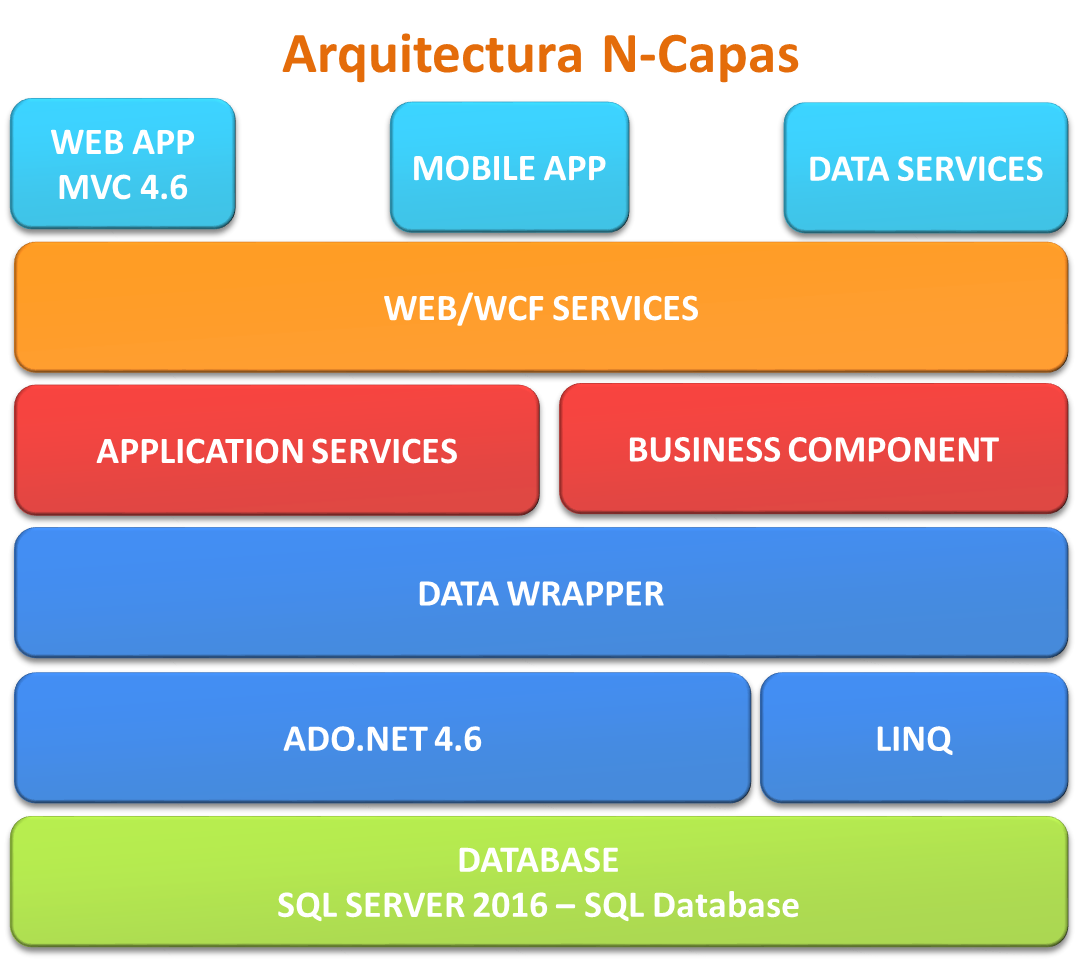


El resultado del modelo puede ser explotado con la opción de generar un Web Service (REST) y así, dicha información pueda ser consumida por una aplicación de negocio, aplicación móvil, entre otros.



Visualización/Experiencia de Usuario

La arquitectura de la Aplicación Web se representa en el siguiente diagrama en capas



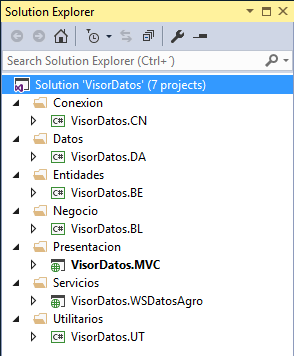
Especificación

La solución tiene las siguientes capas lógicas que interactúan entre sí como lo representa el diagrama de paquetes que se visualiza en seguida:

* Capa de conexión a datos (Librería de clases)
* Capa de lógica de datos (Librería de clases)
* Capa de lógica de Negocio (Librería de Clases)
* Capa de Presentación ([ASP.NET](http://asp.net/) MVC)
* Capa de Utilidades (Librería de Clases)
* Capa de Servicio ([ASP.NET](http://asp.net/) con XML Web Services)



La solución se desarrolló en Visual Studio 2015. A continuación se muestra la implementación de la estructura de la solución

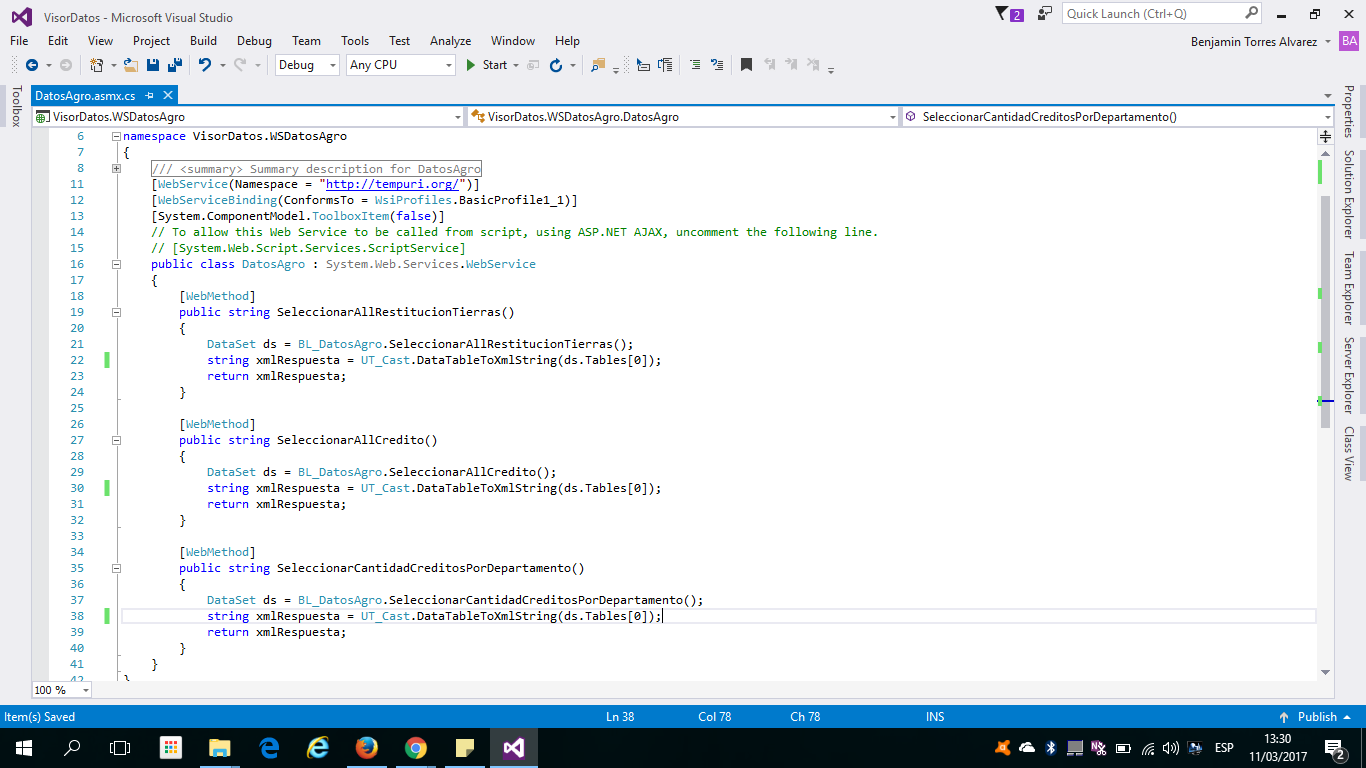


En la capa de presentación se usa Google Charts para generar los elementos gráficos del dashboard. En el prototipo, la carga de datos en el Dashboard se realiza a través de Stored Procedures (Capa acceso a datos) que para el ejemplo Restitución de Tierras se denomina *Agro\_SeleccionarAllRestitucionTierras* y para Créditos por Departamento es *Agro\_CantidadCreditosPorDepartamento*, y XML Web Services (asmx) que se denomina *WSDatosAgro* (Capa Servicios)

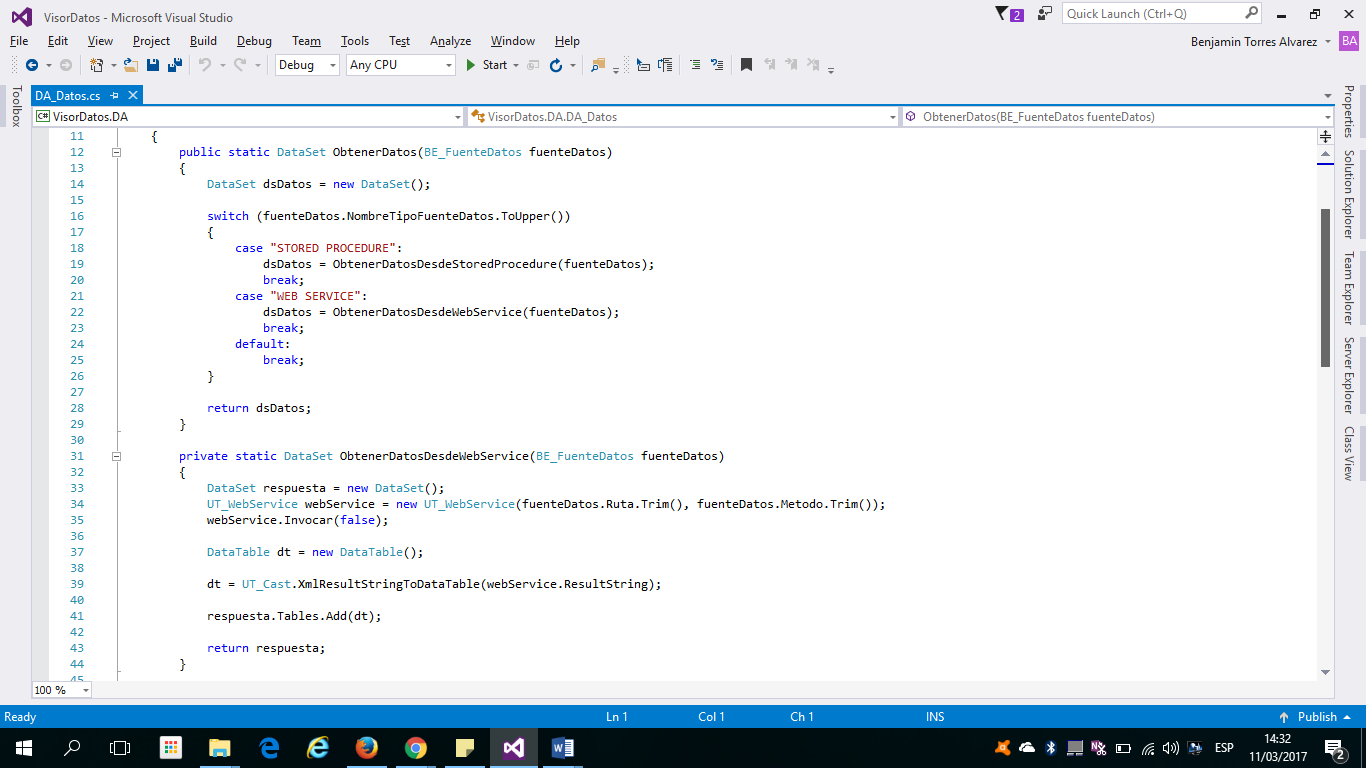
La solución contempla la administración de usuarios, los tipos de fuente de datos y las fuentes de datos. En la base de datos, se registraron fuentes tomadas del portal [datos.gov.co](http://datos.gov.co/)

Imágenes de Código Fuente

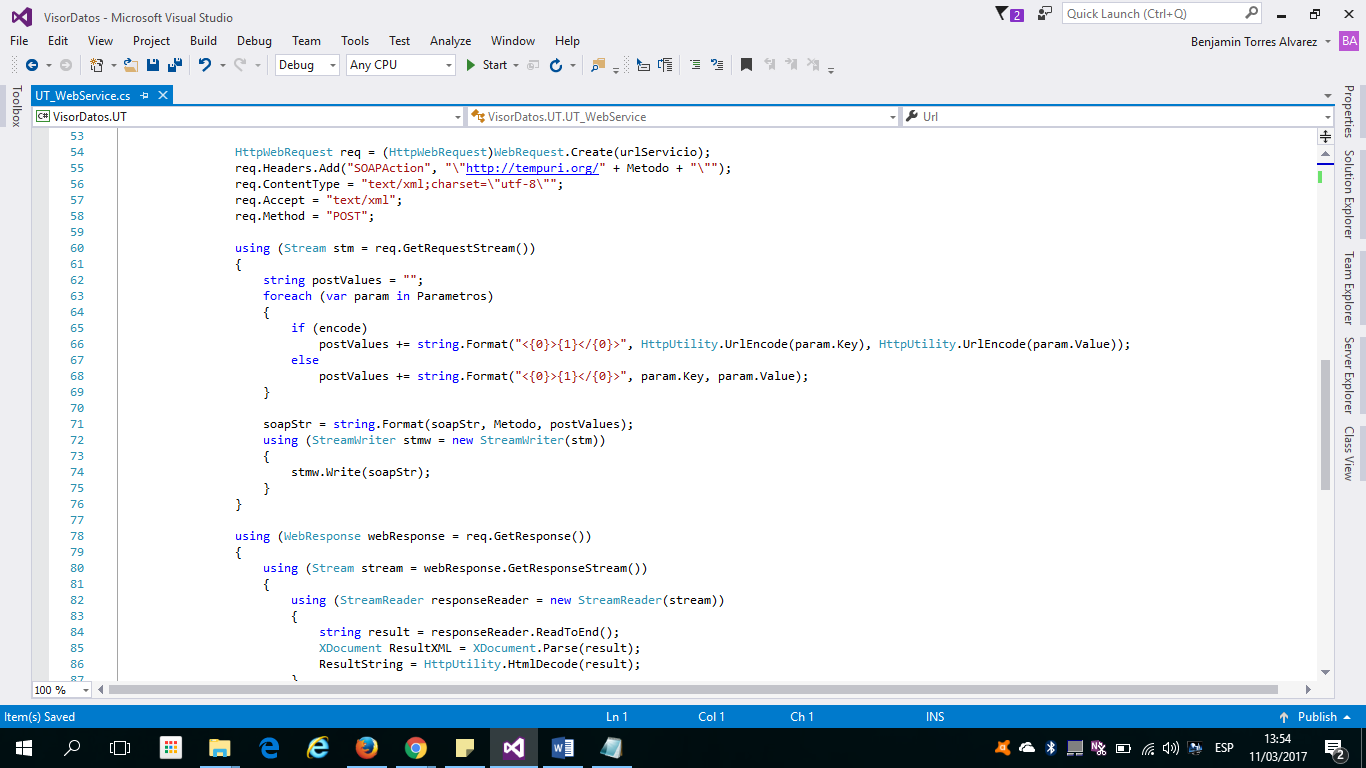
DatosAgro.asmx.cs



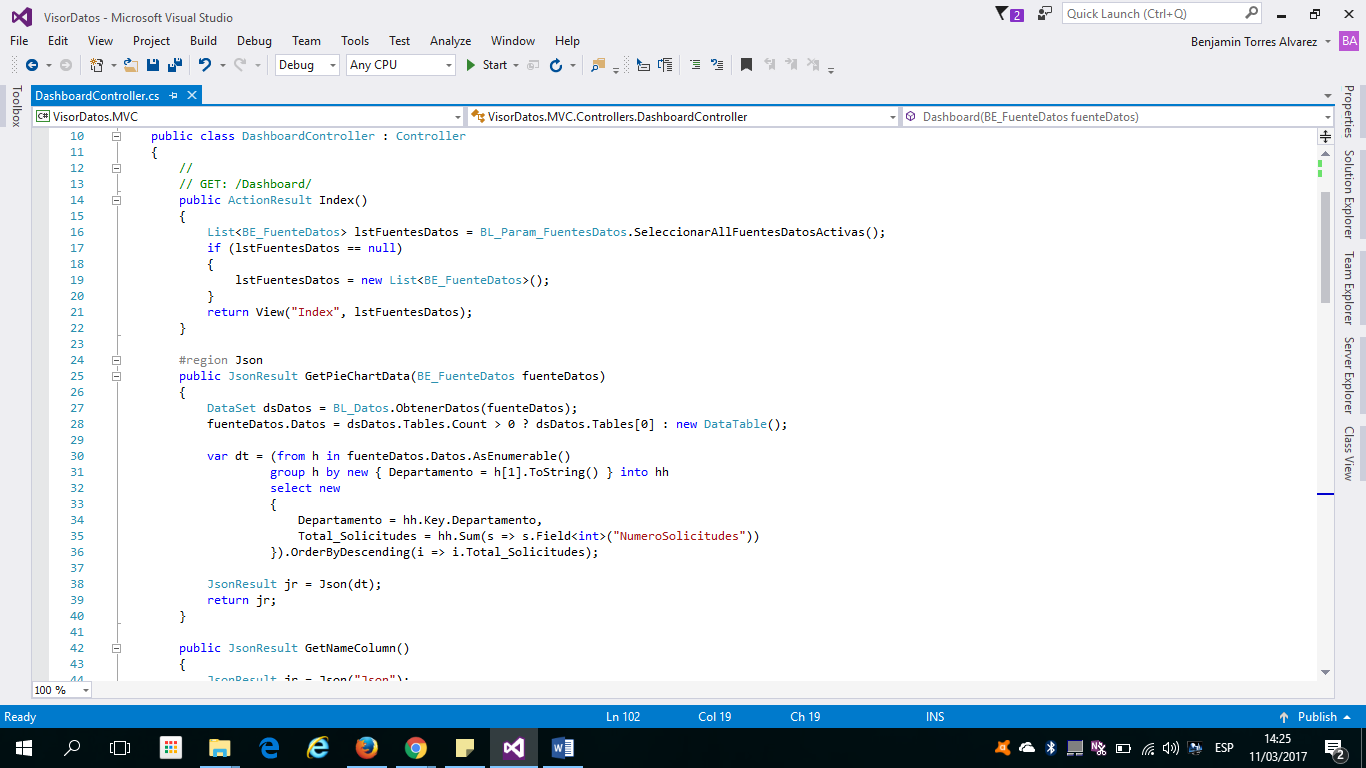
DA\_Datos.cs



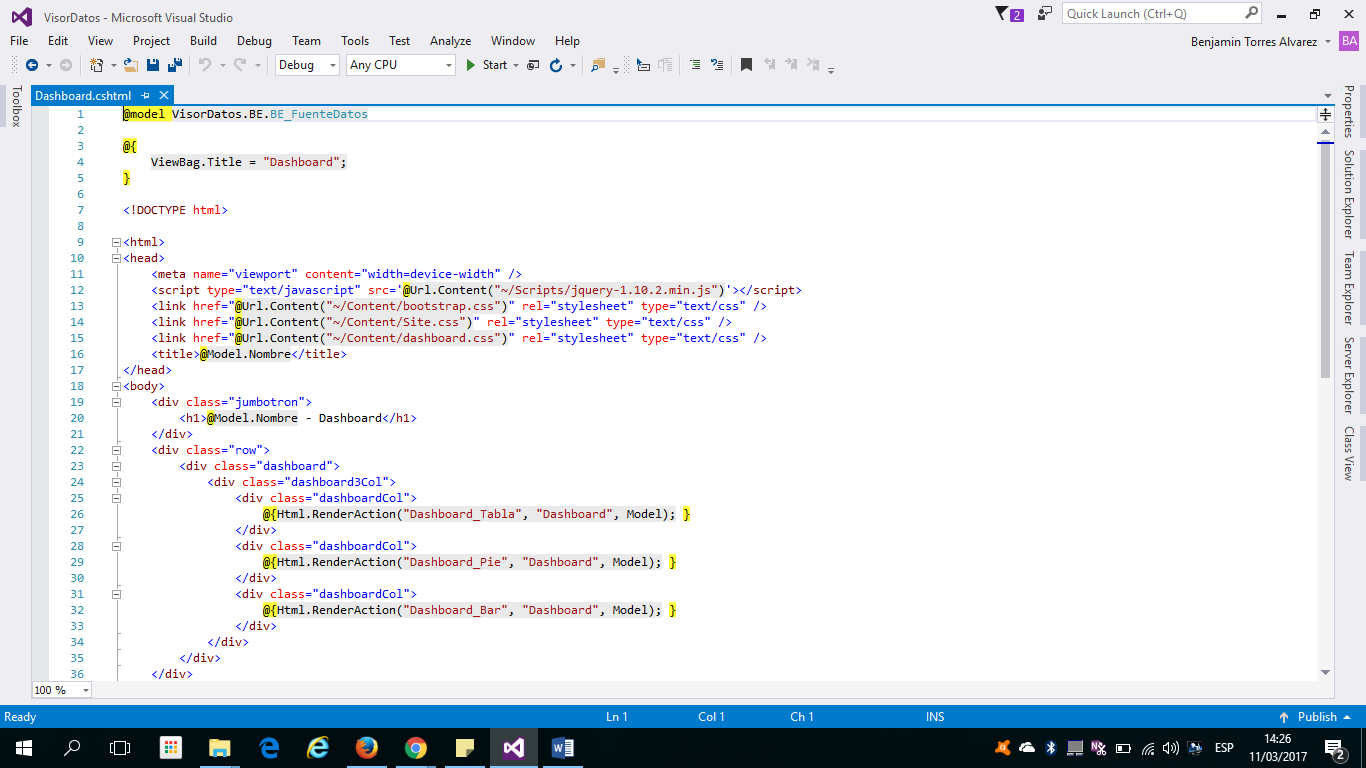
UT\_WebService.cs



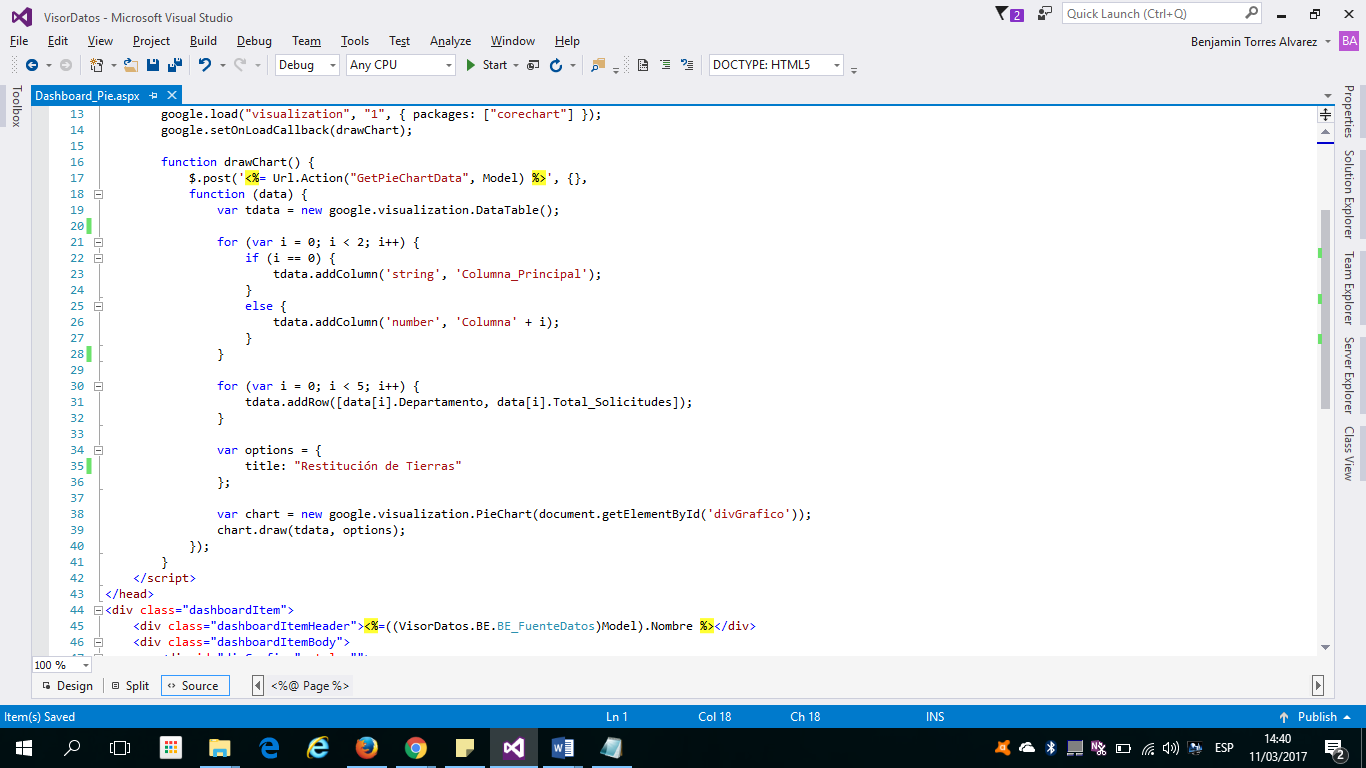
DashboardController.cs



Dashboard.cshtml

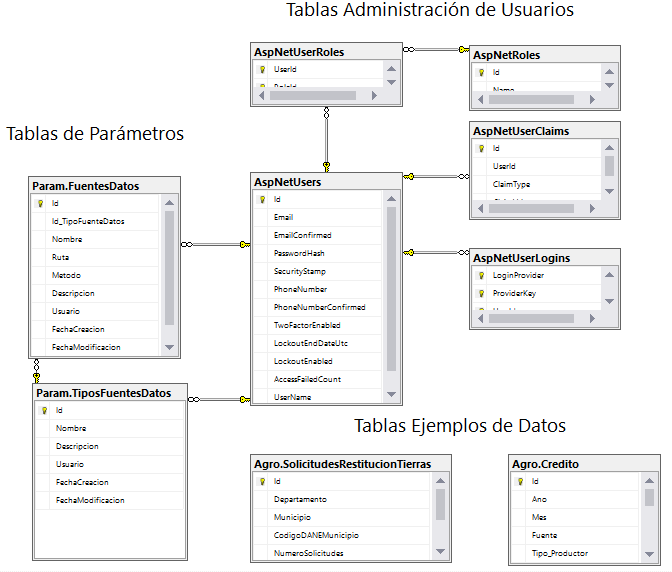


Dashboard\_Pie.aspx



Modelo de Datos

El siguiente diagrama representa el modelo de datos de la Aplicación Web



Este modelo se implementó utilizando SQL Server 2016.

|  |  |
| --- | --- |
| **TABLA** | **DESCRIPCIÓN** |
| AspNetRoles AspNetUserRoles AspNetUserClaims AspNetUserLogins AspNetUsers | Estas tablas conforman el esquema de seguridad para acceso y control de los usuarios |
| Param.TiposFuentesDatos | Esta tabla permite administrar los tipos de fuentes de datos del visor de datos |
| Param.FuenteDatos | Esta tabla permite registrar las fuentes de datos configuradas por el usuario |
| Agro.SolicitudesRestitucionTierrasAgro.Credito | Estas permiten almacenar datos procesados a través de las distintas fuentes como Web Services, análisis de datos con Machine Learning, y procesadores de flujo eventos y datos |

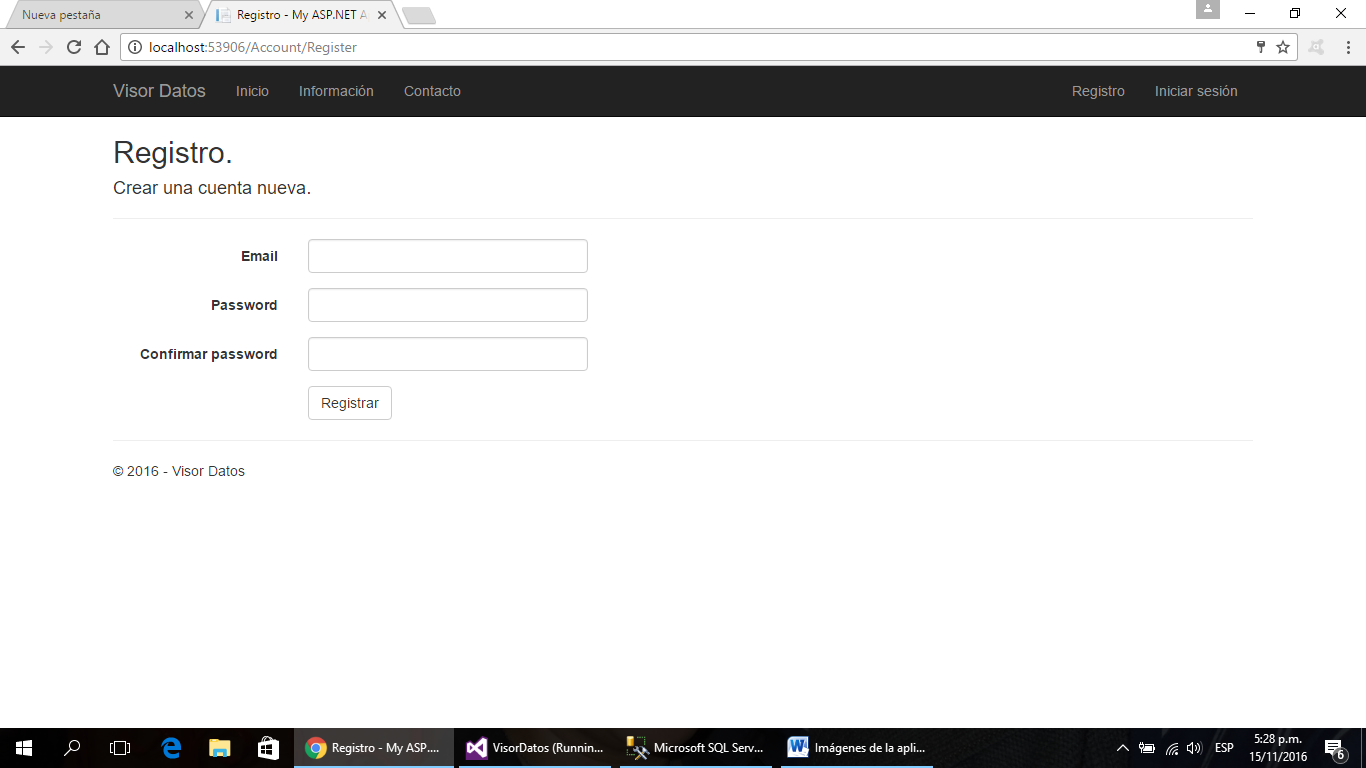
Vistas

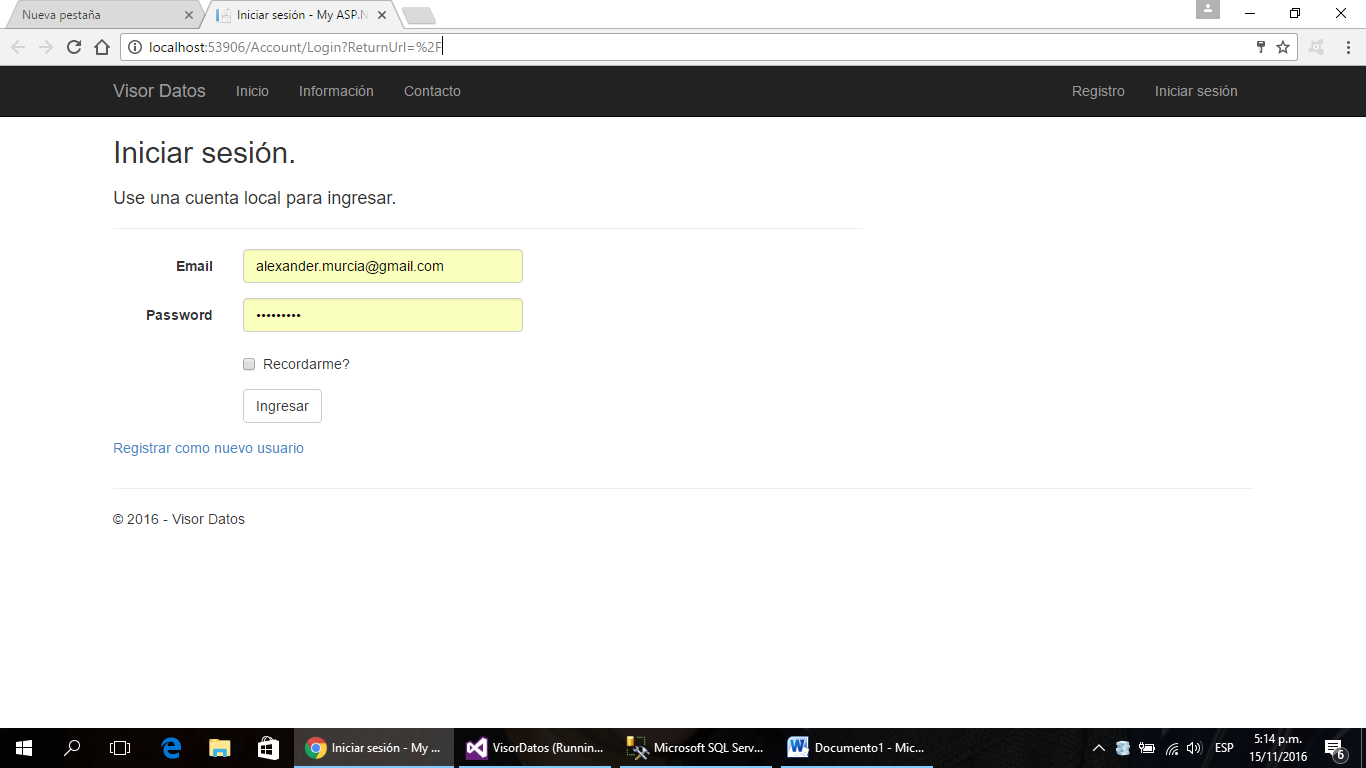
Con la implementación del prototipo se generaron las siguientes vistas

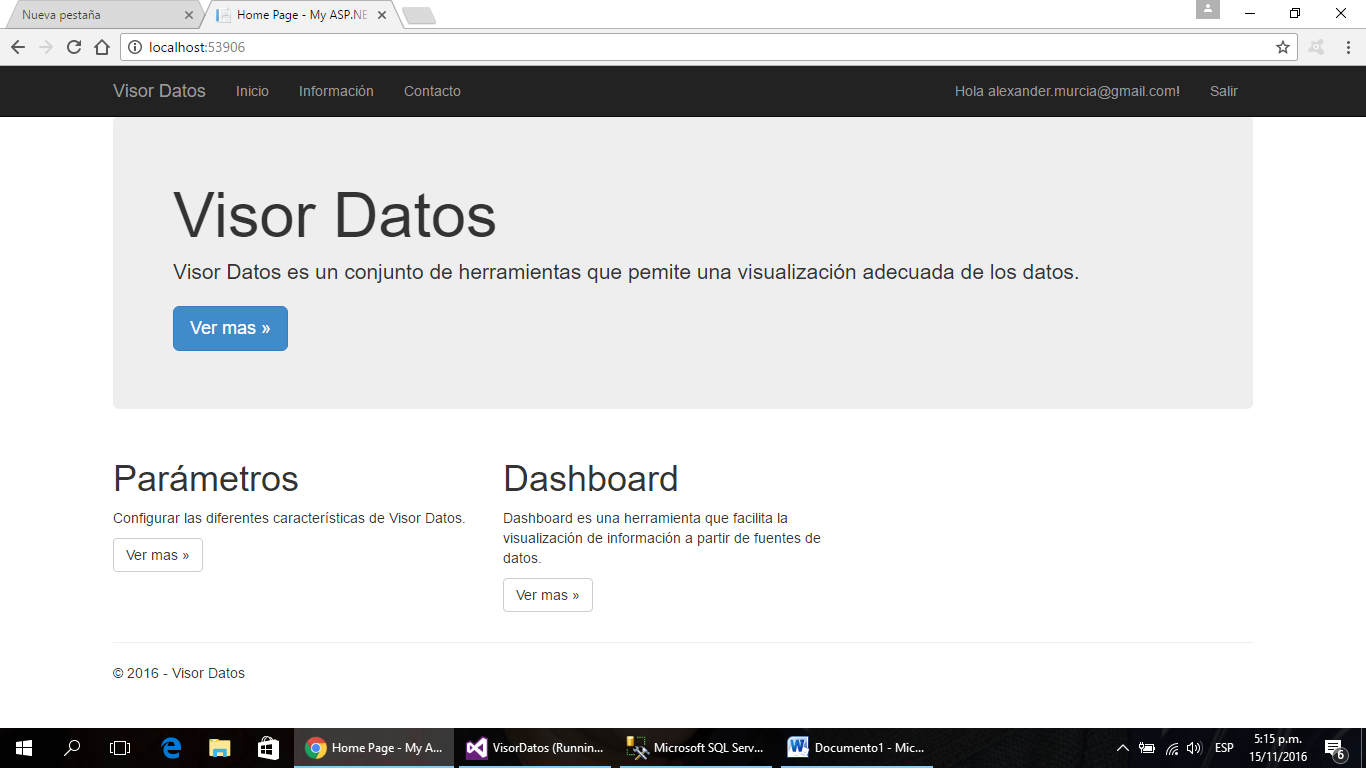
Registro: Captura de datos básicos del usuario para controlar el acceso a la aplicación Web

Login: Permite iniciar sesión dentro de la aplicación Web

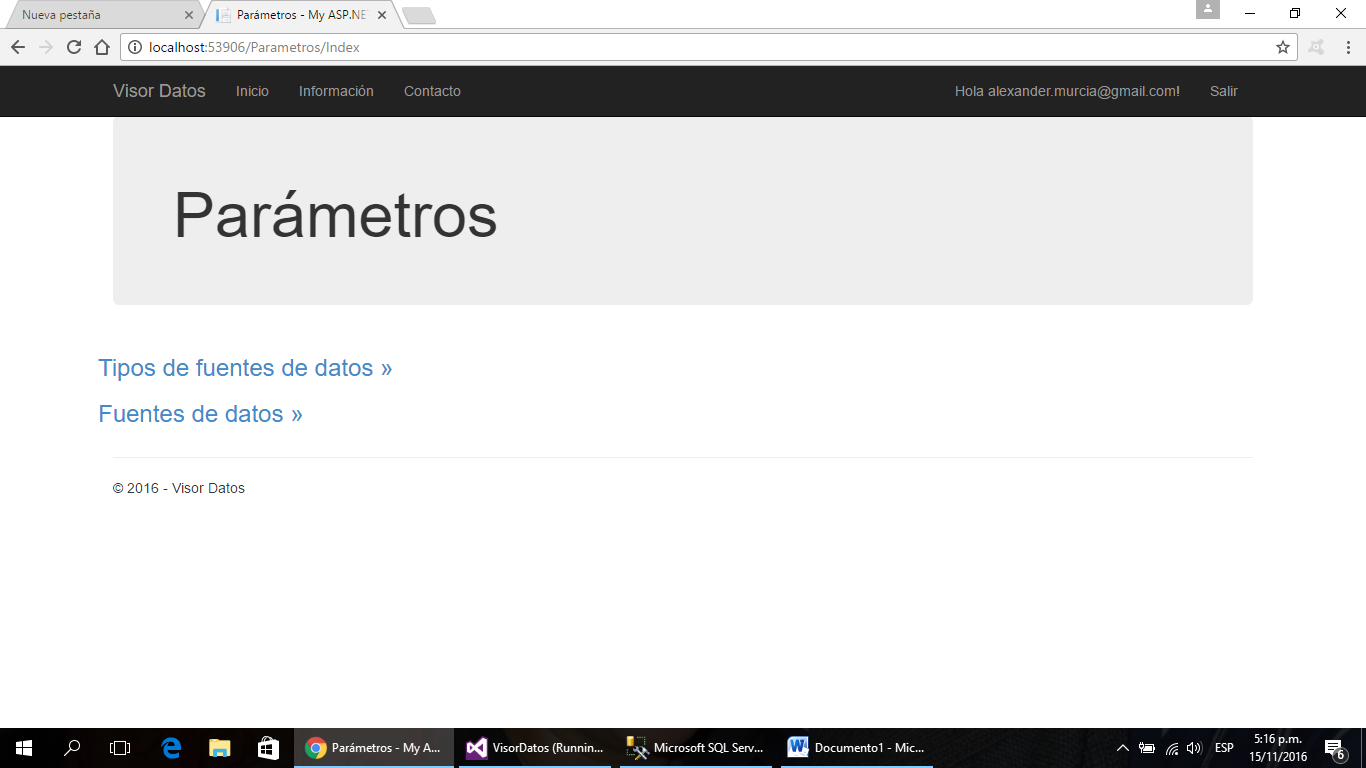
Index: Vista inicial que muestra la opciones brindadas por la aplicación Web



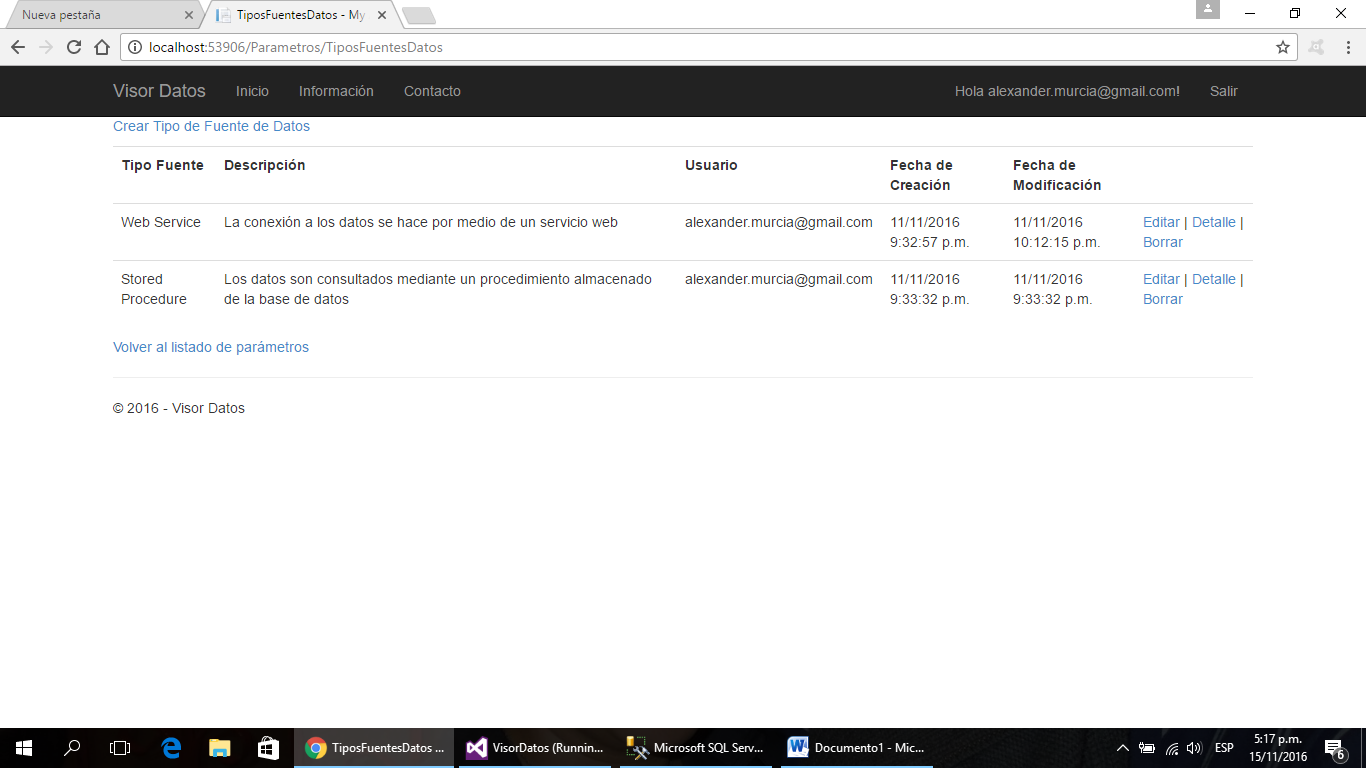


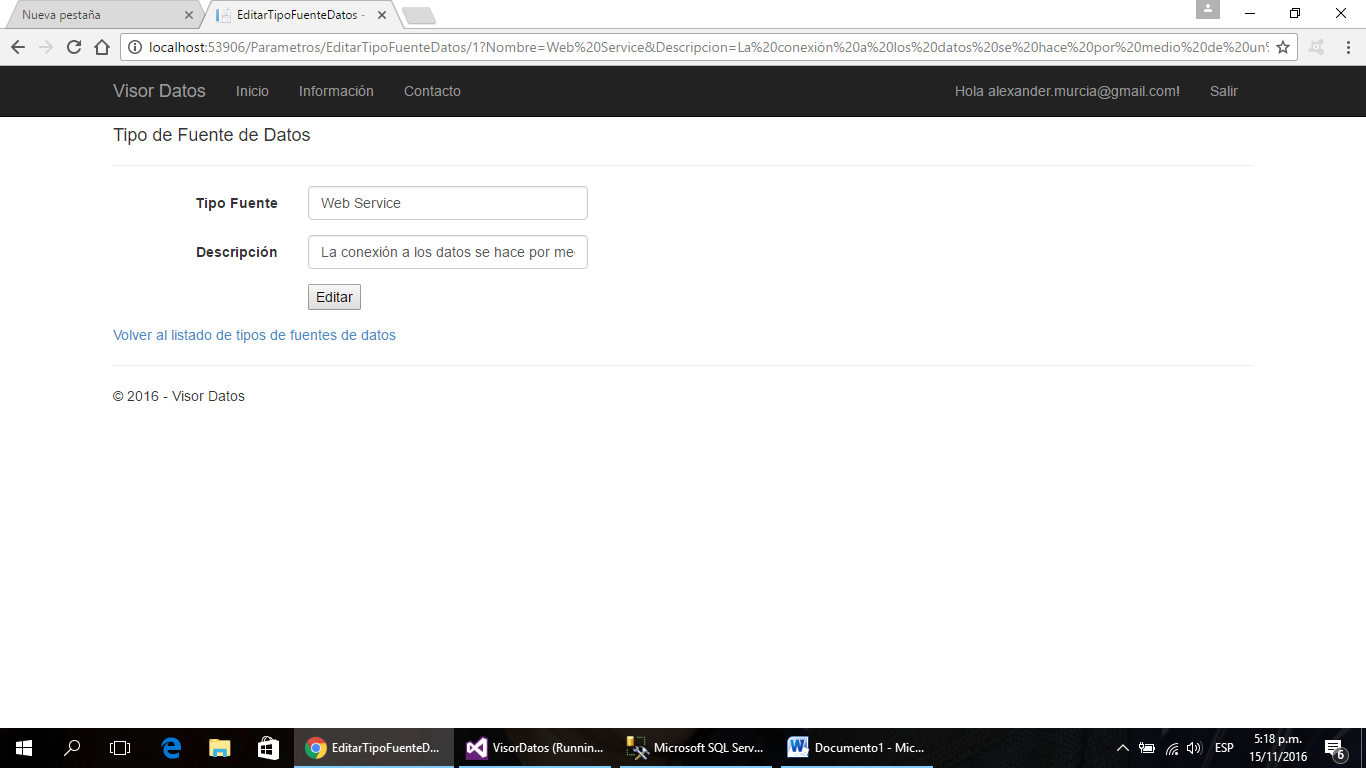


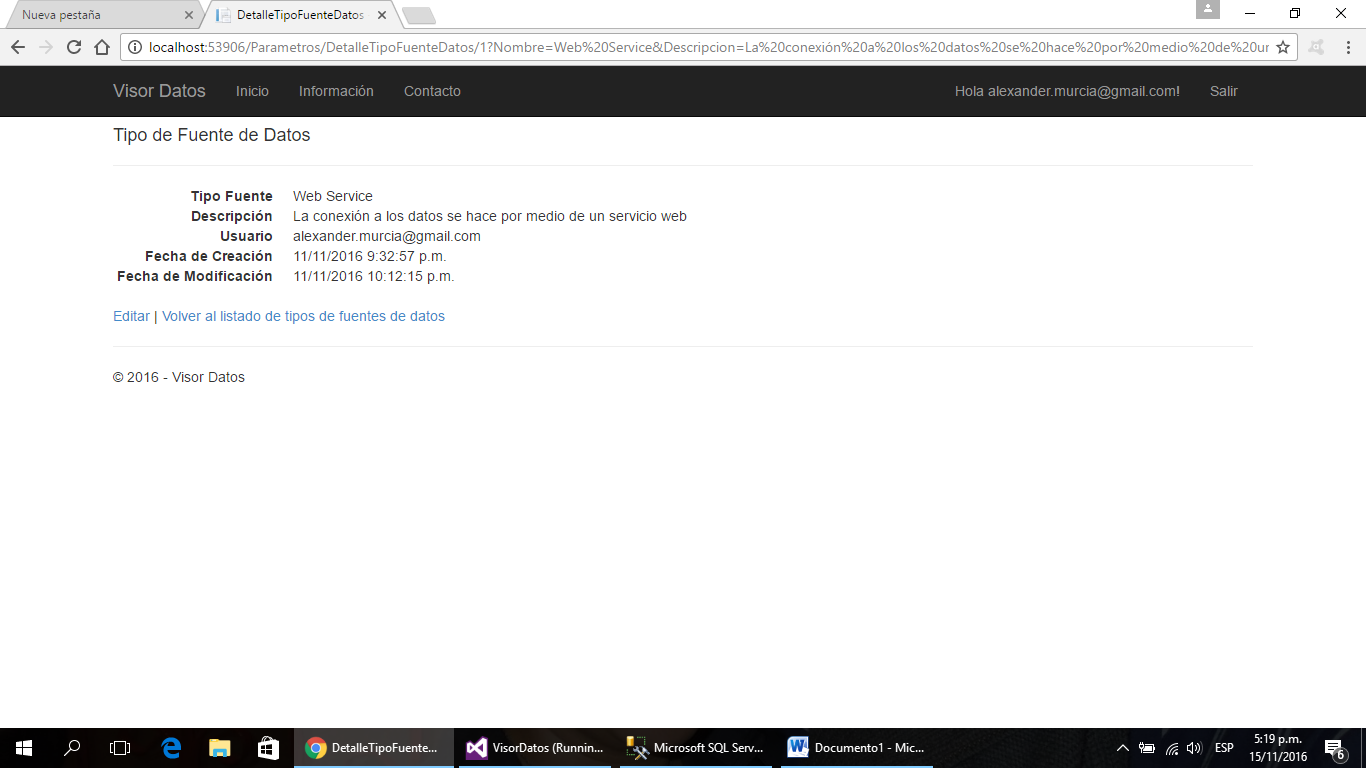
En la Administración de la información del prototipo tenemos principalmente la gestión sobre las fuentes de datos primarias, la cual se agrupo en la sección *Parámetros*

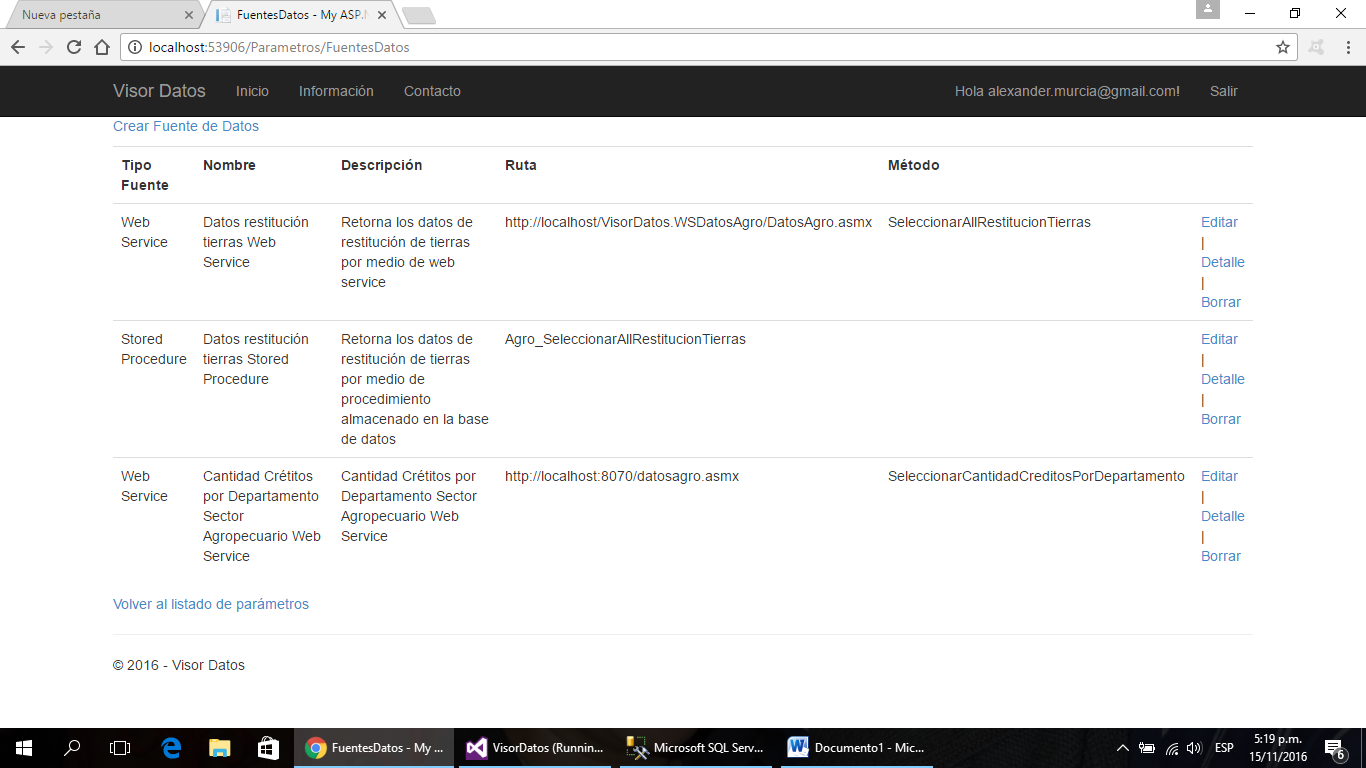


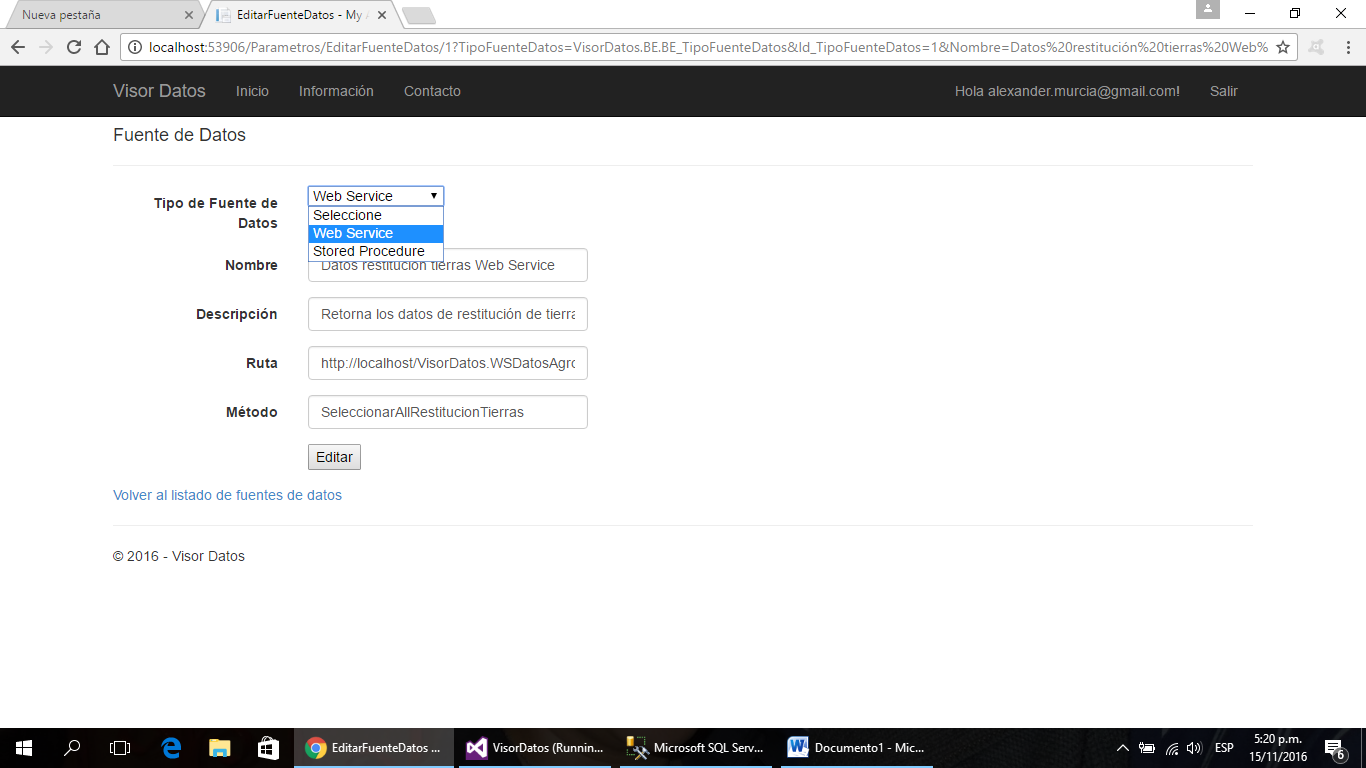
Administración Fuentes de Datos. Permite capturar, validar, visualizar, modificar y eliminar los registros creados por el usuario

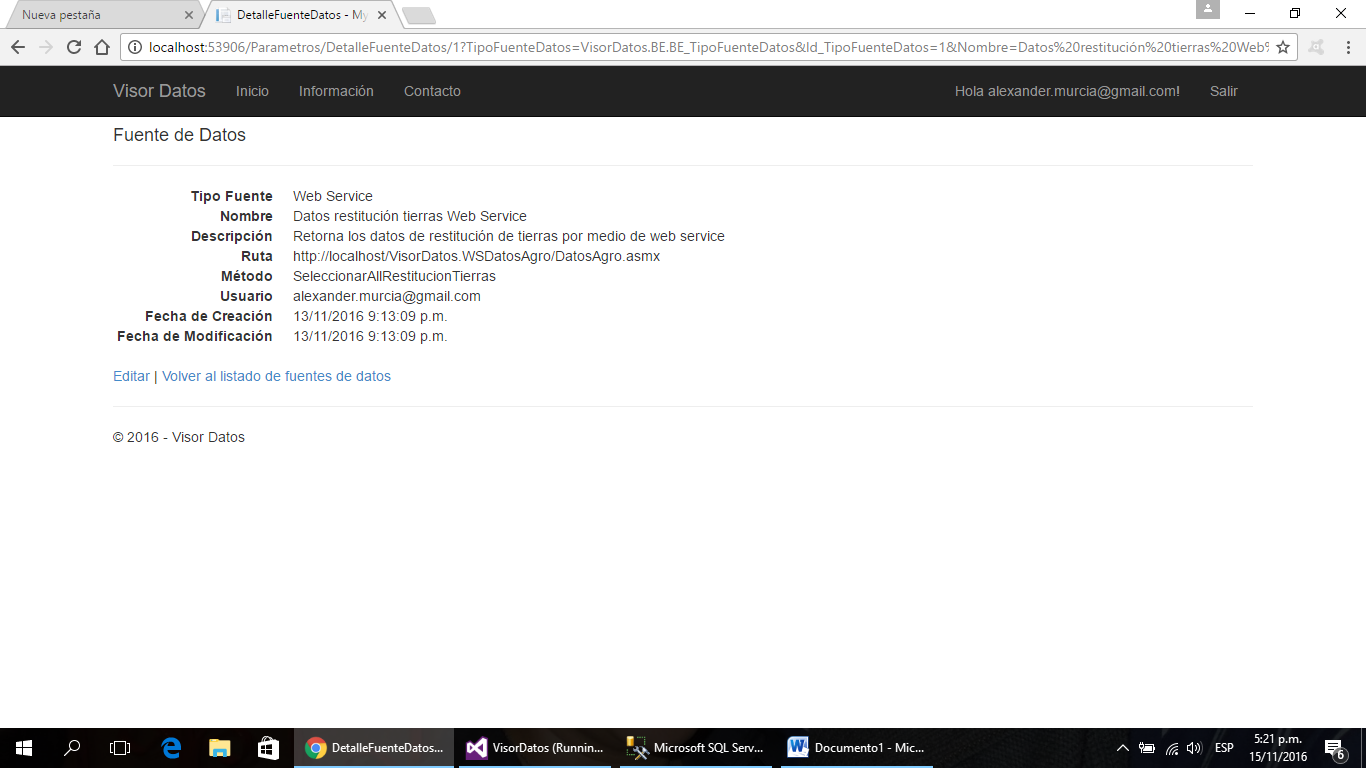




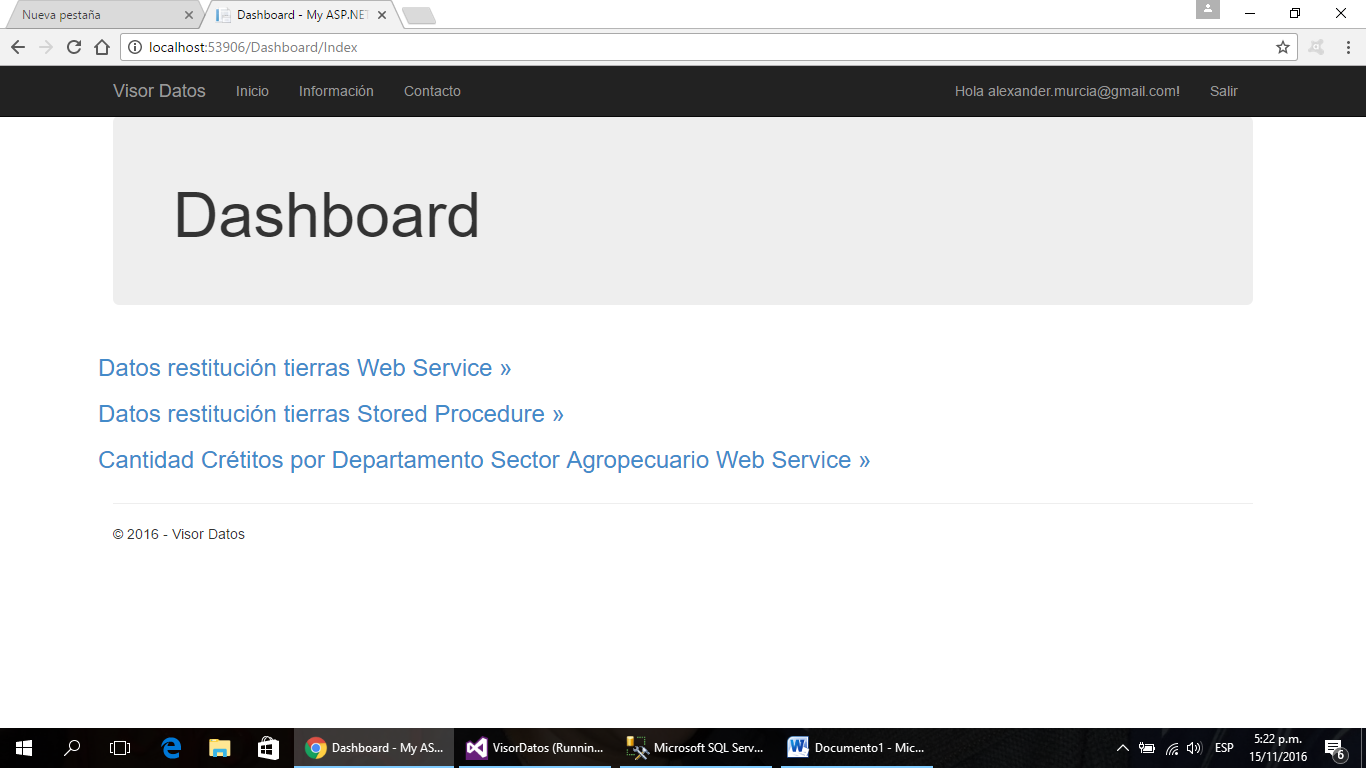








Dashboard. En esta vista aparecen las diferentes opciones de servicios generados por el usuario con base a la parametrización previa



En este caso, se visualiza en el dashboard la información relacionada con créditos para el sector agropecuario por departamento en Colombia con la fuente de datos consumida y parametrizada en la aplicación Web con el recurso datos.gov.co.



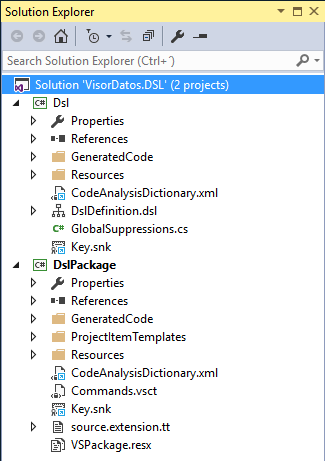
Introducción Domain Specific-Language (DSL)

La definición del modelo contiene elementos de un área de negocio, relaciones entre estos elementos, actores que interactúan para transformar el comportamiento de los objetos contenidos en el modelo, y por consiguiente, sus resultados.

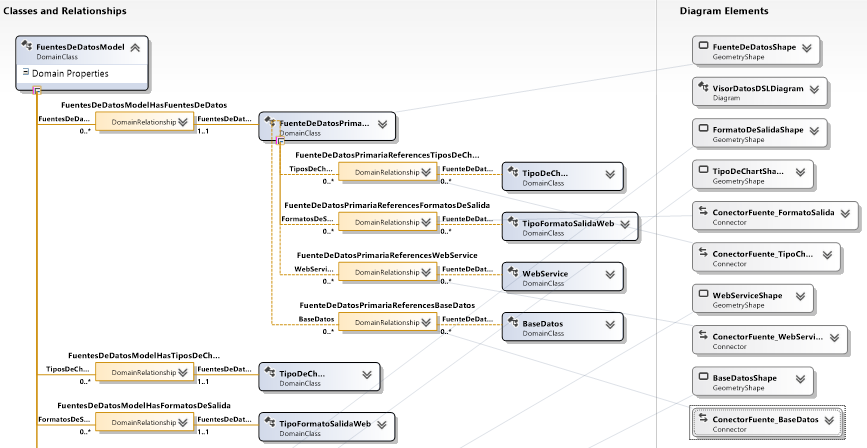
Integrando Modeling SDK for Visual Studio (MSDK) a Visual Studio 2015 podemos desarrollar modelos en forma de un DSL.

Para este modelo se establece que los cambios y/o modificaciones a las fuentes de datos primarias y las fuentes de salida establecidas se pueden automatizar y controlar a través de las características y capacidades incorporadas al modelo DSL apoyándonos en la herramienta de modelamiento sobre Visual Studio 2015. El backend de la arquitectura propuesta será administrado y controlado de forma manual a través de la configuración proporcionada por la plataforma Cloud Azure y los elementos relacionados con ésta se podrán automatizar a medida que los servicios de dicha plataforma así lo permitan.

De igual manera, se utiliza Visual Studio 2015 y la implementación de la estructura de la solución para el modelo se muestra a continuación



En tiempo de diseño, el modelo DSL se puede visualizar en la siguiente figura



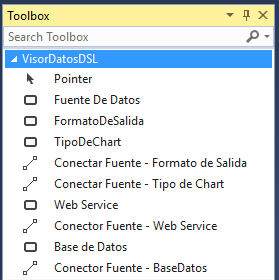
El Meta-Modelo es generado a través *Domain-Specific Language Designer* lo que permite definir un esquema para la sintaxis abstracta.

En el *Designer* observamos la construcción del modelo y clases de dominio y relaciones entre ellas. El modelamiento muestra las clases que representan las fuentes de datos relacionadas con los productores de datos, éstas a su vez, se especializan en *WebService* y *BaseDatos*. En el caso *WebService* permitiría que un sistema externo pueda consumir los datos expuestos en un tipo de formato como XML, JSON, entre otros, que se representa con la clase *TipoFormatoSalidaWeb*. En el caso *BaseDatos* permitiría extraer datos desde la conexión a una base de datos configurada en la arquitectura base.

La visualización de la información en el prototipo también es modelada por la clase *TipoDeChart*, los tipos incluidos son Pie y Barras.

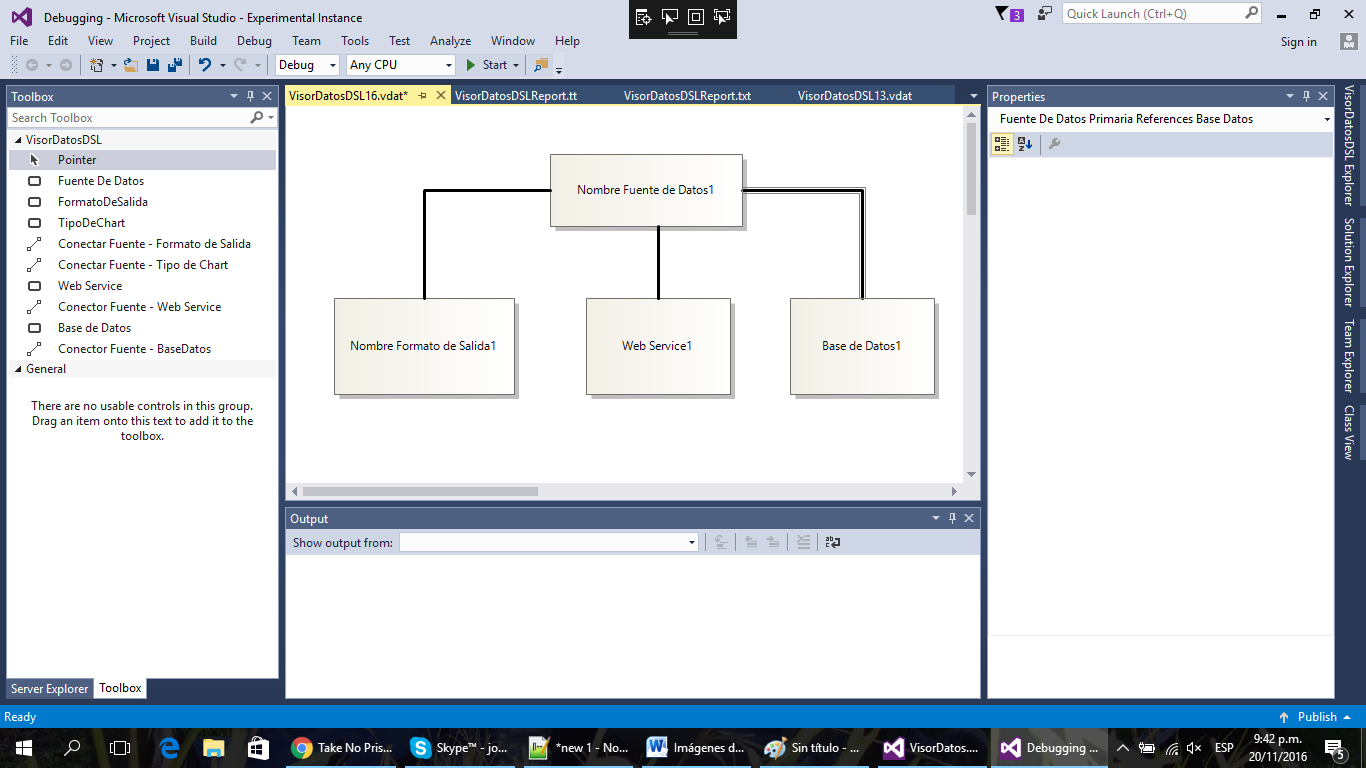
Ejecución del modelo

En el panel Toolbox obtenemos los elementos reutilizables en el modelo



Por último, la ejecución del modelo muestra el resultado en la siguiente imagen

El resultado de la ejecución del modelo nos proporciona la sintaxis concreta y sus elementos asociados.



El modelo ajustado al contexto real permite finalmente crear herramientas de desarrollo basadas en el modelo que podrían contener vistas, generación de código, artefactos, comandos y la capacidad de interactuar con otros componentes u objetos que conformen la solución dentro de Visual Studio 2015

Benjamin Torres Alvarez  
btorresal@gmail.com  
3212148513