FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

CONSTRUCCIÓN DE UN MÓDULO PARA LA PRESENTACIÓN DE INDICADORES ESTRATÉGICOS, COMO COMPLEMENTO AL CORE FINANCIERO DE DENARIUS

Trabajo de Titulación en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Sistemas de Computación e Informática

Profesor Guía
Msc. Jonathan Patricio Carrillo Sánchez

Autor
Franklin Miguel Tapia Guamán

Año
2016

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido este trabajo a través de reuniones periódicas con el/la estudiante, orientando sus conocimientos para un adecuado desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajaos de Titulación”

_____

Msc. Jonathan Patricio Carrillo Sánchez
Magíster en Gestión de las Comunicaciones y Tecnologías de la Información
CI.:1712263084

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”

_____

Franklin Miguel Tapia Guamán

CI.:1720065927

AGRADECIMIENTOS

Agradezco a todas las personas que me apoyaron en la elaboración de este Trabajo de Titulación ya sea con su tiempo o palabras de aliento, y en especial a mi familia, mis padres María y Miguel que siempre encontré una mano paciente y calurosa.

RESUMEN

En la actualidad el desarrollo de sistemas informáticos que faciliten la toma de decisiones ha tenido una tendencia creciente a ser implementados. En las organizaciones es de vital importancia tener ventajas competitivas en el entorno que se encuentran, y para las entidades financieras, caso de estudio de este proyecto, no puede ser de otra manera.

Por este motivo Denariuservis SA, empresa desarrolladora de software y que brinda servicios financieros y facturación electrónica, requiere de una herramienta de gestión que permita ver el estado general de las cuentas de los clientes en las diferentes entidades financieras que han contratado su servicio.

El presente proyecto se desarrolló con el fin de agregar nuevos servicios al sistema denarius Core para complementar los ya existentes y proporcionar valor agregado al producto. Para desarrollar el proyecto de BI se utilizó como metodología de Business Intelligence el ciclo de vida de Kimball y para construir el módulo para la presentación de indicadores estratégicos se utilizó Scrum.

Una vez terminado este proyecto y validado por los responsables del área de desarrollo de Denariuservis, se prevé que el producto permitirá generar reportes consolidados, minimizar los costos operacionales al centralizar la información y mejorar los tiempos de respuesta para el análisis de la información en la toma de decisiones.

ABSTRACT

In this day and age, the development of computer systems to facilitate decision-making has been a growing trend to be implemented. In organizations it is vital to have competitive advantages in the environment they are in, and for financial institutions, case study of this project cannot be otherwise.

For this reason, Denariuservis SA, and software development company that provides financial services and electronic invoicing requires a management tool that allows to see the overall status of customer accounts at various financial institutions that have hired their service.

This project was developed in order to add new services to denarius Core system to complement existing ones and provide added value to the product. To develop the project BI methodology was used as Business Intelligence lifecycle of Kimball and build the module for the presentation of strategic indicators was used Scrum.

Once this project and validated by the development area Denariuservis, it is expected that the product will generate consolidated reports, minimize operational costs by centralizing information and improve response times for data analysis in decision making.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Capítulo I. Estrategia Metodológica

En la presente sección se abordan las generalidades con respecto al proyecto:

1.1 Antecedentes del Proyecto

Denariuservis SA es una empresa dedicada al desarrollo de core financiero. Se encuentra ubicado en la calle Whymper N27-88 y Avenida Francisco de Orellana. Su enfoque principal es brindar servicios bancarios a las entidades financieras, contando con los más altos estándares tecnológicos.

El principal producto de la empresa es denarius Core, sus aplicaciones incluyen: gestión de clientes, cuentas, crédito, inversiones, switch transaccional flujos de trabajo, archivo digital, cumplimiento y servicios generales.

Actualmente el producto no presta el servicio de Business Intelligence (BI), por lo que es necesario incluir en su core financiero un módulo que presente los indicadores estratégicos, como valor agregado a su solución.

1.2 Alcance

El proyecto contará con indicadores estratégicos que serán presentados a través de reportes mediante la agregación de un módulo al sistema Denarius Core.

Se utilizarán herramientas nativas de Microsoft: Visual Studio, SQL Server, Runbook Designer, Release Management, y Team Foundation Server (TFS). Para la creación de los procesos ETL y la presentación de los reportes respectivos, junto con la administración de la ejecución de los procesos para el paso de información.

Las herramientas a utilizar se describen en la sección: 2.3.2 Diseño Arquitectura Técnica. Los indicadores estratégicos a desarrollar en el presente proyecto se encuentran descritos en la sección: 2.2.1 Indicadores a implementar.

1.3 Justificación

Actualmente la empresa Denariuservis SA no cuenta con una herramienta de gestión que permita analizar el estado en el que se encuentran las entidades financieras, por tal motivo se construirá el módulo de indicadores de gestión para integrar en el core principal.

Se prevé que mediante la implementación del módulo de BI en denarius Core, el producto permitirá generar reportes consolidados, minimizar los costos operacionales al centralizar la información y mejorar los tiempos de respuesta para el análisis de la información y en la toma de decisiones.

1.4 Objetivos

A continuación, se describe el objetivo general del proyecto junto con los objetivos específicos.

1.4.1 Objetivo General

Construir un módulo para la presentación de indicadores estratégicos, como complemento al Core Financiero de la empresa Denariuservis SA.

1.4.2 Objetivo Específico

- Realizar el diseño de la solución del visualizador gerencial en función de los indicadores estratégicos.
- Crear los reportes o gráfico de los indicadores entregados para ser visualizados en pantalla.
- Implementar el módulo de indicadores con la herramienta seleccionada al core en el ambiente de pruebas.

1.5 Metodología

Se selecciona dos metodologías para el desarrollo del proyecto: el ciclo de vida de Kimball (solución BI) y scrum (solución aplicativo BI).

1.5.1 Ciclo de Vida de Kimball

La metodología de Kimball, fue creada a mediados de la década de 1980 por los miembros del Grupo de Kimball y otros colegas en Metaphor Computer Systems, una compañía queda apoyo a la toma de decisiones. Desde entonces, se ha utilizado con éxito por los equipos de proyectos dentro de las organizaciones.

Como lo menciona Ross en el 2009, la metodología seleccionada será de Kimball denominada Ciclo de Vida Dimensional, basa en cuatro principios básicos:

- Centrarse en el negocio
- Construir una infraestructura de información adecuada
- Realizar entregas en incrementales significativos
- Ofrecer la solución completa

El enfoque del ciclo de vida Kimball se ilustra en la figura 1 y se puede utilizar para proporciona una hoja de ruta general que representa la secuencia de tareas de alto nivel necesarias para los proyectos de BI / DW.

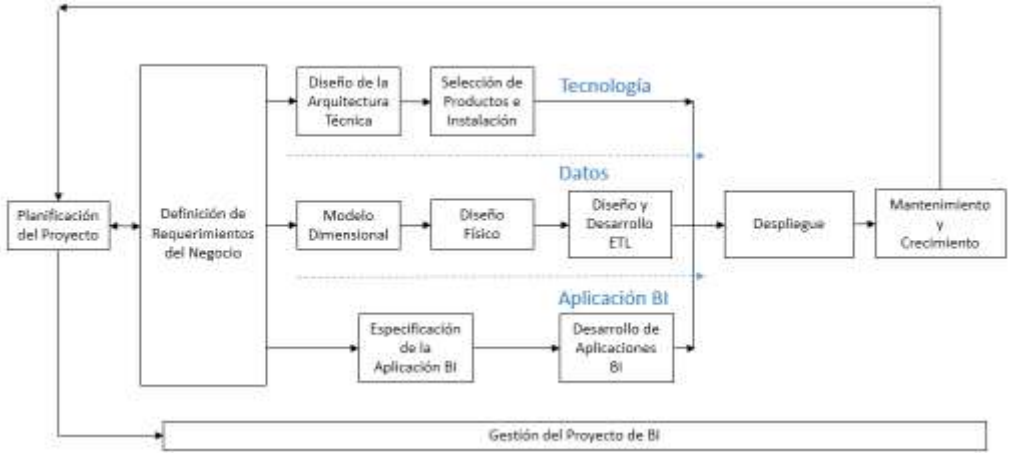


Figura 1: Diagrama del ciclo de vida de Kimball
Adaptado de Kimball y Ross, 2002, p. 332

En la figura 1 se observa la ruta del ciclo de vida de Kimball, se inicia con la planificación del proyecto. La segunda tarea relevante se centra en la definición de requisitos, en la figura notamos la flecha bidireccional entre la planificación y requisitos, es debido a la gran interacción que existe entre ambas actividades.

Se deberá considerar la existencia de tres líneas claramente definidas, como se puede observar en la figura 1: tecnología, datos y aplicación BI. A continuación, se listan las tareas que se realizarán en el proyecto por fases:

En la línea de tecnológica o camino superior con la actividad del diseño de la arquitectura técnica, permite establecer un marco general para la integración de múltiples tecnologías.

La línea de datos o camino medio contiene la creación del modelo dimensional, para luego transformarse a una estructura física que incluyen estrategias de optimización. Por último, el desarrollo del proceso ETL (extracción, transformación y carga).

Línea de aplicación BI o camino inferior, nace a partir de los requerimientos del negocio tomados de las necesidades de los usuarios. Al terminar con las tres líneas se procede a su integración en la actividad de despliegue.

Es necesario realizar un mantenimiento continuo que garantice la solución BI. Por último, se ocupa del crecimiento que da inicio a un nuevo ciclo de vida.

1.5.2 Scrum

La metodología Scrum para el desarrollo del aplicativo BI, fue definida en 1993 y alrededor de 1986 se implementa principalmente en empresas donde manejan productos y su salida al mercado se debe dar en el menor tiempo posible.

> “Jeff Sutherland creó el primer equipo de Scrum en 1993 y trabajó con Ken Schwaber para formalizar Scrum en OOPSLA’95 (Object-Oriented

> Programming, Systems, Languages & Applications). Juntos, expandieron y mejoraron Scrum en muchas compañías de software, ayudaron a escribir el Manifiesto Ágil en 2001, y fueron los autores la guía de Scrum" (Sutherland, s.f.)

Scrum en un marco de trabajo que se enfoca en el desarrollo de proyecto. Su enfoque es la entrega de software que funcione luego de la finalización de una iteración, fomenta el trabajo en equipo de forma auto-organizada. Es muy flexible en cuanto a los cambios que se presente en el sistema a lo largo de su trascurso.

En la figura 2 se presenta el funcionamiento de scrum:

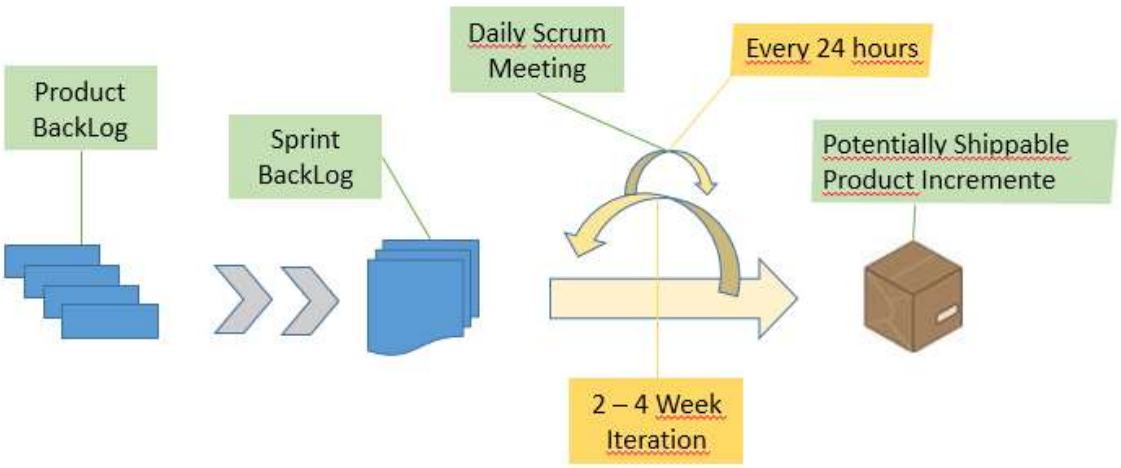


Figura 2: Resumen de funcionamiento de Scrum
Adaptado de Mountain Goat Software, s.f.

El proceso se centra en entregables cortos y fijos con el empleo de **iteraciones** (figura 2) que suelen demorar de 2 semanas hasta un máximo de 4 semanas. Se deberán realizar las tareas establecidas por el equipo. Las actividades o reuniones tienen como fin establecer el camino a seguir durante las iteraciones y reuniones diarias.

A continuación, se listan las actividades o reuniones que se realizan mediante la utilización de scrum:

- Planificación de la liberación (Release Planning)
- Planificación de la iteración (Sprint Planning Meeting)
- Reunión diaria del equipo (Scrum Daily Meeting)
- Scrum de Scrum
- Reunión de revisión de la iteración (Sprint Review Meeting)
- Retrospectiva de la iteración (Sprint Retrospective)

Scrum maneja roles dentro de su metodología, para la gestión del proyecto y sus tareas: Product Owner, es la o las personas que representa a los interesados. Scrum Master, facilitador que ayuda a eliminar obstáculos del equipo de trabajo. Scrum Team, son los miembros del equipo de desarrollo. Stakeholders son los interesados y principal beneficiario del proyecto. Administradores son encargados del confort del ambiente para el desarrollo del sistema. Los roles se asignan luego de la conformación del equipo.

Las herramientas que utiliza scrum para su metodología son: historias de usuario, corresponden a la actividad que realiza el usuario. Product BackLog, expectativas que tienen los clientes del proyecto y son priorizados por sus propias necesidades. Sprint Backlog, tareas que se realizarán en la iteración. Burn Down Chart, gráfico que permite visualizar el trabajo pendiente en el proyecto y el avance que se logra al culminar la iteración. Scrum Daily Meeting (reunión diaria del equipo), son reuniones de máximo 30 minutos, se menciona la actividad realizada el día de ayer, en que está en este momento y que va hacer más adelante. Por último, expresa si tiene ulgun obstáculo para continuar con su actividad.

2. Capítulo II. Desarrollo del Proyecto de Inteligencia de Negocio

En la presente sección se toma como base la metodología del ciclo de vida Kimball de la figura 1, y como referencia la tabla 1 de este documento para el desarrollo del proyecto de BI.

Tabla 1: Actividades separadas por fases del ciclo de vida de Kimball

	Fase del Proyecto	**Ciclo de vida de Inteligencia de Negocios de Kimball**
a	Planificación	• Planificación del proyecto
b	Requerimientos	• Definición de requerimientos del negocio
c	Línea Tecnológica	• Diseño de la arquitectura técnica • Selección de productos e instalación
d	Línea de Datos	• Modelo dimensional • Diseño físico • Diseño e implementación del ETL
e	Línea de Aplicación BI	• Diseño de la aplicación de BI • Desarrollo de la aplicación de BI
F	Implementación	• Despliegue • Crecimiento • Mantenimiento
g	Gestión	• Gestión del proyecto. Según la metodología es una actividad transversal de todas las fases anteriores.

a. Fase 1 (**planificación**), dentro de las actividades de esta fase están presente: definición del alcance, identificación de tareas, programación de tareas, recursos, asignación de la carga de trabajo a los recursos, adquisiciones.

b. Fase 2 (**requerimientos**), la especificación de requerimientos comprende una de las actividades más importantes en esta fase del ciclo, porque desde este punto dan inicio a las fases subsiguientes. Se establecen reuniones

entre las personas interesadas, se incluye informes y procesos internos que se utilizan dentro de la organización.

c. Fase 3 (**línea tecnológica**), presenta el diseño arquitectónico de la solución BI conjuntamente con el empleo de las herramientas específicas a utilizar en el proyecto.

d. Fase 4 (**línea de datos**), actividades relacionadas con los daros. Se diseña y desarrolla el modelo dimensional de la misma manera para el proceso de extracción, transformación y carga, también se incluye el modelo físico.

e. Fase 5 (**línea de aplicación BI**), se relaciona a las actividades del aplicativo BI. Se diseña la arquitectura y se desarrolla las interfaces para los usuarios finales.

f. Fase 6 (**implementación**), en las actividades de despliegue comprende las tareas relacionadas a la puesta en escena en el ambiente de pruebas. Por último, se trata a recomendaciones generales con respecto al crecimiento y mantenimiento.

g. Fase 7 (**gestión**), esta fase se presenta en todas las actividades del proyecto, debido a que monitora el estado del proyecto.

2.1 Planificación

En esta fase encontramos la tarea de Planificación de Proyecto a continuación, se lista las tareas de esta fase:

a. Definir el alcance.
b. Justificación
c. Identificar las tareas
d. Programación de Tareas
e. Asignar la carga de trabajo a los recursos (talento humano).

f. Adquisiciones (hardware, software, servicios)

Dentro de la fase de planificación de la metodología de Kimball, existe una recomendación con respecto a los factores a tener en cuenta dentro de la organización: patrocinador o patrocinadores, motivación empresarial, viabilidad del proyecto, relación entre la organización empresarial y de TI y el último es la cultura analítica dentro de la empresa (intuición, evidencia anecdótica, emocional)

Para los puntos a (alcance) y b (justificación), ya fueron realizados en el anteproyecto y en al actual proyecto en el capítulo 1. Para los puntos c (identificar las tareas) y d (programación de tareas) se emplea el diagrama de Gantt que muestra las tareas, el tiempo que tomará el desarrollo de las y las fechas en las que serán desarrolladas, ver figura 3:

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predec
1		**Indicadores Denarius**	**66 días**	**jue 14/04/16**	**jue 14/07/16**	
2		**Planificación y Requerimientos**	**11 días**	**jue 14/04/16**	**jue 28/04/16**	
3		Revisión anteproyecto	1 día	jue 14/04/16	jue 14/04/16	
4		Selección metodología, cronograma	5 días	vie 15/04/16	jue 21/04/16	3
5		Definición de requerimientos	5 días	vie 22/04/16	jue 28/04/16	4
6		**Línea Tecnológica**	**5 días**	**vie 29/04/16**	**jue 05/05/16**	
7		Diseño arquitectura tecnológica	4 días	vie 29/04/16	mié 04/05/16	5
8		Selección de productos e instalación	1 día	jue 05/05/16	jue 05/05/16	7
9		**Línea de Datos**	**20 días**	**vie 06/05/16**	**jue 02/06/16**	
10		Modelo multidimensional	5 días	vie 06/05/16	jue 12/05/16	8
11		Diseño físico	3 días	vie 13/05/16	mar 17/05/16	10
12		Diseño e implementación del ETL	7 días	mié 18/05/16	jue 26/05/16	11
13		Corrección modelo dimensional y E	5 días	vie 27/05/16	jue 02/06/16	12
14		**Línea de Aplicación BI**	**10 días**	**vie 03/06/16**	**jue 16/06/16**	
15		Diseño de la aplicación	3 días	vie 03/06/16	mar 07/06/16	13
16		Desarrollo de la Aplicación	7 días	mié 08/06/16	jue 16/06/16	15
17		**Implementación**	**10 días**	**vie 17/06/16**	**jue 30/06/16**	
18		Despliegue ETL	3 días	vie 17/06/16	mar 21/06/16	16
19		Despliegue Aplicativo	6 días	mié 22/06/16	mié 29/06/16	18
20		Mantenimiento y Control	1 día	jue 30/06/16	jue 30/06/16	19
21		Revisión y corrección final documento	5 días	vie 01/07/16	jue 07/07/16	20
22		Revisión y corrección final sistema	5 días	vie 08/07/16	jue 14/07/16	21

Figura 3: Diagrama de Gant – Proyecto Indicadores Denarius

e. Asignar la carga de trabajo a los recursos (talento humano)

Se deberá tener en cuenta en primera instancia todo el personal ha signado para el desarrollo del presente proyecto, la conformación del equipo dependerá en gran medida del alcance del proyecto, a continuación, se en lista las principales funciones desde el lado de la organización y el de TI:

Roles de la organización:

- Promotor del negocio
- Director del negocio
- Usuarios del negocio

Roles del equipo de TI:

- Analista de sistema del negocio
- Experto del negocio
- Desarrollador de aplicaciones analíticas
- Educador del almacén de datos

Roles consultores externo (pueden ser asumidos por el equipo de TI)

- Gerente del proyecto
- Arquitecto
- Especialista soporte técnico
- Modelador de datos
- Administrador de base de datos
- Coordinador de metadatos
- Administrador de datos
- Diseñador de estadística de datos
- Soporte para el almacén de datos

De todos los posibles roles a utilizar en el presente, en su mayoría son asumidos por el tesista, en cuanto a los roles de la organización se identifica como

promotor de negocio a Denarius SA. De igual manera ocurre con la asignación de recursos para cada cargo dentro del proyecto

f. Inversión

En este punto se deberá mencionar sobre los costos de inversión de hardware, software y servicios a utilizar en el desarrollo del proyecto. En primera instancia no se requiere la adquisición de nuevo equipos o herramientas, esto debido a que se plantea la implementación a un producto ya existente y ya se cuenta con los equipos y herramientas necesarias. A continuación se presentan los costos de servicios en la tabla 2:

Tabla 2: Costo de Servicios

Descripción	**Costo Mensual**	**Meses Totales**	**Subtotal**
Llamadas Telefónicas	10	4	40
Internet	35	4	140
Energía Eléctrica	10	4	40
Fotocopias-Empastados	15	4	60
Total			**280**

Se deberá tener en cuenta que si en el transcurso del proyecto se tiene la necesidad de adquirir equipo o herramientas se deberá gestionar la compra de los mismos.

2.2 Requerimientos

En esta fase encontramos la tarea de Definición de requerimientos del negocio, a continuación, se describe los requerimientos del proyecto.

2.2.1 Descripción Fuente de datos

La fuente de datos será una base de datos transaccional a continuación, se describe de manera general sus características.

Si bien es cierto que los registros en la base de datos vienen desde un sistema transaccional (Denarius Core), las tablas implicadas tienen una transaccional baja, el promedio de registros es más de 3000 mensuales en las tablas principales, en la tabla 4 se muestra el número total de registros ingresados desde el mes de enero hasta julio de 2016.

Tabla 3: Número de registros en las tablas e origen

Tabla	Total registros	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio
Cuenta	668500	2707	2784	3293	3244	3311	3713	3291
Cliente	373620	4160	4326	4780	4596	6456	6257	4940

Nota: En la columna N° total registros muestra el número total de filas en la tabla.

2.2.2.2 Tablas origen

A continuación, se presentan los campos principales de las tablas:

Tabla 4: Tabla Cuenta

Tabla	Campo	Tipo
Cuenta	IdCuenta	int
Cuenta	IdProducto	tinyint
Cuenta	IdSubproducto	smallint
Cuenta	IdOficialAdministrador	int
Cuenta	IdCliente	int
Cuenta	IdSucursal	smallint
Cuenta	IdOficinaOficial	smallint
Cuenta	Historico	bit
Cuenta	Nombre	varchar

Tabla 5: Tabla Cliente

Tabla	Campo	Tipo
Cliente	IdCliente	int
Cliente	TipoPersona	tinyint
Cliente	FechaRegistro	smalldatetime
Cliente	FechaModificacion	smalldatetime
Cliente	TipoIdentificacion	decimal
Cliente	Identificacion	varchar
Cliente	IdentificacionExtranjera	varchar
Cliente	NombreCompleto	varchar
Cliente	Nacionalidad	int
Cliente	IngresoAnual	money
Cliente	Patrimonio	money
Cliente	Oficial	int
Cliente	Estado	int

2.2.2 Definición de Requerimientos del negocio

A continuación, se plantean dos indicadores a implementar en base a la información entregada por el cliente. Además, se considera únicamente tomar la información a partir del 2016, esto debido a la consolidación de diferentes sistemas operacionales al nuevo core.

Primer indicador, proceso de la gestión de apertura de cuentas e ingreso de clientes a partir del 2016, de cada una de las sucursales de la entidad financiera.

Tabla 6: Gestión apertura de cuentas e ingreso de clientes

Identificador	IndAper01	**Nombre**	Apertura de cuentas e ingreso de clientes.
Fecha de definición	20-05-16	**Módulos**	Cuentas , clientes
Descripción	Permite generar reportes del número de cuentas aperturadas e ingreso de clientes a partir del 2016, con el empleo de diferentes tipo de filtros: estado, sucursal, ciudad, fecha, edad, sexo y en el caso de cuentas el tipo de subproducto estado (ver anexo 1)		
Datos Origen	Base de datos operacional (Tenant)		

A continuación, se describen los reportes a implementar:

- Número total de clientes **ingresados** mensualmente por sucursal.
- Número total de clientes ingresados que se encuentren en estado **activo** mensualmente por sucursal.
- Número total de clientes ingresados que se encuentren en estado **pasivo** mensualmente por sucursal.
- Número total de cuentas **aperturadas** por subproducto mensualmente y por sucursal.
- Número total de cuentas **activas** por subproducto mensualmente y por sucursal.
- Número total de cuentas liquidadas por subproducto mensualmente.

Segundo indicador es el saldo de apertura de una cuenta de ahorros a partir del 2016, de cada una de las sucursales de la entidad financiera.

Tabla 7: Saldo de apertura de una cuenta de ahorros

Identificador	IndSald02	**Nombre**	Saldo apertura de las cuentas de ahorros.
Fecha de definición	20-05-16	**Módulo**	Cuentas, clientes
Descripción	Permite generar reportes del saldo de las cuentas aperturadas a partir del 2016, con el empleo de diferentes tipo de filtros: estado, sucursal, ciudad, fecha, edad, sexo y tipo de subproducto		
Datos Origen	Base de datos operacional (Tenant)		

A continuación, se describen los reportes a implementar:

- Saldo promedio de apertura del subproducto **cuenta de ahorros** de los clientes segmentado por sucursal desde el año 2016.
- Saldo promedio de los clientes de sexo femenino que se han mayores a 25 años y por sucursal en el mes de febrero.

2.3 Línea Tecnológica

En esta fase encontramos las siguientes tareas: diseño de la arquitectura técnica y la selección de productos e instalación. A continuación, vamos a tratar cada una de estas actividades.

2.3.1 Diseño de la arquitectura técnica

A continuación, en la figura 4 se representa la arquitectura de BI que se utiliza en el proyecto:

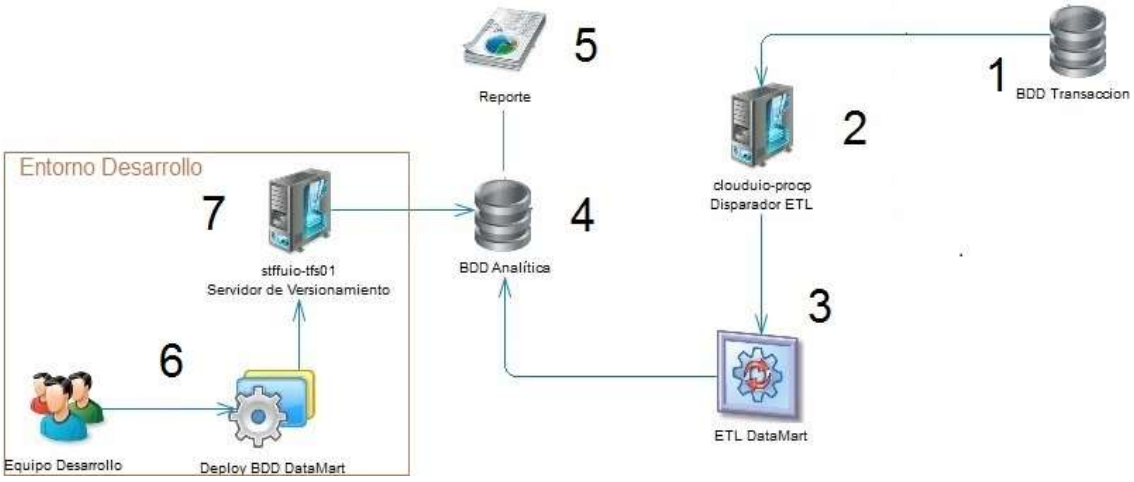


Figura 4: Solución de la arquitectura BI.

1 BDD Transaccional, representa el origen de la fuente de datos desde el sistema transaccional.

2 Disparador ETL, permite gestionar la ejecución del proceso ETL

3 ETL Datamart, la ejecución del ETL permitirá depositar la data tratada en una nueva base de datos.

4 BDD Analítica, luego de la finalización de la ejecución del ETL se deposita la data generada en una base de datos analítica para su posterior consumo.

5 Reporte, comprende la presentación del indicador por pantalla.

Entorno Desarrollo, se incluye el proceso de implementación desde el ambiente de desarrollo a los servidores de producción Deploy BDD DataMart (6). Se publican los cambios a la línea base del sistema para luego realizar la integración del código en el servidor de versionamiento (7).

A continuación, se presenta la tabla listando los componentes antes descritos de la arquitectura de la solución BI:

Tabla 8: Componentes de la solución BI.

#	**Componente**	**Descripción**	**Procesador/Memoria**
1	Fuente de Datos **clouduio-bddcc**	Base de datos transaccional: **BDD Transaccional** SQL Server 2012	4 procesadores (2.80 GHz) Ram: 20.00 Gb
2	Disparador ETL **clouduio-procp**	Runbook Designer: **Disparador ETL** System Center 2012 R2	2 procesadores (2.80 GHz) Ram: 6.00 Gb
3	ETL DataMart	Integration Services Project Visual Studio 2012	
4	Data Warehouse **clouduio-bddcc**	BDD Analítica SQL Server 2012	4 procesadores (2.80 GHz) Ram: 20.0 Gb
5	Reportes	Consultados desde el core	
6	Versionamiento **staffuio-tfs01**	Servidor para pasos de versión TFS 2012	
7	Deploy DataMart **localhost**	Proyecto BDD, desarrollo	4 procesadores (2.67 GHz) Ram: 4.00 Gb

2.3.2 Selección de productos e instalación

Al tratarse de una implementación a un sistema ya existente no es necesario realizar una instalación adicional para el desarrollo del aplicativo BI, a continuación, se describe las herramientas a utilizar para el presente proyecto y sus requerimientos:

Visual Studio

Visual Studio es un conjunto de herramientas y otras tecnologías de desarrollo de software basado en componentes para crear aplicaciones eficaces y de alto rendimiento.

Requerimientos

Sistemas Operativos Compatibles:

- Windows 7 SP1 (x86 and x64)

- Windows 8 (x86 and x64)
- Windows Server 2008 R2 SP1 (x64)
- Windows Server 2012 (x64)

Requisitos de Hardaware

- Procesador de 1,6 gigahercios (GHz) o más rápido.
- 1 gigabyte (GB) de RAM (1,5 GB si se ejecuta en una máquina virtual)
- 1 GB de espacio disponible en el disco duro.
- Unidad de disco duro de 5400 RPM.
- Tarjeta de vídeo compatible con DirectX 9 que funcione con una resolución de 1024 x 768 o superior. (Visual Studio, s.f.)

Runbook Designer

Runbook Designer es la herramienta que se utiliza para crear, administrar y ejecutar los runbooks en System Center 2012 – Orchestrator.

Requerimientos

Sistemas Operativos Compatibles:

- Windows Server 2008 R2
- Windows Server 2012
- Windows 7, 32 o 64 bits

Requisitos de Hardaware

- 1 gigabyte (GB) de RAM como mínimo, 2 GB o más recomendado
- 200 megabytes (MB) de espacio disponible en disco duro
- Microprocesador Intel de doble núcleo de 2,1 gigahercios (GHz) o superior. (Microsoft System Center, s.f.)

SQL Server

Microsoft® SQL Server™ es un sistema de administración y análisis de bases de datos relacionales de Microsoft para soluciones de comercio electrónico, línea de negocio y almacenamiento de datos.

Requerimientos

Sistemas Operativos Compatibles:

- Windows Server 2008 SP2 Standard, Enterprise y Datacenter
- Windows Server 2008 R2 SP1 Standard, Enterprise y Datacenter.
- Windows Server 2012 Datacenter y Standard.

Requisitos de Hardaware

- SQL Server 2012 requiere un mínimo de 6 GB de espacio disponible en disco.
- 1 gigabyte (GB) de RAM como mínimo, 4 GB o más recomendado y debe aumentar a medida que el tamaño de la base de datos aumente para asegurar un rendimiento óptimo.
- Procesador x64: AMD Opteron, AMD Athlon 64, Intel Xeon compatible con Intel EM64T Intel Pentium IV compatible con EM64T de 1.0 GHZ o superior
- Procesador x86: compatible con Pentium III o superior de 1.4 GHz o superior. (TechNet Microsoft, s.f.)

Telerik Reporting

Como lo mensiona Telerik en el 2009, permite almacenar, gestionar y visualizar reportes a través de la interfaz web. Proporciona un acceso transparente a los informes de usuarios dentro de la organización desde cualquier dispositivo.

Sistemas Operativos Compatibles:

- All Windows versions which support .NET4+
- For Azure applications: Windows Azure SDK 1.3+

Soporte .NET y IIS

- .NET Framework 4+
- Todas las versiones de Internet Information Services que soporten .NET4 + (IIS) 6+

Soporte IDE

- Microsoft Visual Studio 2010, 2012, 2013, 2015
- Las versiones beta de Visual Studio no se admiten

Soporte Navegadores

Tabla 9: Lista de navegadores compatibles con Telerik

	Windows	**Mac OS**	**Linux**	**Mobile**
Internet Explorer	9.0+	-	-	HTML5-compliant versions
Firefox	HTML5-compliant versions	HTML5-compliant versions	HTML5-compliant versions	HTML5-compliant versions
Chrome	HTML5-compliant versions	HTML5-compliant versions	HTML5-compliant versions	HTML5-compliant versions
Opera	10+	10+	10+	HTML5-compliant versions
Safari	4.0+	4.0+	-	HTML5-compliant versions

Tomado de: (Telerik, s.f.)

Team Foundation Server 2012

> “Team Foundation Server (TFS) ofrece un conjunto de herramientas de colaboración que funcionan con su editor o IDE existente, para que su equipo pueda trabajar de manera eficiente en proyectos de software de cualquier índole y envergadura” (Visual Studio, s.f.)

A continuación, se listan las características principales del TFS:

- Control de versión.
- Herramientas para equipos ágiles.
- Integración continua.
- Lenguaje y herramientas.
- Integración (servicio de terceros)

Requerimientos:

Tabla 10: Sistema Operativo – Servidor

TFS Version	**Supported server operating systems**
TFS 2012	Windows Server 2012 R2 (Essentials, Standard, Datacenter)
	Windows Server 2012 (Essentials, Standard, Datacenter)
	Windows Server 2008 R2 (Standard, Enterprise, Datacenter)
	Windows Server 2008 (minimum SP2)
	Windows Small Business Server 2011 (Standard, Essentials, Premium Add-On)
	Windows Home Server 2011

Adaptado de Visual Studio, s.f.

Tabla 11: Sistema Operativo – Cliente

TFS Version	**Supported client operating systems**
TFS 2012	Windows 8.1 (Basic, Professional, Enterprise)
	Windows 7 (Home Premium, Professional, Enterprise, Ultimate)

Adaptado de Visual Studio, s.f.

Tabla 12: Versión de SQL Server

TFS Version	Supported SQL Server versión
TFS 2012	SQL Sever 2012
	SQL Sever 2008 R2

Adaptado de Visual Studio, s.f.

Tabla 13: Versión de Sharepoint

TFS Version	Supported SharePoint versions
TFS 2012	SharePoint 2013 (Foundation, Standard, Enterprise)
	SharePoint 2010 (Foundation, Standard, Enterprise)
	Office SharePoint Server 2007 (Standard, Enterprise)
	Windows SharePoint Services 3.0

Adaptado de Visual Studio, s.f.

Tabla 14: Versión de Office

TFS Version	Supported Office versions
TFS 2012	Office 2010, 2007

Adaptado de Visual Studio, s.f.

Tabla 15: Versión de Build

TFS Version	Supported Build versions
TFS 2012	TFS 2012 Xaml Controller
	TFS 2010 Xaml Controller

Adaptado de Visual Studio, s.f.

En cuanto a las recomendaciones de hardware Visual Studio maneja dos opciones del servidor de despliegue (deploymet):

Despliegue servidor simple (Single Server deployment), la recomendación mínima es un procesador dual-core con 4 GB de RAM con 500 usuarios.

Despliegue múltiples servidores (Multi Server deployment), la recomendación Mínima un procesador dual-core con 8 GB de RAM más de 500 usuarios.

Release Management

> “Cuanto más rápida la implementación del software, menos tardará en recibir comentarios Con una administración de versiones en Visual Studio, usted puede configurar, aprobar e implementar sus aplicaciones para cualquier entorno. Cree orquestaciones de implementación automatizadas para cada entorno, sin importar la complejidad de la configuración. Al entregar el software con más frecuencia y facilidad a un entorno, los evaluadores pueden validar antes el sistema y las partes interesadas siguen implicadas en el proceso de entrega de comentarios” (Visual Studio, s.f.)

Visual Studio menciona los siguientes requerimientos:
Servidor – Administrador del Servidor (Release Management Server)

Sistemas operativos:

- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2, Service Pack 1

Team Foundation Server (TFS):

- Team Foundation Server 2015
- Team Foundation Server 2013
- Team Foundation Server 2012
- Team Foundation Server 2010

SQL Server

- SQL Server 2014
- SQL Server 2012

- SQL Server 2008 R2
- SQL Server 2008

Tabla 16: Requisitos de hardware

Requisito	**Mínima**	**Se recomienda**
CPU	Procesador Pentium de 1 GHz o equivalente	Procesador Pentium de 2 GHz o equivalente
RAM	1 GB	2 GB
Disco duro	2,2 GB	3,2 GB para el primer año. El tamaño de la base de datos puede aumentar hasta 1 GB por año o tener tan solo 10 MB. Depende del uso.
Pantalla	1024 x 768 color de alta densidad, colores de 16 bits	1280 x 1024 color de alta densidad de 32 bits

Tomado de Visual Studio, s.f.

Cliente – Administrador de Lanzamiento (Release Management Client)

Sistemas Operativos:

- Windows 8.1
- Windows 8
- Windows 7, Service Pack 1
- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2, Service Pack 1

Team Foundation Server (TFS):

- Team Foundation Server 2015
- Team Foundation Server 2013
- Team Foundation Server 2012
- Team Foundation Server 2010

Tabla 17: requisitos de hardware

Requisito	**Mínima**	**Se recomienda**
CPU	Procesador Pentium de 1 GHz o equivalente	Procesador Pentium de 2 GHz o equivalente
RAM	512 MB	1 GB
Disco duro	2,2 GB[1]	Igual que el mínimo
Pantalla	1024 x 768 color de alta densidad, colores de 16 bits	1280 x 1024 color de alta densidad de 32 bits

Tomado de Visual Studio, s.f.

Agente de implementación de Microsoft (Microsodt deployment agent)

Sistema Operativo:

- Windows 8.1
- Windows 8
- Windows 7, Service Pack 1 (obtener SP1)
- Vista, Service Pack 2 (obtener SP 2), PowerShell 2.0 (obtener PowerShell 2.0 para Vista)
- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2, Service Pack 1
- Windows Server 2008, Service Pack 2 (obtener SP2), PowerShell 2.0 (obtener PowerShell 2.0 para Windows Server 2008)

Tabla 18: Requerimientos de hardware

Requisito	**Mínima**	**Se recomienda**
CPU	Procesador Pentium de 400 MHz o equivalente	Procesador Pentium de 1 GHz o equivalente
RAM	256 MB	512 MB
Disco duro	2,2 GB	Además del espacio en disco mínimo necesario para el agente de implementación, también necesita suficiente espacio en disco para implementar la aplicación.
Pantalla	1024 x 768 color de alta densidad, colores de 16 bits	1280 x 1024 color de alta densidad de 32 bits

Tomado de Visual Studio, s.f.

2.4 Línea de Datos

En esta fase encontramos las siguientes tareas: modelo dimensional, diseño físico y diseño e implementación del ETL. A continuación, vamos a tratar cada una de estas actividades.

2.4.1 Modelo Dimensional

A continuación, se describe el proceso del desarrollo del modelo multidimensional.

2.4.1.1 Procesos del Negocio

Los procesos seleccionados para la generación de indicadores son la gestión del ingreso de nuevos clientes y la apertura cuentas y el segundo es el valor del monto de apertura de las cuentas de ahorros.

2.4.1.2 Dimensiones

Las dimensiones que se crearán para implementar el modelo son:

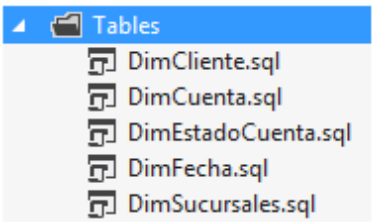


Figura 5: Tablas de Dimensiones

Se adjunta en la sección de anexos el diccionario de datos del modelo (ver anexo 2)

2.4.1.3 Jerarquía

Dentro de las dimensiones se pueden formar elementos jerárquicos relacionados, como Latino BI (2013) expresa La agrupación de dimensiones según una relación de dependencia lógica, se le denomina **jerarquías.**

A continuación, se exponen los elementos jerárquicos de las dimensiones creadas:
Dimensión: DimSucursal
Jerarquía: Ciudad -> Sucursal
Dimensión: DimFecha
Jerarquía: Año -> Mes

2.4.1.4 Dimensiones y Tablas de Hechos

Con el empleo de la herramienta Visual Studio se crea la tabla de hechos y sus dimensiones, para poder generar la estructura. A continuación, en la figura 6 se muestra la creación de la dimensión del cliente:

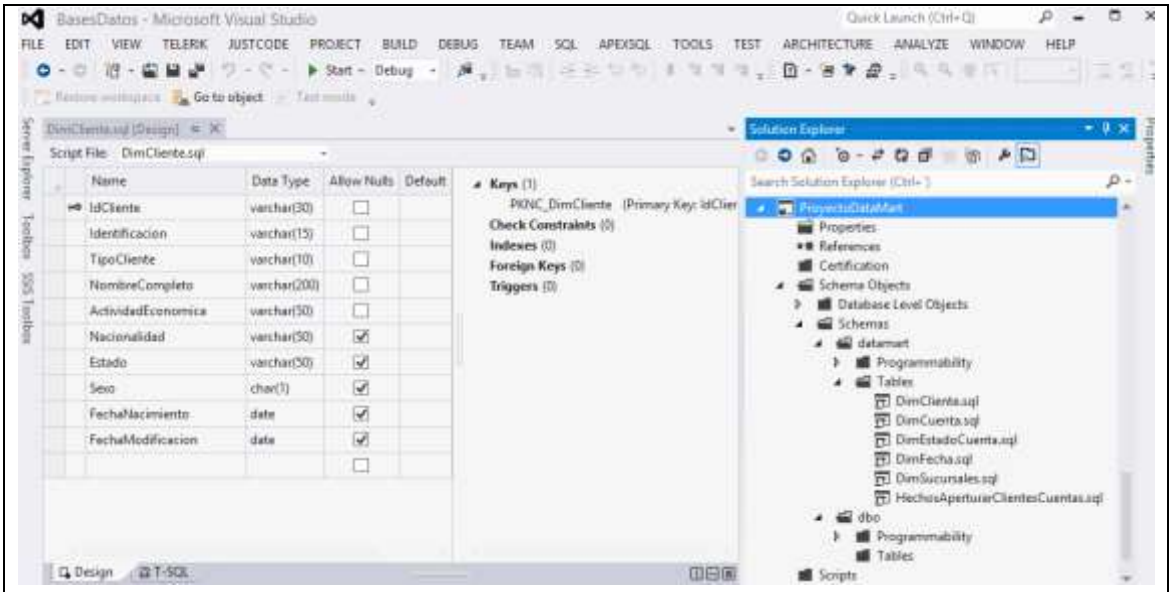


Figura 6: Creación de tablas dimensionales y hechos

Se procede a generar el modelo de dimensiones con su correspondiente tabla de hechos del proceso de análisis de requisitos. En la siguiente figura se muestra la relación de la tabla de hechos con sus dimensiones:

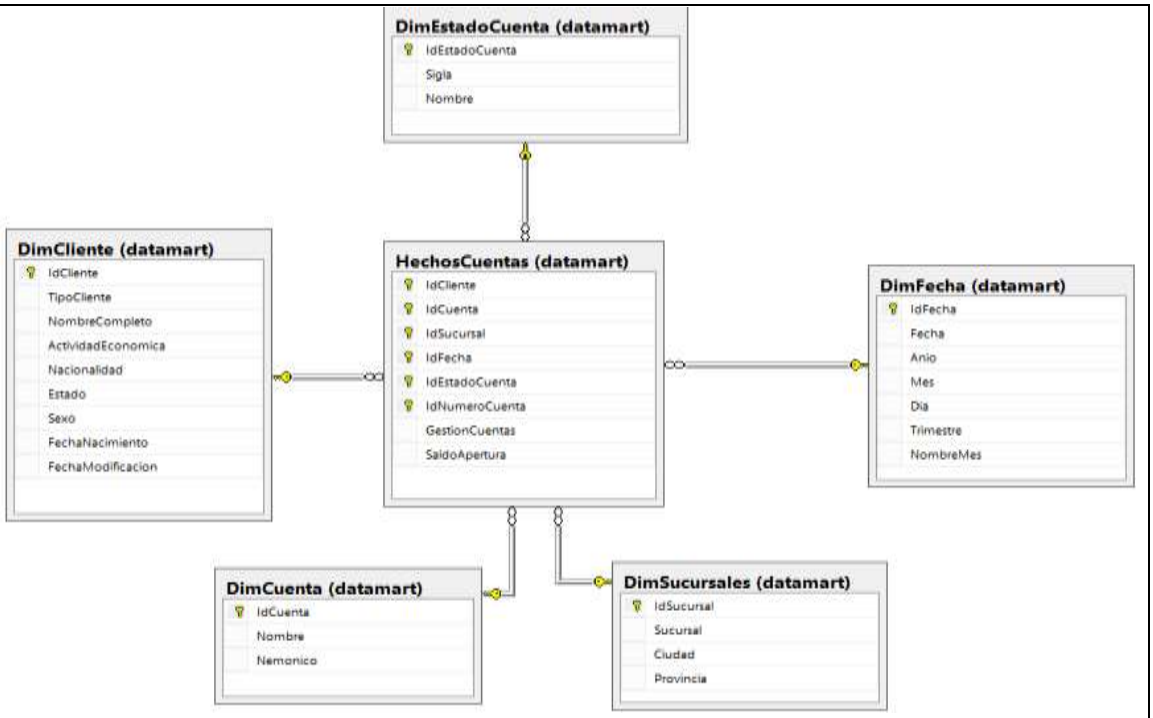


Figura 7: Relación entre las tablas de dimensiones y hechos

2.4.2 Diseño Físico

A continuación, se describe el proceso del diseño físico.

2.4.2.1 Preparación Base de Datos

El entorno de la base de datos se realiza utilizando Visual Studio, con la creación del proyecto ProyectoDataMart y el uso de la plantilla genérica del proyec SQL Server Database Project.

A continuación, se presenta la pantalla principal donde se realizarán las configuraciones necesarias para la creación del proyecto:

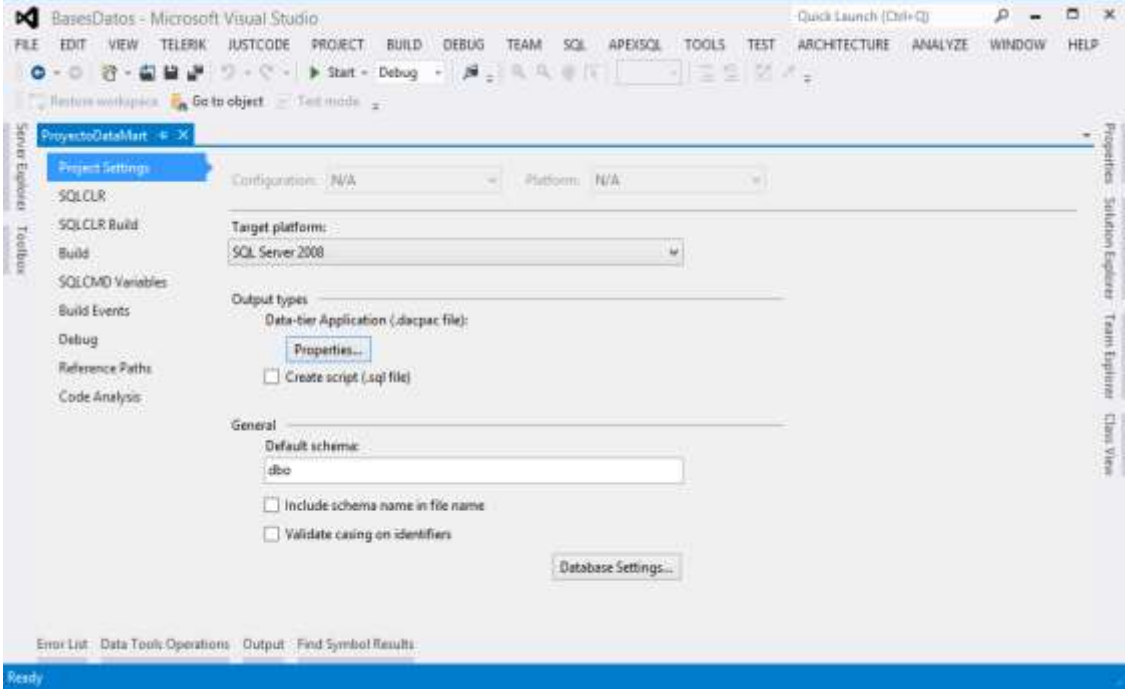


Figura 8: Configuración proyecto base de datos.

Para la configuración de la base de datos, se deben dar click en el bobón Database Settings como se muestra en la figura 7, se realiza la selección de las siguientes opciones comunes, operacionales y varios como se observa en las figuras 9, 10 y 11:

Comunes (Common)**:**

Figura 9: Configuración base de datos, comunes

Operacional (Operational):

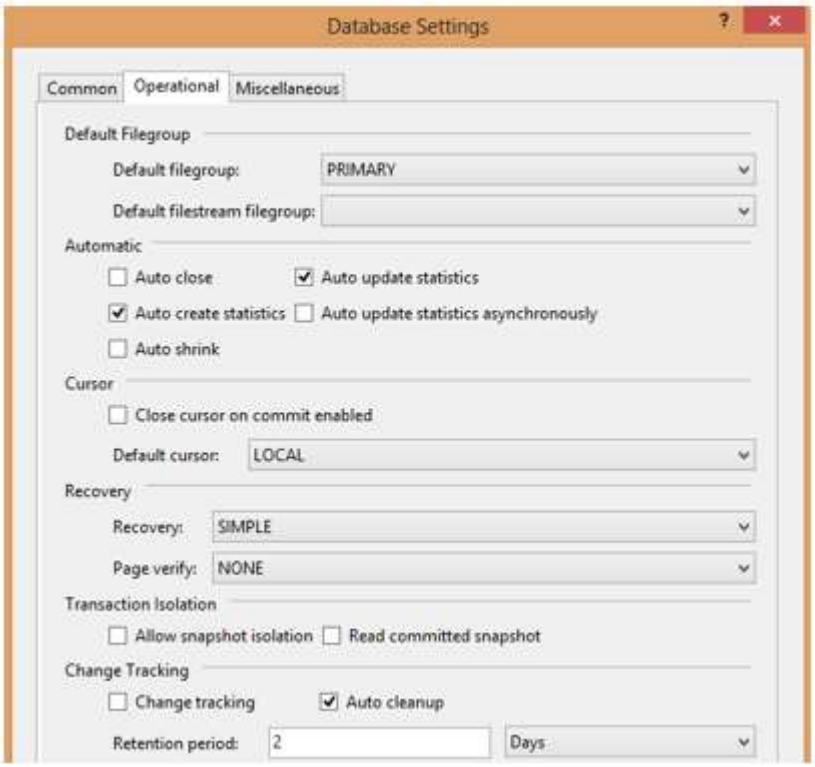


Figura 10: Configuración base de datos, operacional.

Varios (Miscellaneous)

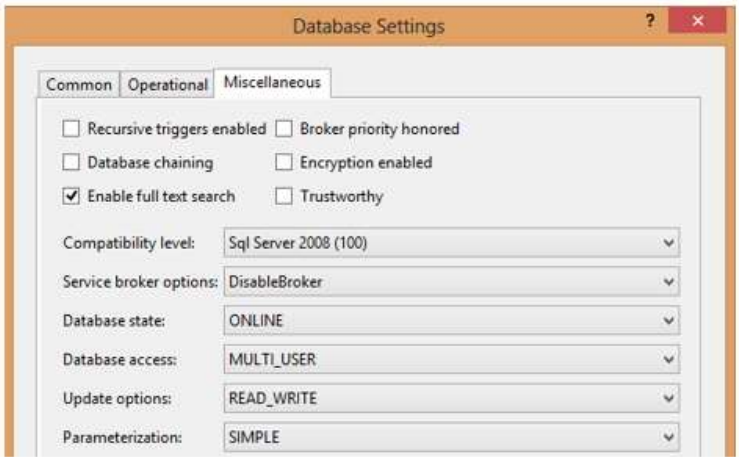


Figura 11: Configuración base de datos, mescellaneous

En las subsiguientes pestañas son configuraciones propias del proyecto: SQLCLR, SQLCLR Build, Build, SQLCMD Variables, Build Events, Debug, Code Analysis.

Luego de realizar el afinamiento de las configuraciones de la base de datos y el proyecto, continuamos con la creación de la estructura de nuestra solución se crean carpetas que contendrán loa filegroup, los esquemas que se han necesarios a continuación se presenta gráficamente lo que se ha creado para el presente proyecto:

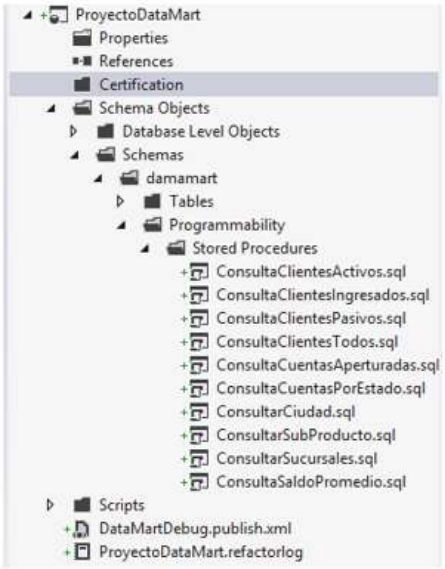


Figura 12: Estructura del proyecto Datamart

En la sección azul de la anterior figura está el archivo DataMartDebug.publish.xml, permite ejecutar y configurar donde se creará la base de datos.

2.4.2.2 Seguridades

La seguridad para la base de datos contará con el tipo de control de acceso discrecional. En la actualidad se habla de dos tipos de mecanismos de seguridad en las bases de datos, nos referiremos únicamente al seleccionado para el proyecto:

> "Los mecanismos de seguridad discrecional para otorgar privilegios a los usuarios, incluido el acceso a archivos, registros o campos de datos específicos en un determinado modo.
> Además, solo se manejará un único nivel de privilegio: Nivel de cuenta, en este nivel el administrador especifica los privilegios particulares que tiene cada usuario, independiente de las tablas de la BD (CREATE TABLE, CREATE VIEW ALTER, MODIFY, SELECT)" (Domínguez, 2014, p. 6 - 11)

A continuación, se crear el script para la generación del usuario en la base de datos, figura 13. Adicional para realizar el paso de versión el equipo de desarrollo debe generar el script para la creación de la base de datos y la persona encargada de la seguridad deberá realizar el cambio y actualización de las claves de acceso como lo estime necesario.

```
IF db_name() = 'CooprogresoOdsOut'
BEGIN
    CREATE USER [DataMart] FOR LOGIN [DataMart2016] WITH DEFAULT_SCHEMA=[dbo];
    EXEC sp_addrolemember 'db_owner', 'cooprogreso';
END
```

Figura 13: Script generación de usuario.

2.4.2.3 Afinamiento

Se prioriza la creación y utilización de índices en las tablas como parte del afinamiento de la base de datos.

> “Al igual que el índice de un libro, el índice de una base de datos permite encontrar rápidamente información específica en una tabla o vista indizada. Un índice contiene claves generadas a partir de una o varias columnas de la tabla o la vista y punteros que asignan la ubicación de almacenamiento de los datos especificados. Puede mejorar notablemente el rendimiento de las aplicaciones y consultas de bases de datos creando índices correctamente diseñados para que sean compatibles con las consultas. Los índices pueden reducir la cantidad de datos que se deben leer para devolver el conjunto de resultados de la consulta. Los índices también pueden exigir la unicidad en las filas de una tabla, lo que se garantiza la integridad de los datos de la tabla” (TechNet Microsoft, 2012)

Cada una de las tablas se encuentra con su respectivo índice, se crea a partir de las claves primarias de las tablas dimensionales, así como las claves foráneas en la tabla de hechos.

2.4.3 Diseño y Desarrollo ETL

En la sección del diseño de la arquitectura (2.3.2) se habla de la herramienta que se utilizará para la construcción del ETL, Integration Services Project. A continuación, se describe los pasos del respectivo desarrollo:

a. Como primer punto se toma en cuenta el proceso ETL (extracción, transformación y carga), y la estrategia que se aplicará para el proyecto. Siguiendo este paso se decide crear un esquema que contendrá tablas dimensionales y de hechos para para el proceso seleccionado. En cuanto a las etapas de extracción y transformación se hace el uso de un esquema ODS (Operational Data Store).

> “El ODS es una base de datos sobre la que se aplica una o varias bases de los sistemas transaccionales. Esta replica puede incluir además la identificación de vínculos entre las diferentes fuentes o la supresión de datos anómalos. En un ODS los datos son volátiles” (Gauchet, 2015, p. 26)

b. En el siguiente paso se selecciona y crean las conexiones necesarias para el origen de datos y su destino, así como también las variables que se estimen necesarias:

Conexiones:

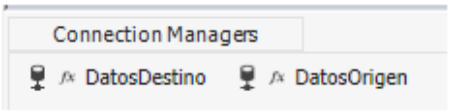


Figura 14: Conexión de las bases de datos

Variables:

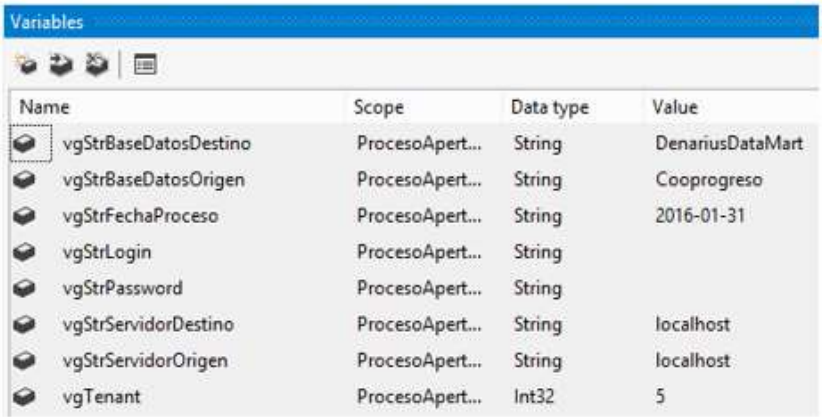


Figura 15: Variables creadas en el ETL

c. En este paso con el uso de la herramienta se crea un contenedor de secuencias (Sequence Container), que manejará el proceso ETL (extracción transformación y carga) y permitirá el poblamiento de las tablas dimensionales y de hechos, a continuación, se presenta gráficamente en la figura 16:

Figura 16: Creación del contenedor ETL

d. Continuamos con la creación de los elementos necesarios para la extracción y transformación como se muestra en la figura 17:

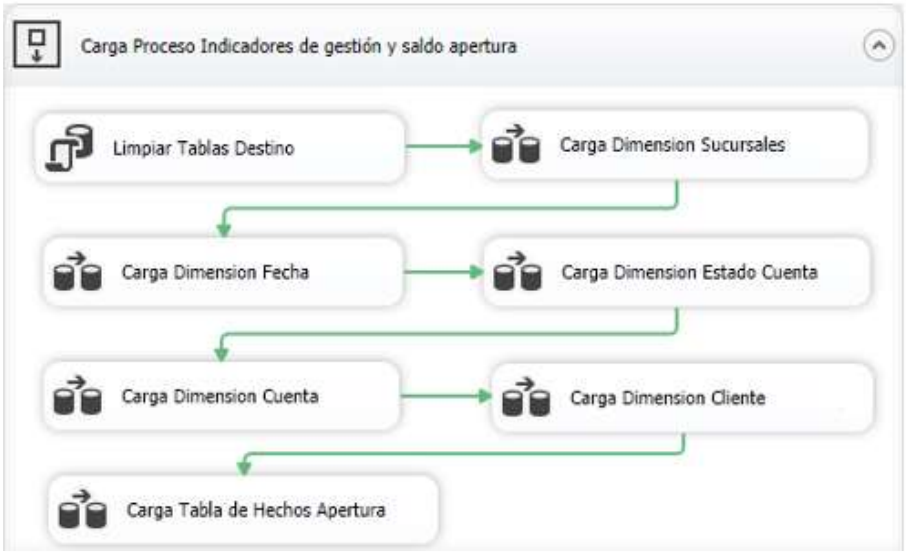


Figura 17: Creación estructura ETL

Se crean una tarea de ejecución SQL (Execute SQL Task) que manejan el limpiado de las tablas de destino, para realizar la carga de data se emplean los datos de flujos de tareas (Data Flow Task)

Una vez terminado la creación de la estructura del proceso ETL, se procede con los procesos que nos permiten realizar la carga de datos para la visualización de los indicadores.

Con respecto a la utilización de herramientas ETL se hace énfasis en la integración con los datos de origen es por esta razón que Kimball y Caserta (2004, p. 216) que menciona, muchas de las herramientas ETL han integrado depósitos de metadatos que pueden sincronizar los metadatos de los sistemas de origen, base de datos de datos de destino, y otras herramientas de BI.

a. Como primer paso procedemos a ingresar y configurar la sentencia en la tarea SQL: Limpiar Tablas Destino, como se muestra en la figura 18.

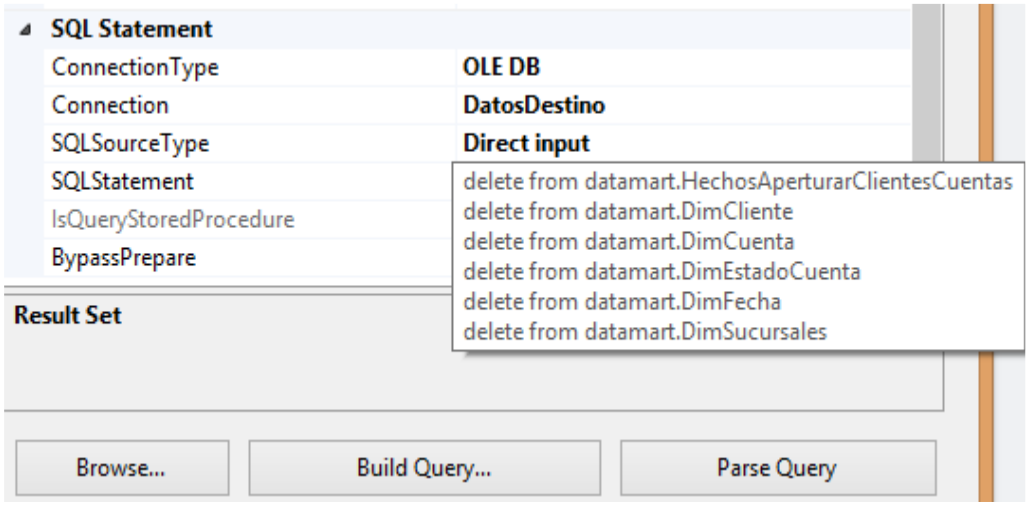


Figura 18: Configuración de la tarea de ejecución SQL

En este paso se debe seleccionar la conexión adecuada (DatosDestino) e ingresar la sentencia SQL como se muestra en la figura de este paso.

b. A continuación procedemos a ingresar y configurar los datos de flujos de tareas, de cada una que fue creada en la sección 2.5.1. Se tomará como ejemplo: Carga Dimensión Cliente, para explicar el proceso.

c. En este paso se va a insertar dos nuevos elementos dentro del flujo de tarea: OLE DB Source y OLE DB Destination, se encuentran ubicados dentro de la caja de herramientas del SSIS (SQL Server Integration Services) y arrastramos los componentes mencionados anteriormente tal como se muestra en la figura 19:

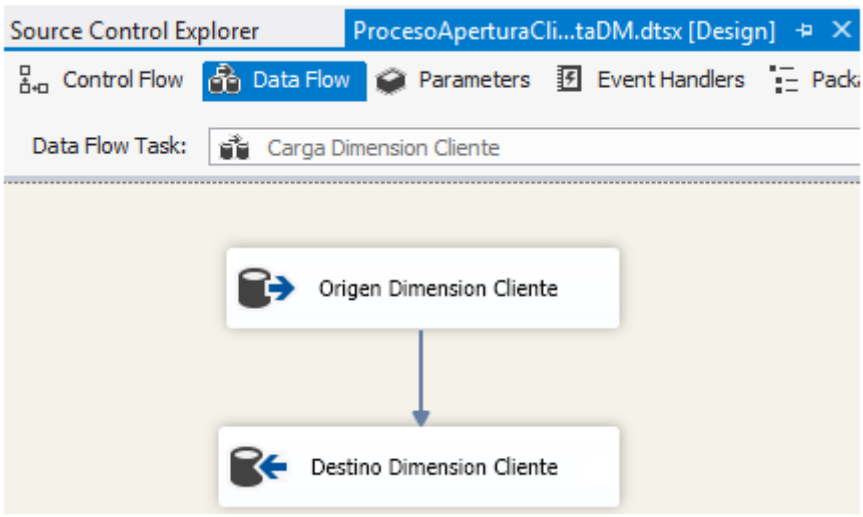


Figura 19: Elementos relacionados dentro del flujo de tareas

d. Procedemos a realizar las configuraciones necesarias en el elemento OLE DB Source, se seccionamos la conexión, el modo del tipo de acceso y por último la sentencia SQL tal como se muestra la figura 20:

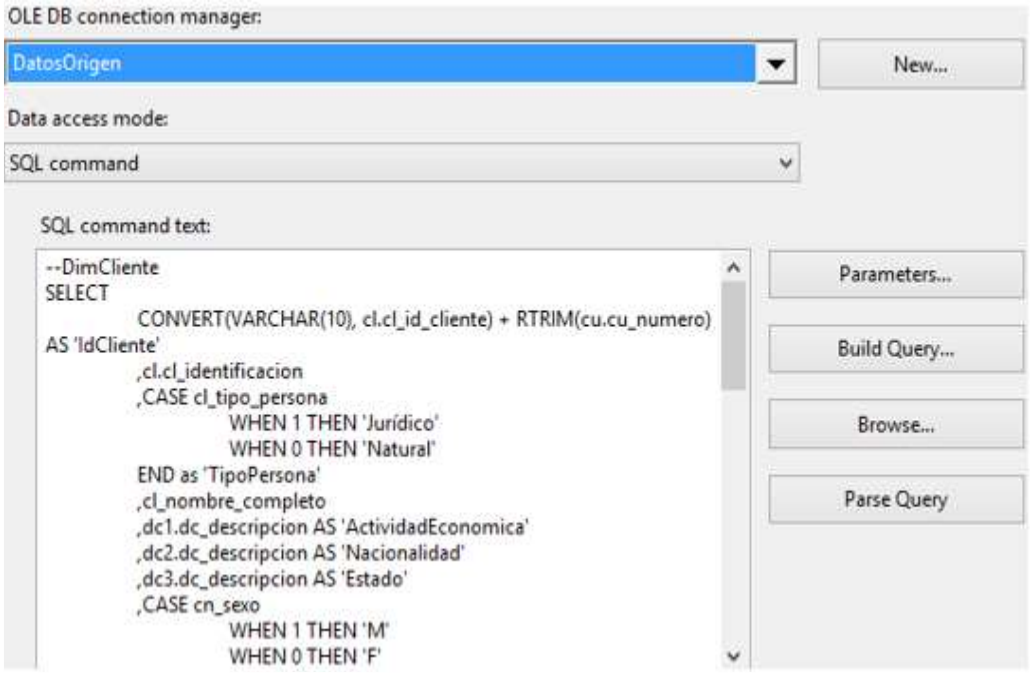


Figura 20: Configuración elemento de origen en el flujo de tareas.

e. En este paso procedemos a realizar las configuraciones necesarias en el elemento OLE DB Destination se seccionamos la conexión, el modo del tipo de acceso, tal como se muestra en la figura 21:

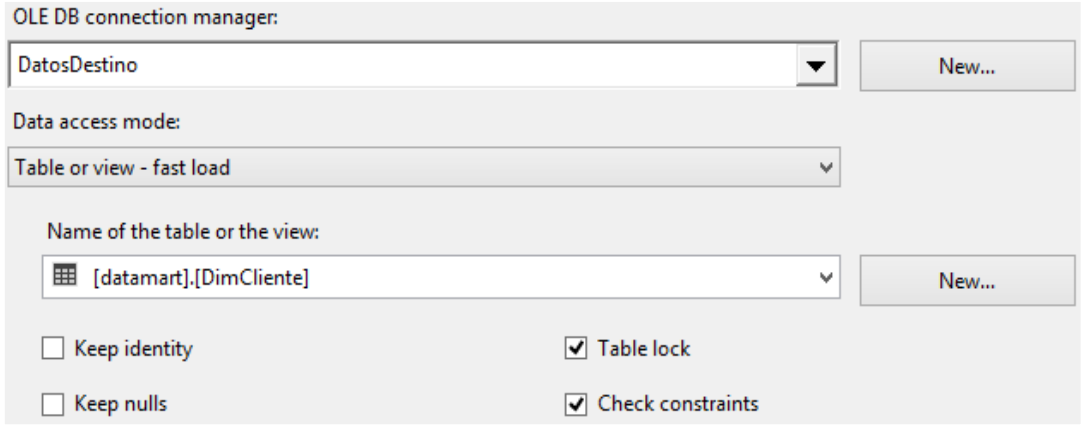


Figura 21: Configuración elemento destino en el flujo de tareas.

f. Se procede a realizar el mapeo de los campos entre los datos de origen y de destino de tal manera que corresponda ambos lados, a continuación, se presenta el mapeo ya realizado como se muestra en la figura 22:

Input Column	Destination Column
IdCliente	IdCliente
TipoPersona	TipoCliente
cl_nombre_completo	NombreCompleto
ActividadEconomica	ActividadEconomica
Nacionalidad	Nacionalidad
Estado	Estado
Sexo	Sexo
cn_fecha_nacimiento	FechaNacimiento
cl_fecha_modificacion	FechaModificacion
cl_identificacion	Identificacion

Figura 22: Mapeo de los campos de origen y llegada.

Se procederá a realizar los pasos c, d, e y f para cada uno de los flujos de trabajo. A continuación, se presenta los flujos de tareas restantes y los scripts se encuentran en el anexo 3:

2.5 Línea de Aplicación BI

En esta fase encontramos las siguientes tareas: diseño de la aplicación de BI, desarrollo de la aplicación de BI. A continuación, vamos a tratar cada una de estas actividades.

2.5.1 Diseño de la aplicación de BI

Entre sus actividades encontramos el diseño de la arquitectura y el diseño de interfaces.

2.5.1.1 Diseño arquitectura

A continuación, en la figura 23 se representa la arquitectura de la aplicación BI que va integrado al core financiero.

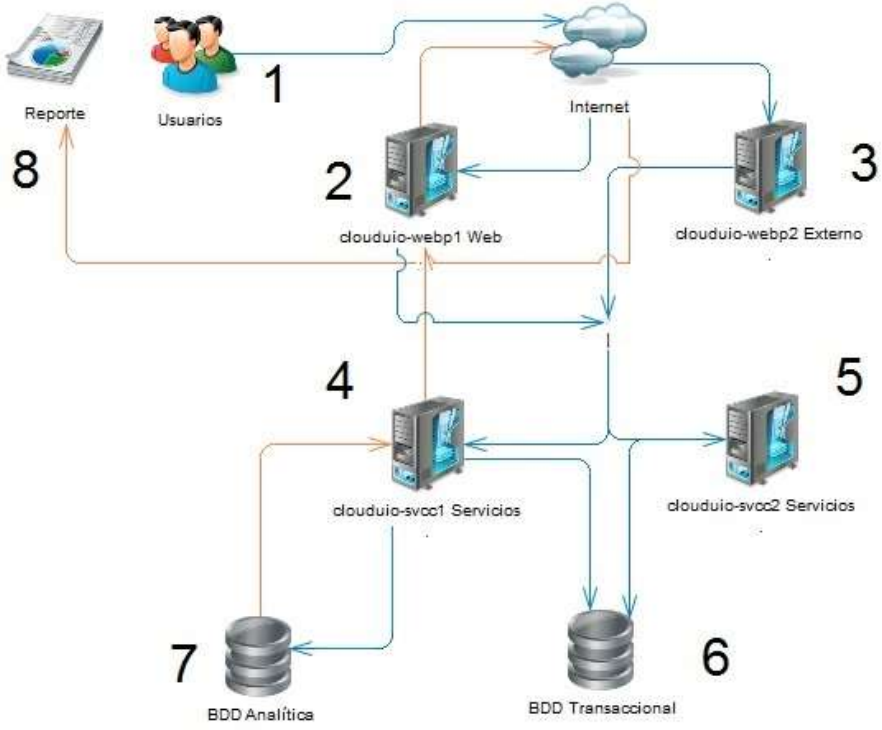


Figura 23: Arquitectura del Core denarius.

a. Las flechas de color azul son el camino de ingreso al sistema y las flechas de color naranja es la respuesta a la solicitud.

Como se observa en la figura 23 la arquitectura muestra diferentes elementos que la componen, a continuación, en la tabla 25 se describen cada uno de ellos.

Tabla 19: Descripción de los elementos que componen la arquitectura del Core Denarius.

#	**Componente**	**Descripción**	**Características**
1	Usuarios	Representan los usuarios de la entidad financiera.	Se autentifican por medio del active directory.
	Internet	Medio por el cual se ingresa al core.	Servicio enfocado en computación en la nube
2	Web	Los usuarios autentificados ingresan por el servidor al core.	6 procesadores (2.80 GHz) Ram: 10.00 Gb
3	Externo	Ingreso de servicios externos de los satélites (Servipagos)	6 procesadores (2.80 GHz) Ram: 8.00 Gb
4	Servicios Web	Servidores de servicios para usuarios de la entidad financiera.	4 procesadores (2.80 GHz) Ram: 6.00 Gb
5	Servicios Externos	Servidores de servicios externos de la entidad financiera.	4 procesadores (2.80 GHz) Ram: 6.00 Gb
6	BDD Transaccional	Base de datos transaccional.	4 procesadores (2.80 GHz) Ram: 6.00 Gb
7	BDD Analítica	Base de datos analítica.	
8	Reportes	Son los reportes consultados a través del core por los usuarios.	

2.5.2 Desarrollo de la Aplicación de BI

A continuación, se emplea la metodología de scrum para el desarrollo del aplicativo. Entre las actividades a revisar están: Product BackLog, Sprint BackLog, Iteration, Scrum Daily Meeting (reunión diaria del equipo), Product Incremente. Para el presente proyecto el Product Owner es la UDLA y los interesados (stakeholders) es Denariuservis SA.

2.5.2.1 Product Backlog

El primer paso a tomar en cuenta es el Product BackLog, es un documento que contiene una lista genérica de los requerimientos priorizados. A continuación, se crea la plantilla del product backlog que se manejara en la aplicación de BI, contiene en total tres historias de usuario se identifican con la letra *H* luego con un valor numérico, ver tabla 19:

Tabla 20: Product Backlog

Id	Enunciado de la Historia	Estado	SP	Iteración (Sprint)	Prioridad
H1	Preparación del ambiente de desarrollo		0	Sprint 0	
H2	Los gerentes, necesitan visualizar el número de cuentas apertura e ingreso de nuevos clientes a partir del 2016, con la finalidad de observar la tendencia del proceso.	Planificada	15	Sprint 1	1
H3	Los gerentes, necesitan visualizar el promedio del saldo de cuentas apertura a partir del 2016, con la finalidad de visualizar la tendencia del proceso.	Planificada	15	Sprint 1	2

Nota: La columna Story points (SP) corresponde al esfuerzo que toma realizar las historias. Adaptado de PMOinformatica, s.f.

La primera historia de usuario ***H1*** corresponde al sprint cero y las siguientes dos ***H2*** y ***H3*** corresponden al desarrollo de la aplicación.

2.5.2.2 Planificación de la iteración (Sprint Planning Meeting)

Como se mencionó anteriormente en la planificación de la iteración se toman las historias de usuario del Product BackLog, luego de lo cual el equipo de desarrollo crea las tareas necesarias para realizar la historia. Se debe tomar en cuenta las historias de usuario que son entregadas por el Product Owner.

2.5.2.2.1 Historia de usuario H1

A continuación, se presentan en detalle la historia de usuario del sprint 0, Preparación del ambiente de desarrollo.

Título Historia de Usuario
Preparación del ambiente de desarrollo.

Descripción
A continuación, se explica las necesidades para la creación del ambiente de desarrollo:

Verificación proyectos actuales:

- En este punto se verifica la factibilidad de la creación de un nuevo proyecto para el desarrollo de lo solicitado en las historias de usuario descritas en el Product BackLog.

Afinamiento proyectos:

- Dependiendo del paso de Verificación de proyectos actuales si se crea un nuevo proyecto se deberá proceder con la configuración necesaria del ambiente o por el contrario si se decide trabajar sobre uno ya existente se actualizará la última versión del código.

Revisar complementos del proyecto:

- En este punto se debe identificar si es necesario utilizar algún complemente o herramienta adicional para el desarrollo de las historias.

Control de cambios

- Responsable: Franklin T.
- Razón: Propuesta inicial
- Fecha 25/06/2016

2.5.2.2.2 Historia de usuario H2 y H3

En cuanto a las historias H2 y H3, son entregadas por el Product Owner, a continuación, se describen en detalla:

Titulo Historia(s) de usuario

- Visualización del número de cuentas aperturadas e ingreso de clientes (***H2***).
- Visualización de saldos promedios de apertura de cuentas de ahorros (***H3***).

Descripción

A continuación, se explica las necesidades por las secciones de cuentas, clientes y saldos promedios de apertura:

Cuentas*:*

- El sistema debe mostrar el número total de cuentas aperturadas en un gráfico de barras, se debe considerar incluir los siguientes filtros para realizar la consulta: sucursal, fechas, edad y género de los clientes.
- El sistema debe mostrar el número total de cuentas activas en un gráfico de barras, se debe considerar incluir los siguientes filtros para realizar la consulta: sucursal, fechas, edad y género de los clientes.
- El sistema debe mostrar el número total de cuentas liquidadas en un gráfico de barras, se debe considerar incluir los siguientes filtros para realizar la consulta: sucursal, fechas, edad y género de los clientes.

Clientes:

- El sistema debe mostrar el número total de clientes ingresados en un gráfico de barras y un reporte estático del resultado de la consulta, se debe considerar incluir los siguientes filtros para realizar la consulta: sucursal y entre fechas.
- El sistema debe mostrar el número total de clientes activos en un gráfico de barras y un reporte estático del resultado de la consulta, se debe considerar incluir los siguientes filtros para realizar la consulta: sucursal y entre fechas.
- El sistema debe mostrar el número total de clientes pasivos en un gráfico de barras y un reporte estático del resultado de la consulta, se debe considerar incluir los siguientes filtros para realizar la consulta: sucursal y entre fechas.

Saldo Promedio de Apertura:

- El sistema debe mostrar el saldo promedio de apertura de cuentas en un gráfico de barras, se debe considerar incluir los siguientes filtros para realizar la consulta: sucursal, subproducto, fechas, edad y género de los clientes.

Condiciones de Satisfacción

- Validación de los gráficos presentados por pantalla.
- Validación de los filtros en la realización de la consulta.
- Validación de reportes estáticos en las consultas de clientes ingresados.

Control de cambios

- Responsable: Franklin T.
- Razón: Propuesta inicial
- Fecha 257/06/2016

2.5.2.2 Sprint Backlog

En la planificación del sprint se crea el sprint backlog, es un documento detallado que se conforma de actividades que se necesitan realizar para completar las historias de usuario del product backLog ver tabla 19. Las actividades son identificadas por el equipo de desarrollo.

A continuación, se describe en la tabla 20 las actividades del Sprint BackLog basados en la historia de usuario. Tomar en cuenta que para la nomenclatura de las historias el identificador (**Id**) se utiliza la letra **H** seguido por un secuencial. El Identificador para las actividades en el Sprint BackLog es con la letra **A** seguido por un secuencial.

Tabla 21: Sprint Backlog

Id	Enunciado de la Actividad	Estado	S.	Asignado	T.	Inicio	Fin
H1	**Preparación del ambiente de desarrollo**.						
A1	Verificación proyectos actuales	Pendiente	0	Franklin T.	4	8/6/16	8/6/16
A2	Afinamiento proyectos	Pendiente	0	Franklin T.	1	8/6/16	8/6/16
A3	Revisar complementos del proyecto	Pendiente	0	Franklin T.	1	8/6/16	8/6/16
H2	**Los gerentes, necesitan visualizar el número de cuentas apertura e ingreso de nuevos clientes a partir del 2016, con la finalidad de observar la tendencia del proceso.**						
A4	Diseño de la aplicación para la historia H2	Pendiente	1	Franklin T.	4	9/6/16	9/6/16
A5	Creación de procesos almacenados para la historia H2	Pendiente	1	Franklin T.	4	9/6/16	10/6/16
A6	Lógica para la historia H2	Pendiente	1	Franklin T.	4	10/6/16	10/6/16
A7	MVC para la historia H2	Pendiente	1	Franklin T.	6	13/6/16	13/6/16
H3	**Los gerentes, necesitan visualizar el promedio del saldo de cuentas apertura a partir del 2016, con la finalidad de visualizar la tendencia del proceso.**						
A8	Diseño de la aplicación para la historia H3	Pendiente	1	Franklin T.	4	14/6/16	14/6/16

A9	Creación de procesos almacenados para la historia H3	Pendiente	1	Franklin T.	4	14/6/16	15/6/16
A10	Lógica para la historia H3	Pendiente	1	Franklin T.	4	15/6/16	15/6/16
A11	MVC para la historia H3	Pendiente	1	Franklin T.	6	16/6/16	16/6/16
A12	Presentación demo	Pendiente	1	Franklin T.		17/6/16	17/6/16
	Corrección	Pendiente		Franklin T.			

Adaptado de PMOinformatica, s.f.
Notas: La columna T hace referencia al tiempo en horas y la columna S hace referencia al número de Sprint

2.5.2.3 **Sprint** (iteración)

Al terminar la Planificación del sprint se inicia con la ejecución de las actividades del proyecto. A continuación, se realiza la descripción del desarrollo de las actividades:

2.5.2.3.1 Sprint 0

Verificación proyectos actuales (A1)

En esta actividad del Sprint 0, se verifica si es factible la realización de las historias de usuario dentro de los ambientes actuales de desarrollo. Luego del respectivo análisis se toma la decisión de seleccionar el proyecto **DevCore** para implementar los cambios, debido a la complejidad que representa la historia de usuario.

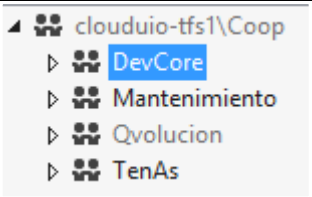


Figura 24: Proyectos desarrollo en Denarius SA

Afinamiento proyectos (A2)

En la presente actividad luego de su análisis se identifica que no es necesario realizar ningún afinamiento al proyecto de desarrollo ya existente, debido a que el proyecto seleccionado cuenta con las configuraciones requeridas. Únicamente

se baja la última versión del código del proyecto, como se muestra en la figura 25.

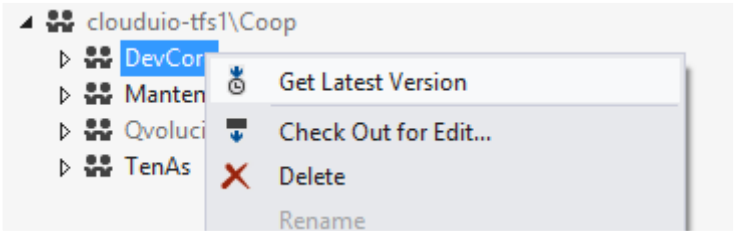


Figura 25: Actualización código versión desarrollo.

Verificación de complementos del proyecto (A3)

Se analiza si existen herramientas que son necesarias para la resolución de alguna actividad en específico, se concluye que para el proyecto no se necesita realizar la instalación o actualización de nueva herramienta.

2.5.2.3.2 Sprint 1

El Sprint 1 cuenta con dos historias de usuario (H1 y H2), con la duración de siete días hábiles como se muestra en la tabla 20.

Historia H2.- Los gerentes, necesitan visualizar el número de cuentas apertura e ingreso de nuevos clientes a partir del 2016, con la finalidad de observar la tendencia del proceso.

Diseño de la aplicación para la historia H2 (A4)

Objetivo

Realizar la vista para la generación de los indicadores de gestión.

Descripción del proceso

La vista permite generar reportes de gráficos en barras o reportes estáticos.

Diagrama Multidimensional

Se hace uso del modelo multidiemnsional creado en la sección 2.4.1.4

Diseño de la pantalla

Vista Reporte Indicadores de Gestión

Figura 26: Diseño interfaz indicadores de gestión

Se crearán enlaces para generar los reportes para cada tipo de reporte tanto para clientes como para cuentas. En la pantalla se debe mostrar el estado del cliente, un listado de las sucursales y rango de fechas como filtros de la consulta.

Botón Generar Reporte, al dar click sobre este botón permitirá realizar la consulta en la base de datos del indicador seleccionado.

Botón Limpiar, restablecerá los valores por defecto en la pantalla.

Servicios a Publicar

Servicios creados desde la lógica que son consumidos por el MVC.

- ServicioCumplimiento.ConsultaClientesIngresados

  Permite consultar el reporte de clientes ingresados.

- ServicioCumplimiento.ConsultaClientesActivos

  Permite consultar el reporte de clientes activos.

- ServicioCumplimiento ConsultaClientesPasivos

  Permite consultar el reporte de clientes pasivos.

- ServicioCumplimiento.ConsultaClientesTodos

  Permite consultar el reporte de clientes.

- ServicioCumplimiento.ConsultaCuentasAperturadas

  Permite consultar el reporte de cuentas aperturadas.

- ServicioCumplimiento ConsultaCuentasPorEstado

  Permite consultar el reporte de cuentas por estado.

Para las siguientes actividades al tratarse en su totalidad del desarrollo y debido a lo extensos que pueden ser, se procederá a indicar la actividad y un anexo donde se podrá evidenciar el código digitado.

Creación de procesos almacenados para la historia H2 (A5)

En esta actividad se procede a crear los procesos almacenados para la historia H2, de acuerdo a las especificaciones en el diseño se crean los siguientes SP:

- ConsultaClientesActivos
- ConsultaClientesIngresados
- ConsultaClientesPasivos
- ConsultaClientesTodos
- ConsultaCuentasAperturadas
- ConsultaCuentasPorEstado

El código fuente de los procedimientos almacenados está en el anexo 4.

Lógica para la historia H2 (A6)

En esta actividad se crean los servicios que van a ser consumidos desde el MVC, el código fuente está en el anexo 5

MVC para la historia H2 (A7)

Con la utilización del patrón de arquitectura MVC se desarrolla la interfaz del usuario, el código fuente se encuentra en el anexo 6.

Historia H3.- Los gerentes, necesitan visualizar el promedio del saldo de cuentas apertura a partir del 2016, con la finalidad de visualizar la tendencia del proceso.

Diseño de la aplicación para la historia H3 (A8)

Objetivo

Realizar la vista para la generación de los indicadores del saldo promedio.

Descripción del proceso

La vista permite generar reportes de gráficos en barras o reportes estáticos.

Diagrama Multidimensional

Se hace uso del modelo multidiemnsional creado en la sección 2.4.1.4

Diseño de la pantalla

Vista Reporte Indicadores de Saldo de Apertura

Figura 27: Diseño interfaz indicador saldo de apertura

En la pantalla se debe mostrar un listado de las sucursales, el tipo de subproducto, rango de fechas, rango de edad y el género del cliente como filtros de la consulta.

Botón Generar Reporte, al dar click sobre este botón permitirá realizar la consulta en la base de datos del indicador seleccionado.

Botón Limpiar, restablecerá los valores por defecto en la pantalla.

Servicio a Publicar

Servicios creados desde la lógica que son consumidos por el MVC.

- ServicioCumplimiento.ConsultaSaldoPromedio

Se procede de igual manera que con la historia H1, se indica la actividad y el anexo correspondiente.

Creación de procesos almacenados para la historia H3 (A9)

En esta actividad se procede a crear el proceso almacenado para la historia H3, de acuerdo a las especificaciones en el diseño se crea el SP: ConsultaSaldoPromedio, el código fuente del SP ver el anexo 7.

Lógica para la historia H3 (A10)

En esta actividad se crean los servicios que van a ser consumidos desde el MVC, el código fuente está en el anexo 8

MVC para la historia H3 (A11)

Con la utilización del patrón de arquitectura MVC se desarrolla la interfaz del usuario, el código fuente se encuentra en el anexo 9.

Presentación Demo (A12)

Para proceder con la presente actividad se realiza una reunión con los interesados donde se presenta el producto realizado durante el sprint. A continuación, se presentan el resultado de los reportes generados:

Indicador del número total de clientes ingresados mensualmente por sucursal

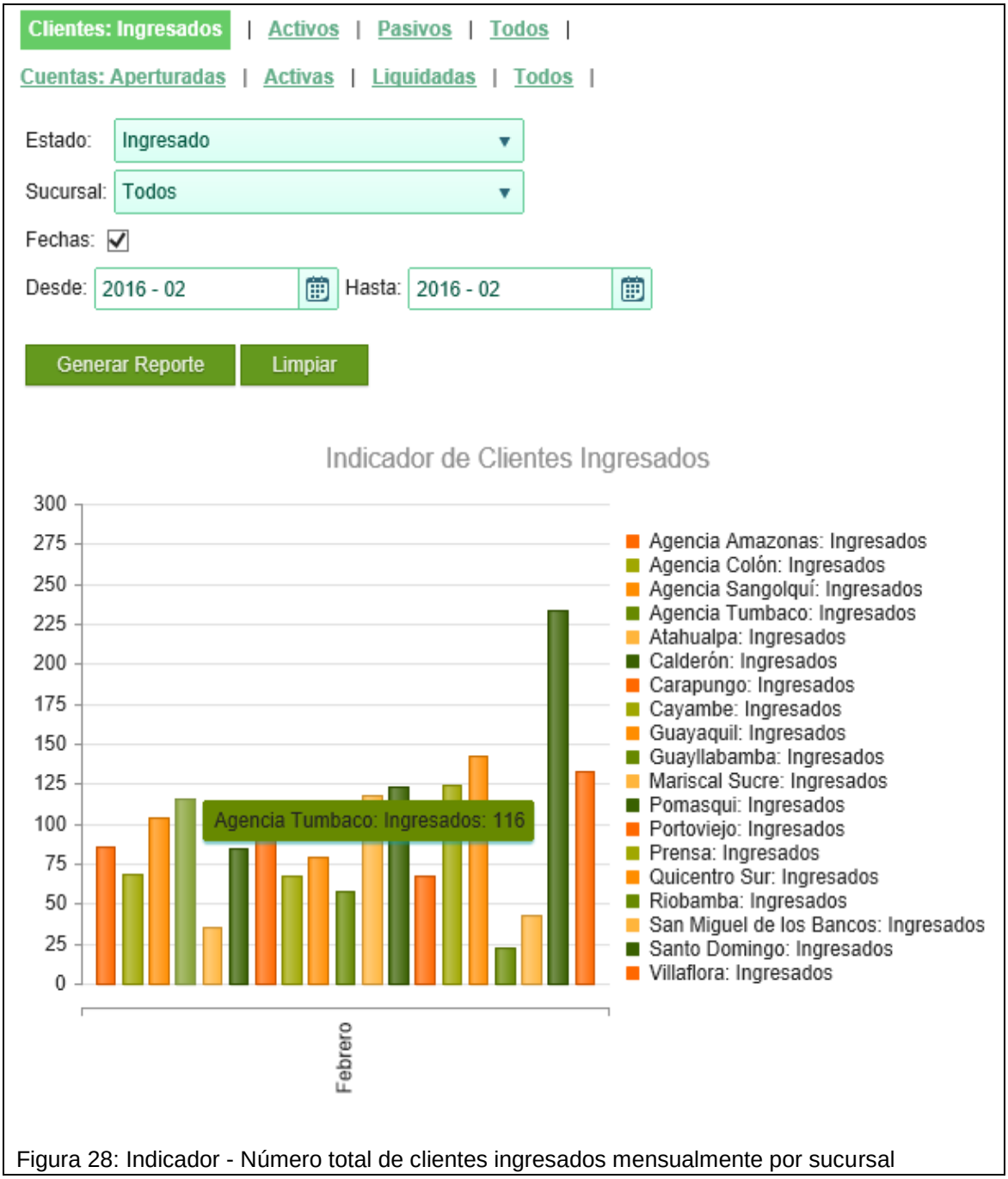


Figura 28: Indicador - Número total de clientes ingresados mensualmente por sucursal

Indicador del número total de clientes ingresados que se encuentren en estado activo mensualmente por sucursal.

Figura 29: Indicador – Número total de clientes ingresados que se encuentren en estado activo mensual por sucursal.

Indicador del número total de clientes ingresados que se encuentren en estado pasivo mensualmente por sucursal.

Figura 30: Indicador - Número total de clientes ingresados que se encuentran en estado pasivo mensualmente por sucursal

Indicador del número total de cuentas **aperturadas** por subproducto mensualmente y por sucursal.

Figura 31: Indicador – Número total de cuentas aperturadas por subproducto y mensualmente.

Indicador del número total de cuentas **activas** por subproducto mensualmente y por sucursal.

Figura 32: Indicador – Número total de cuentas activas por subproducto mensualmente y por sucursal.

Indicador de número total de cuentas **liquidadas** por subproducto mensualmente y por sucursal.

Figura 33: Indicador – Número total de cuentas liquidadas por subproducto mensualmente y por subproducto.

Indicador del saldo promedio de apertura del subproducto **cuenta de ahorros** de los clientes segmentado por sucursal desde el mes de febrero del año 2016.

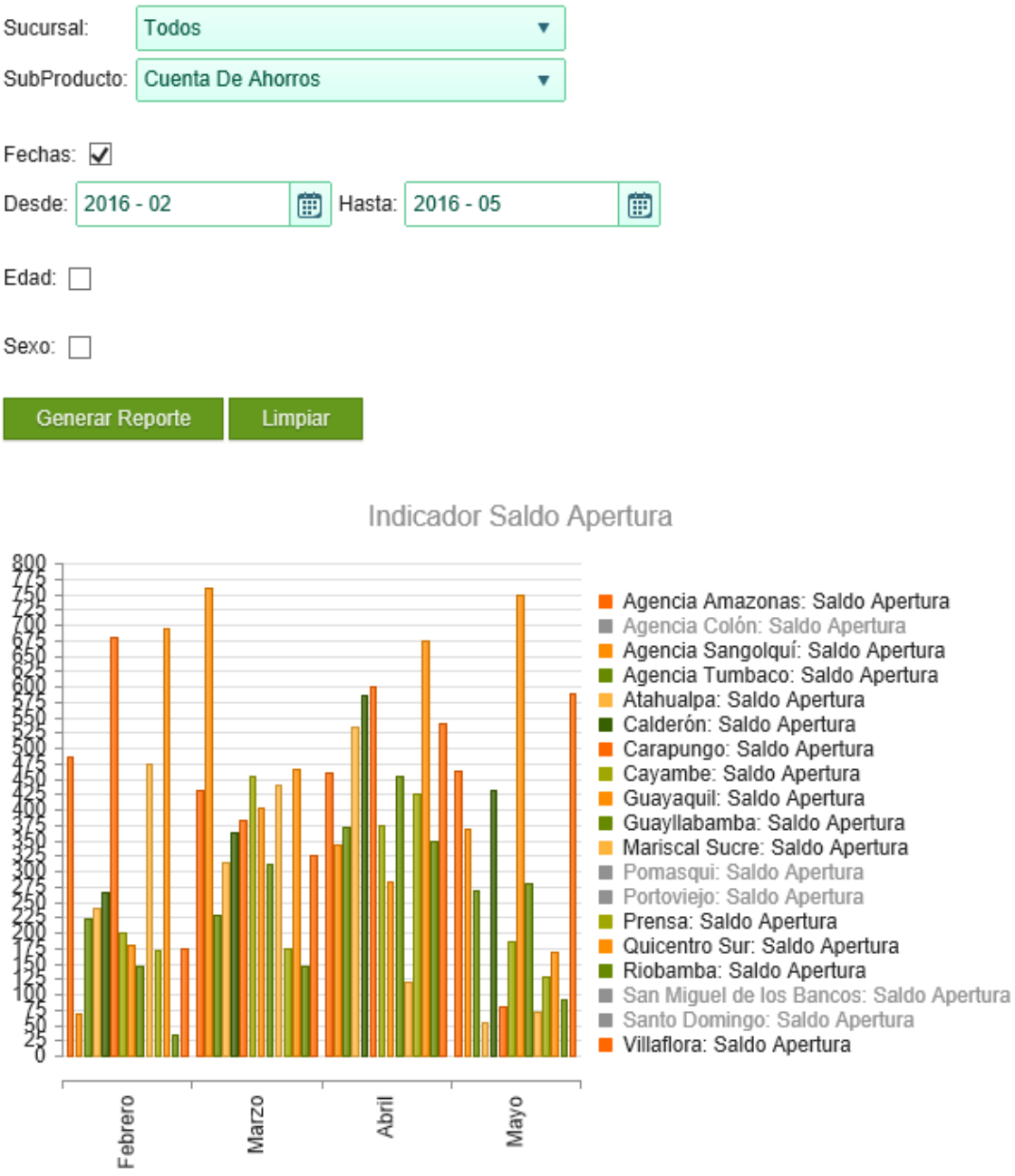


Figura 34: Indicador – Saldo promedio de apertura del subproducto de cuenta de ahorros de los clientes segmentados por sucursal desde el mes de febrero del 2016.

Indicador del saldo promedio de los clientes de sexo femenino que se han mayores a 25 años y por sucursal en el mes de febrero.

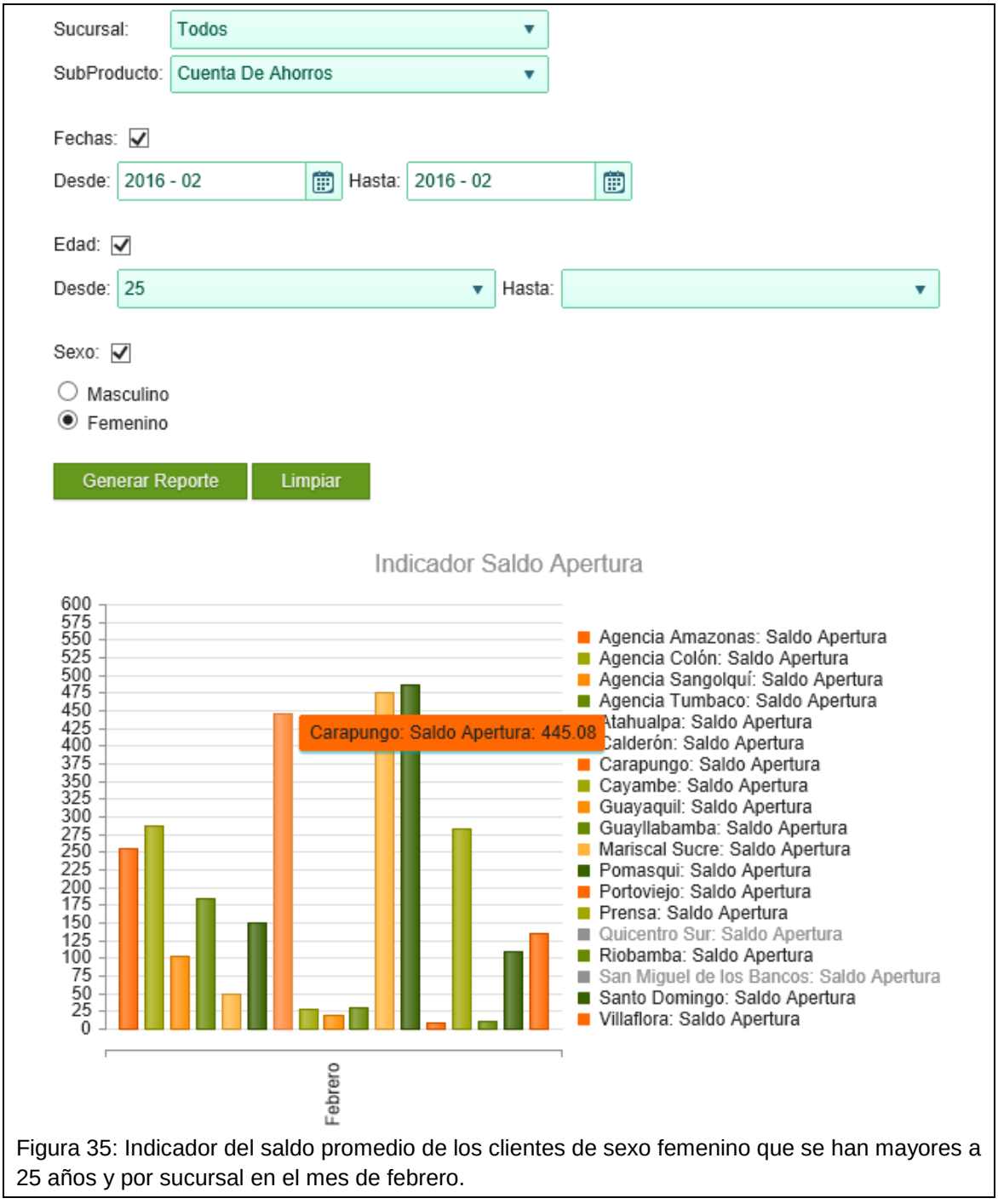


Figura 35: Indicador del saldo promedio de los clientes de sexo femenino que se han mayores a 25 años y por sucursal en el mes de febrero.

2.5.2.4 Daily Scrum Meeting (reuniones diarias)

Un punto importante cuando se emplea scrum son las reuniones diarias, se debe tener en cuenta que son de corta duración. Cada miembro del equipo deberá indicar tres respuestas a las siguientes preguntas:

- ¿Qué tarea hiciste ayer?
- ¿Qué tarea estas haciendo?
- ¿Qué tareas vas hacer más adelante?

Al no contar con otros miembros del equipo más que el tesista para el proyecto no aplica las reuniones diarias.

2.5.2.5 Product Incremente

Representa el producto final luego de terminar el sprint, se presenta en el demo a los usuarios interesados.

2.6 Implementación

En la fase de despliegue se compone de las siguientes tares: implementación, crecimiento y mantenimiento. Nos enfocaremos principalmente en el despliegue de la del ETL y la aplicación

2.6.1 Despliegue ETL

A continuación, se explica los pasos necesarios para la ejecución automatizada del ETL:
a. Se inicia con la creación y configuración de las variables necesarias para el proceso:
Data Warehouse, se especifica el servidor de destino.

Name	Description	Value
Ambiente	Variable utilizada en el envio de correo con ...	Cloud-Produccion
DataWareHouse	Variable de servidor de datawarehouse	CLOUDUIO-BDDCC
RunbookServer	Variable para manejo de los ETLS	CLOUDUIO-PROCP

Figura 36 Creación variable – Servidor de destino

BD-DenariusDataMart, se especifica la base de datos de destino.

Name	Description	Value
BD-DenariusDataMart		DenariusDataMart
BDOrigenMigracion	Base de origen de datos para migración	CooprogresoOds

Figura 37: Creación Variable – Base de datos

b. Se crear un Runnbok, en el artículo publicado por Microsoft Developer Network (2016, párr. 1) En líneas generales, un Runbook es una serie de actividades que utilizan datos, realizan tareas y publican datos para que los utilicen otras actividades del Runbook. Cada Runbook tiene una colección de propiedades configurables. Dichas propiedades permiten personalizar el comportamiento de un Runbook.

Se debe tomar en cuenta el esquema con el que se ha trabajado anteriormente, es decir se selecciona o crea una carpeta a fin de crear el nuevo Runbook, como se muestra en la figura 38:

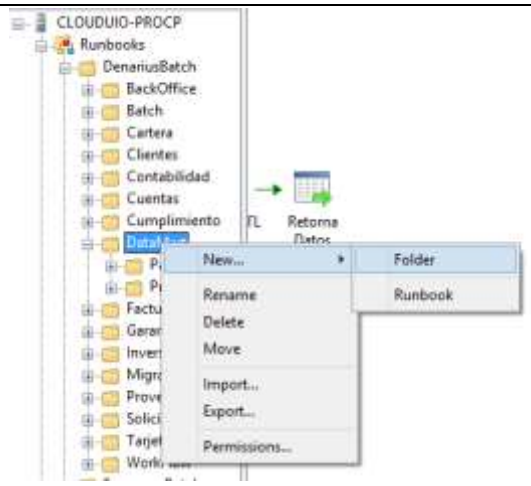


Figura 38: Creación del Runbook

c. A continuación con el empleo de las actividades del controlador de runbook se emplean los elementos necesarios para el proceso, como se muestra en la figura 39:

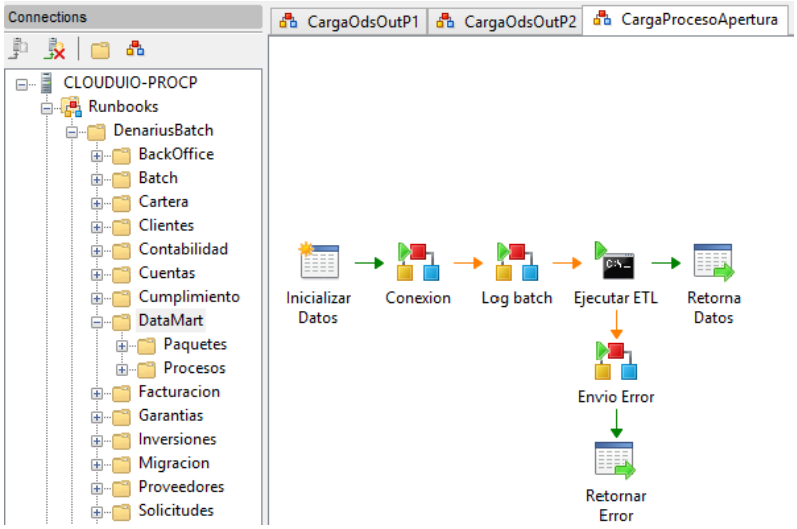


Figura 39: Proceso automatizado de la ejecución del ETL

2.6.2 Despliegue Aplicativo

Con la utilización de las herramientas TFS y Release Management se procederá a realizar el versionamiento desde al ambiente de desarrollo a entorno de pruebas de la siguiente manera:

a. Verificar que todos los cambios de código se encuentren subidos al TFS.

b. A continuación con la utilización de una plantilla de build se procederá a enviar el build con todos los cambios que estén en el TFS, como se muestra en la figura 40.

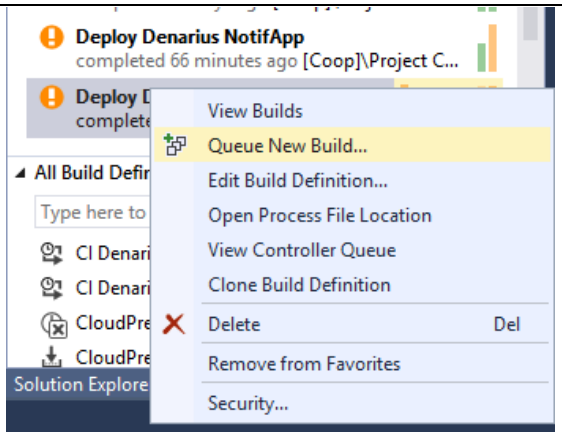


Figura 40: Envío de un nuevo build para el paso de versión.

Al enviar un nuevo build nos mostrará la siguiente figura que permite dar un seguimiento al tiempo que demora en generar la nueva versión de código.

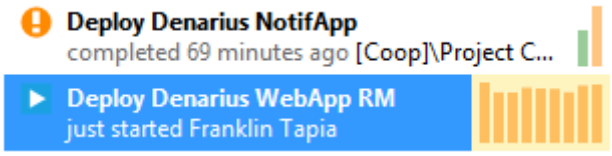


Figura 41: Generación de la nueva versión de código.

c. Luego de terminar el build procedemos a utilizar el Release Management, igualmente se utiliza una plantilla ya creada para el paso del nuevo código, seleccionamos: Paso a Ambiente Cloud Core Financiero Denarius. Se deberá ingresar el nombre al paso de versión, la plantilla y el build generado por el TFS, así como lo muestra la siguiente figura 42:

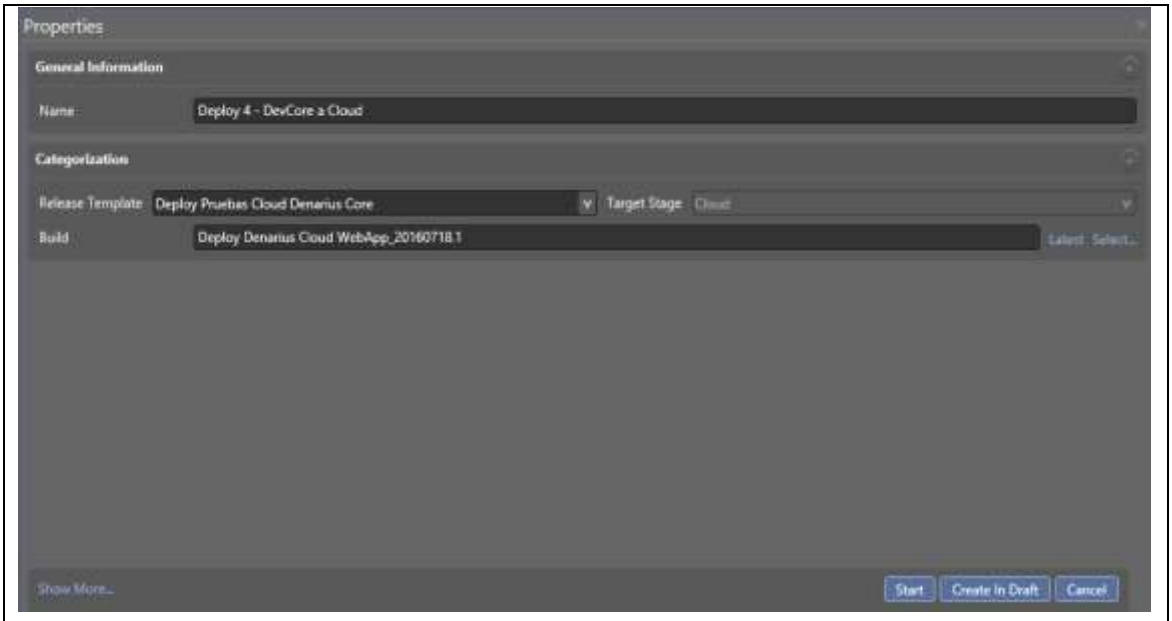


Figura 42: Paso de versión al ambiente de pruebas.

d. Una vez terminado el paso de versión se muestra la siguiente pantalla, observamos los pasos realizados para el paso de versión el tiempo que tardo el responsable de aprobación y el estado de los pasos ejecutados:

Figura 43: Estado del paso de versión.

Con esto se concluye el paso de versión, a continuación, se deberá ingresar al ambiente de pruebas para validar los cambios en el aplicativo.

Luego de concluir con el paso de versión se podrá acceder al sitio web de pruebas en el anexo 10 se muestra los pasos para ingresar.

2.6.3 Mantenimiento

Cuando el sistema está en producción incluye tareas técnico operacionales que son necesarias para mantener el sistema operando óptimamente, algunas incluyen:

- Monitorio del uso
- Tuning del desempeño
- Mantenimiento de la tabla de índices
- Backup del sistema
- Apoyo permanente, capacitación y comunicación con los usuarios finales

2.6.4 Crecimiento

Cuando el sistema está en producción es necesario tener en cuenta el crecimiento que puede llegar a tener, si los resultados son satisfactorios para el cliente a continuación se generan nuevos requisitos lo que lleva a un nuevo ciclo:

- Los DWH tienden a expandirse (si son exitosos es considerado como un signo de éxito)
- Nuevos requerimientos deben ser priorizados
- Empezar el ciclo de nuevo
- Construir sobre las bases ya establecidas
- Enfoque en los nuevos requerimientos

2.7 Gestión

La gestión del proyecto se realiza a lo largo de todo el ciclo de vida. Sus actividades se centran en el monitoreo del estado del proyecto, seguimiento de problemas y control de cambios con el objetivo de mantener el proyecto en curso.

Se encarga de la coordinación y ejecución de las distintas fases que conforman el proyecto de BI, es decir:

- Planificación del proyecto (Project Planin)
- Definición de Requerimientos del negocio (Business Requirements Definition)
- Diseño de la arquitectura técnica (Technical Architecture Design)
- Modelo Dimensional (Dimensional Modeling)

- Diseño de la aplicación de BI (BI Application Design)
- Selección e instalación de productos (Product Selection & Installation)
- Diseño físico (Physical Design)
- Diseño y desarrollo del ETL (ETL Design & Development)
- Desarrollo de la aplicación de BI (BI Application Development)
- Despliegue (Deployment)
- Crecimiento (Growth)
- Mantenimiento (Maintenance)

3 Conclusiones y Recomendaciones

3.1 Conclusiones

La construcción de un módulo para la presentación de los indicadores estratégicos dentro del core de Denariuservis SA, permite generar información visual de manera consolidada, minimizando los costos operacionales al centralizar la información y mejorando los tiempos de respuesta que se emplea en el análisis de la información y la toma de decisiones.

La metodología del ciclo de vida de **Kimball** define aspectos fundamentales y bases sólidas para la creación del módulo de indicadores estratégicos. Con la identificación de la línea tecnológica, datos y aplicación de BI, es importante que la línea de datos no sufra retrasos debido a que esto puede generar que no se cumpla con las fechas programadas del proyecto e impide el avance en la línea de aplicación BI.

La metodología **Scrum** empleada para el desarrollo de la aplicación de BI, podemos notar que con la utilización de sus actividades entre ellas: historias de usuarios, Product Backlog, Sprint Backlog y de más empleadas se pudo identificar claramente las tareas necesarias para desarrollar cada iteración durante la planificación del Sprint Backlog.

Las herramientas empleadas para el desarrollo de la solución BI y la aplicación, se emplea la misma plataforma en la que se desarrolló Denarius Core, debido a que las herramientas Microsoft se encuentran sincronizadas entre si y se cuenta con el respaldo de partners como Microsoft Corporation. Es así que desde visual studio se crean los proyectos de base de datos, el proceso de ETL y el desarrollo de la aplicación (MVC 4 – Framework 4.0) y no se presentan problemas de integración con el funcionamiento del core.

La implementación de la solución a denarius Core, genera cambios importantes para Denariuseris, ya que por medio de la visualización de los indicadores

estratégicos presenta una solución más completa hacia los clientes que estén interesados en la contratación de sus servicios financieros.

Es importancia tomar en consideración que las soluciones BI ayudan con el soporte en la toma de decisiones, brindando datos de calidad sobre los procesos de gestión de cuentas y el saldo promedio de apertura, a pesar de lo expuesto las personas encargadas de tomar las decisiones son el factor más importante, debido a que en sus manos están conseguir una ventaja competitiva de las decisiones que se tomen.

Para el desarrollo de un proyecto de una solución BI se deberá tener presente conceptos claros y bien definidos de bases de datos, que si bien es cierto que en la actualidad existen herramientas que nos facilitan el desarrollo del proyecto debe existir una persona que gestione la utilización de las herramientas a ser utilizadas.

Se deberá tomar en cuenta los elementos de origen de datos y su procedencia. Esto debido a que en la metodología de Kimball no presenta de manera evidente como se encuentran estructurados los datos de origen, el número de registros en las principales tablas, como crecen sus registros en el tiempo.

Los indicadores descritos en el proyecto están enfocados hacia temas operacionales en las entidades financieras.

3.2 Recomendaciones

En la entrega de requerimientos por parte del usuario se debe plantear como una estrategia de calidad, que el usuario tenga claro el objetivo que pretende alcanzar sean estos estratégicos, operativos o tácticos, y se identifique todos los tipos de usuarios finales a los que va dirigida la solución.

La persona encargada de recibir e indicar la factibilidad de generar los indicadores debe ser altamente calificada con respecto al entorno en el que se desenvuelve la organización y conocer los procesos internos a fin de dar la mejor evaluación sobre el proceso seleccionado.

Se debe tomar en cuenta la factibilidad de cada una de las herramientas con las que planifica realizar el proyecto, debido a que pueden surgir imprevistos durante el desarrollo que limitan el producto final a entregar.

Es indispensable que luego de la entrega de los indicadores a los usuarios finales se brinde un constante monitoreo y mantenimiento a fin de garantizar resultados eficientes y que no decaigan con el tiempo, como parte de la gestión del ciclo de vida de Kimball.

Es necesario que Denariuservis SA contrate una herramienta OLAP que permita explotar todo el potencial de los modelos multidimensionales a fin de generar un valor mucho más alto.

El enfoque para la presentación de los indicadores debe ser acorde a los usuarios a los que van dirigidos, en el presente proyecto el indicador de gestión se pude cambiar al porcentaje creciente o decreciente de nuevos clientes ingresados.

4. Referencias

Domínguez, J. (s.f.). Principios Básicos de Seguridad de Base de Datos. Recuperado el 15 de abril de 2016 de http://www.academia.edu/13928757/Principios_B%C3%A1sicos_de_Seguridad_en_Bases_de_Datos

Kimbal, R., & Caserta, J. (2004). The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Indianapolis, USA: John Wiley & Sons.

Kimball, R., & Ross, M. (2002). The data warehouse toolkit: the complete guide to dimensional modeling. Canada: Wiley Computer Publishing.

Latino BI. (s.f.). Latino BI Inteligencia de Negocios + Soluciones Estratégicas. Recuperado el 7 de julio de 2016 de http://www.latino-bi.com/espanol/fundamentos-bi/concepto-dimensiones-metricas.php

Microsoft Developer Network. (s.f.). MSDN Library. Requisitos de sistema de Release Management. Recuperado el 21 de julio de 2016 de https://msdn.microsoft.com/es-es/library/dn593703.aspx

Microsoft System Center. (s.f.). Requisitos de Runbook Designer de Orchestrator en System Center 2012 SP1. Recuperado el 21 de julio de 2016 de https://technet.microsoft.com/es-es/library/hh420351(v=sc.12).aspx.

PMOinformatica. (s.f.). PMOinformatica. La oficina de proyectos de informática. Recuperado el 12 de mayo de 2016 de http://www.pmoinformatica.com/2013/11/plantillas-scrum-pila-producto-product.html

Ross, M. (s.f.). kimball Group. Recuperado el 30 de junio de 2016 de http://www.kimballgroup.com/2009/08/design-tip-115-kimball-lifecycle-in-a-nutshell/

Visual Studio (s.f.). Requisitos del sistema para Visual Studio. Recuperado el 10 de mayo de 2016 de https://www.visualstudio.com/es-es/downloads/visual-studio-2015-system-requirements-vs.aspx#27.

Sutherland, J. (s.f.). sruminc. Recuperado el 21 de julio de 2016 de http://scrum.jeffsutherland.com/

TechNet Microsoft. (s.f.). Requisitos de hardware y software para instalar SQL Server 2012. Recuperado el 16 de julio de 2016 de https://technet.microsoft.com/es-es/library/ms143506(v=sql.110).aspx

Telerik Demos. (s.f.). Telerik UI for ASP.NET MVC. Bar Charts / Basic usage Recuperado el 21 de julio de 2016 de http://demos.telerik.com/aspnet-mvc/bar-charts/index

Visual Studio. (s.f.). Visual Studio. Release Management Recuperado el 21 de julio de 2016 de https://www.visualstudio.com/es-es/dn469491.aspx

Visual Studio. (s.f.). Team Foundation Server. Recuperado el 14 de mayo de 2016 de https://www.visualstudio.com/es-es/products/tfs-overview-vs.aspx

Visual Studio. (s.f.). Visual Studio. Requirements and compatibility. Recuperado el 26 de mayo de 2016 de https://www.visualstudio.com/en-us/docs/setup-admin/requirements

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M. Sutherland, J. (s.f.). Agile Manifesto. Recuperado el 13 de mayo de 2016 de http://agilemanifesto.org/iso/es/

Cohn, M. (s.f.). Mountain Goat Software. Recuperado el 4 de julio de 2016 de http://www.mountaingoatsoftware.com/topics/scrum

ANEXOS

Anexo 1: Tablas filtros de consulta, subproducto y sucursal

Subproducto

Id	Descripción	Nemónico
1	CUENTA DE AHORROS	PB1
2	CERTIFICADOS DE APORTACION	PB7
3	AHORROS PROGRAMADOS	PB6
4	CUENTA PROGRESAR CLIENTES	PB0
5	CUENTA AHORRO PROGRESAR YO SOY	PB2
6	CUENTA CORPORATIVA PROGRESAR	PB3
11	CUENTA DE GERENCIA	CCG
7	CERTIFICADOS	PB4
8	AHORRO ESTUDIANTIL	PB5
9	CUENTA PROGRESAR ORO	PB8
10	CUENTA PROGRESAR PLATINUM	PB9

Sucursal

IdSucursal	Nombre
1	Pomasqui
3	Atahualpa
5	Calderón
7	Villaflora
9	San Miguel de los Bancos
11	Carapungo
13	Guayllabamba
15	Agencia Amazonas
17	Agencia Colón
19	Mariscal Sucre
21	Agencia Tumbaco
23	Agencia Sangolquí
25	Cayambe
27	Prensa
29	Santo Domingo
31	Quicentro Sur
33	Guayaquil
35	Portoviejo
37	Centro de Inversiones
39	Matriz
41	Riobamba
43	Tena

Anexo 2: Diccionario de datos del proceso de ingreso de clientes, apertura de cuentas y saldos promedios

DimCliente

Tabla	Número	Columna	Tipo Dato	Tamaño	Nulos	Clave Primaria	Clave Foránea
DimCliente	1	IdCliente	Int	4	N	Y	N
DimCliente	2	TipoCliente	Varchar	10	N	N	N
DimCliente	3	NombreCompleto	Varchar	300	N	N	N
DimCliente	4	ActividadEconomica	Varchar	150	N	N	N
DimCliente	5	Nacionalidad	Varchar	50	Y	N	N
DimCliente	6	Estado	Varchar	15	Y	N	N
DimCliente	7	Sexo	Varchar	10	Y	N	N
DimCliente	8	FechaNacimiento	Date	8	Y	N	N
DimCliente	9	FechaModificacion	Date	8	Y	N	N

DimCuenta

Tabla	Número	Columna	Tipo Dato	Tamaño	Nulos	Clave Primaria	Clave Foranea
DimCuenta	1	IdCuenta	Int	4	N	Y	N
DimCuenta	2	Nombre	Varchar	30	N	N	N
DimCuenta	3	Nemonico	Varchar	10	N	N	N

DimEstadoCuenta

Tabla	Número	Columna	Tipo Dato	Tamaño	Nulos	Clave Primaria	Clave Foranea
DimEstadoCuenta	1	IdCuenta	Int	4	N	Y	N
DimEstadoCuenta	2	Sigla	Chart	1	N	N	N
DimEstadoCuenta	3	Nombre	Varchar	10	N	N	N

DimTiempo

Tabla	Número	Columna	Tipo Dato	Tamaño	Nulos	Clave Primaria	Clave Foranea
DimFecha	1	IdFecha	Int	4	N	Y	N
DimFecha	2	Fecha	Date	8	N	N	N
DimFecha	3	Anio	SmallInt	2	N	N	N
DimFecha	4	Mes	Tinyint	1	N	N	N
DimFecha	5	Dia	Tinyint	1	N	N	N
DimFecha	6	Trimestre	Tinyint	1	N	N	N

DimSucursales

Tabla	Número	Columna	Tipo Dato	Tamaño	Nulos	Clave Primaria	Clave Foranea
DimSucursales	1	IdFecha	Int	4	N	Y	N
DimSucursales	2	Provincia	Varchar	100	N	N	N
DimSucursales	3	Ciudad	Varchar	100	N	N	N
DimSucursales	4	Sucursal	Varchar	100	N	N	N

Tabla de hechos

Tabla	Número	Columna	Tipo Dato	Tamaño	Nulos	Clave Primaria	Clave Foranea
HechosCuentas	1	IdCliente	Int	4	N	Y	Y
HechosCuentas	2	IdCuenta	Int	4	N	Y	Y
HechosCuentas	3	IdSucursal	Int	4	N	Y	Y
HechosCuentas	4	IdFecha	Int	4	N	Y	Y
HechosCuentas	5	IdEstadoCuenta	Int	4	N	N	N
HechosCuentas	6	IdNumeroCuenta	Varchar	18	N	N	N
HechosCuentas	7	GestionCuentas	Tinyint	4	Y	N	N
HechosCuentas	8	SaldoApertura	Money	8	Y	N	N

Anexo 3: Scripts carga de dimensiones

DimSucursal

```
SELECT
       de_id_division_empresarial
       ,de_nombre AS 'NombreSucursal'
       ,dc1.dc_descripcion AS 'Ciudad'
       ,dc2.dc_descripcion AS 'Provincia'
FROM   dbo.DivisionEmpresarial de
JOIN   dbo.DetalleCatalogo dc1 ON de.de_ciudad = dc1.dc_id_detalle_catalogo
JOIN   dbo.DetalleCatalogo dc2 ON dc1.dc_codigo_padre =
dc2.dc_id_detalle_catalogo
WHERE  de_tipo = 70005 --Tipo_Sucursal
```

DimFecha

```
SET LANGUAGE 'SPANISH'

SELECT DISTINCT
       CONVERT(DATE, ac.ac_fecha_apertura) AS 'FechaFull'
       ,CONVERT(SMALLINT, (DATENAME(YEAR, ac.ac_fecha_apertura))) AS 'Anio'
       ,CONVERT(TINYINT, (DATEPART(MONTH, ac.ac_fecha_apertura))) AS 'Mes'
       ,CONVERT(TINYINT, (DATEPART(DAY, ac.ac_fecha_apertura))) AS 'Dia'
```

```
	,CONVERT(TINYINT, (DATEPART(QUARTER, ac.ac_fecha_apertura))) AS
'Trimestre'
	,CONVERT(VARCHAR, (DATENAME(MONTH, ac.ac_fecha_apertura))) AS 'NombreMes'
FROM	cuentas.AnalisisCuenta ac
WHERE	CONVERT (DATE, ac.ac_fecha_apertura) >= '2016-01-01'
ORDER BY CONVERT (DATE, ac.ac_fecha_apertura)
```

DimEstadoCuenta

```
SELECT DISTINCT
		ca_estado AS 'Sigla'
		,CASE ca_estado
			WHEN 'A' THEN 'Activa'
			WHEN 'C' THEN 'Cerrada'
			WHEN 'I' THEN 'Inactiva'
			WHEN 'L' THEN 'Liquidada'
		END AS 'Nombre'
FROM	cuentas.CuentaAhorros
ORDER BY ca_estado
```

Carga dimisión DimCuentas

```
SELECT
	su_id_subproducto AS 'IdCuenta'
	,su_descripcion AS 'Nombre'
	,su_nemonico AS 'Nemonico'
FROM	cuentas.Subproducto
WHERE	su_id_producto = 1 --CuentasAhorros
AND	su_estado = 'V' --Cuentas Vigentes
```

Carga dimisión DimCliente

```
SELECT
	cl.cl_id_cliente
	,CASE cl_tipo_persona
		WHEN 1 THEN 'Jurídico'
		WHEN 0 THEN 'Natural'
	END as 'TipoPersona'
	,cl_nombre_completo
	,dc1.dc_descripcion AS 'ActividadEconomica'
	,dc2.dc_descripcion AS 'Nacionalidad'
	,dc3.dc_descripcion AS 'Estado'
	,CASE cn_sexo
		WHEN 1 THEN 'Masculino'
		WHEN 0 THEN 'Femenino'
	END AS 'Genero'
	,CONVERT (DATE, cn_fecha_nacimiento) AS 'FechaNacimiento'
	,CONVERT (DATE, cl_fecha_modificacion) as 'FechaModificacion'
FROM clientes.Cliente cl
JOIN clientes.ClienteNatural cn ON cl.cl_id_cliente = cn.cn_id_cliente_natural
JOIN cuentas.Cuenta cu ON cl.cl_id_cliente = cu.cu_id_cliente
JOIN dbo.DetalleCatalogo dc1 ON cl.cl_actividad = dc1.dc_id_detalle_catalogo
JOIN dbo.DetalleCatalogo dc2 ON cl.cl_nacionalidad = dc2.dc_id_detalle_catalogo
JOIN dbo.DetalleCatalogo dc3 ON cl.cl_estado = dc3.dc_id_detalle_catalogo
WHERE cl_fecha_registro >= '2016-01-01'
UNION
SELECT
```

```
       cl.cl_id_cliente
       ,CASE cl_tipo_persona
              WHEN 1 THEN 'Jurídico'
              WHEN 0 THEN 'Natural'
       END AS 'TipoPersona'
       ,cl_nombre_completo
       ,dc1.dc_descripcion AS 'ActividadEconomica'
       ,dc2.dc_descripcion AS 'Nacionalidad'
       ,dc3.dc_descripcion AS 'Estado'
       ,'NA' AS 'Sexo'
       ,NULL AS 'FechaNacimiento'
       ,CONVERT (DATE, cl_fecha_modificacion) as 'FechaModificacion'
FROM clientes.Cliente cl
JOIN clientes.ClienteJuridico cj ON cl.cl_id_cliente = cj.cj_id_cliente_juridico
JOIN cuentas.Cuenta cu ON cl.cl_id_cliente = cu.cu_id_cliente
JOIN dbo.DetalleCatalogo dc1 ON cl.cl_actividad = dc1.dc_id_detalle_catalogo
JOIN dbo.DetalleCatalogo dc2 ON cl.cl_nacionalidad = dc2.dc_id_detalle_catalogo
JOIN dbo.DetalleCatalogo dc3 ON cl.cl_estado = dc3.dc_id_detalle_catalogo
WHERE cl_fecha_registro >= '2016-01-01'
```

Carga tabla de hechos

```
SELECT
       cl.cl_id_cliente
       ,su.su_id_subproducto 'IdCuenta'
       ,de.de_id_division_empresarial AS 'IdSucursal'
       ,die.IdEstadoCuenta
       ,df.IdFecha
       ,RTRIM(cu.cu_numero) AS 'IdNumeroCuenta'
       ,1 as 'GestionCuentas'
       ,ac_saldo1_disponible AS 'SaldoApertura'
FROM cuentas.Subproducto su
JOIN cuentas.Cuenta cu ON su.su_id_subproducto = cu.cu_id_subproducto
JOIN cuentas.CuentaAhorros ca ON cu.cu_id_cuenta = ca.ca_id_cuenta
JOIN datamart.DimEstadoCuenta die ON ca.ca_estado = die.Sigla
JOIN cuentas.AnalisisCuenta ac ON cu.cu_id_cuenta = ac.ac_id_cuenta
JOIN datamart.DimFecha df ON CONVERT(DATE, ac.ac_fecha_apertura) = df.Fecha
JOIN clientes.Cliente cl ON cu.cu_id_cliente = cl.cl_id_cliente
JOIN dbo.DivisionEmpresarial de ON cu_id_oficina_oficial =
de.de_id_division_empresarial
WHERE cl_fecha_registro >= '2016-01-01'
ORDER BY cl_id_cliente
```

Anexo 4: Creación de procesos almacenados para la historia H1 (A5)

Consulta Clientes Activos

```
CREATE PROCEDURE [datamart].[ConsultaClientesActivos]
(
       @i_estado                          VARCHAR(15) = NULL,
       @i_sucursal                        VARCHAR(100) = NULL,
       @i_ciudad                          VARCHAR(100) = NULL,
       @i_fecha_desde                     DATE = NULL,
       @i_fecha_hasta                     DATE = NULL
)
```

```
AS

DECLARE @w_anio_inicio      SMALLINT,
              @w_anio_fin          SMALLINT,
              @w_mes_inicio TINYINT,
              @w_mes_final  TINYINT

SET @w_anio_inicio = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_desde)))
SET @w_anio_fin = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_hasta)))
SET @w_mes_inicio = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_desde)))
SET @w_mes_final = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_hasta)))

IF (@i_ciudad IS NOT NULL)
BEGIN

       IF(@i_ciudad = 'Todos')
              SET @i_ciudad = NULL

       SELECT
                     df.Anio
                     ,'NombreMes' =
                            CASE df.Mes
                                   WHEN 1 THEN 'Enero'
                                   WHEN 2 THEN 'Febrero'
                                   WHEN 3 THEN 'Marzo'
                                   WHEN 4 THEN 'Abril'
                                   WHEN 5 THEN 'Mayo'
                                   WHEN 6 THEN 'Junio'
                                   WHEN 7 THEN 'Julio'
                                   WHEN 8 THEN 'Agosto'
                                   WHEN 9 THEN 'Septiembre'
                                   WHEN 10 THEN 'Octubre'
                                   WHEN 11 THEN 'Nomviembre'
                                   WHEN 12 THEN 'Diciembre'
                            END
                     ,ds.Provincia
                     ,ds.Ciudad
                     ,'' AS 'Sucursal'
                     ,count(*) AS 'NumeroClientesActivos'
                     ,@i_estado AS 'EstadoRpt'
       FROM   datamart.HechosAperturarClientesCuentas h
       JOIN   datamart.DimCliente dc ON h.IdCliente = dc.IdCliente
       JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
       JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
       WHERE  ds.Ciudad = ISNULL(@i_ciudad, ds.Ciudad)
       AND           df.Anio BETWEEN ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
       AND           df.Mes BETWEEN ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)
       AND           dc.FechaModificacion IS NOT NULL
       GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad
END
ELSE
BEGIN

       IF(@i_sucursal = 'Todos')
              SET @i_sucursal = NULL

       SELECT
                     df.Anio
```

```
                    ,'NombreMes' =
                           CASE df.Mes
                                  WHEN 1 THEN 'Enero'
                                  WHEN 2 THEN 'Febrero'
                                  WHEN 3 THEN 'Marzo'
                                  WHEN 4 THEN 'Abril'
                                  WHEN 5 THEN 'Mayo'
                                  WHEN 6 THEN 'Junio'
                                  WHEN 7 THEN 'Julio'
                                  WHEN 8 THEN 'Agosto'
                                  WHEN 9 THEN 'Septiembre'
                                  WHEN 10 THEN 'Octubre'
                                  WHEN 11 THEN 'Nomviembre'
                                  WHEN 12 THEN 'Diciembre'
                           END
                    ,ds.Provincia
                    ,ds.Ciudad
                    ,ds.Sucursal
                    ,count(*) AS 'NumeroClientesActivos'
                    ,@i_estado AS 'EstadoRpt'
      FROM   datamart.HechosAperturarClientesCuentas h
      JOIN   datamart.DimCliente dc ON h.IdCliente = dc.IdCliente
      JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
      JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
      WHERE  ds.Sucursal = ISNULL(@i_sucursal, ds.Sucursal)
      AND           df.Anio BETWEEN ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
      AND           df.Mes BETWEEN ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)
      AND           dc.FechaModificacion IS NOT NULL
      GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad, ds.Sucursal
END

RETURN 0
```

Consulta Clientes Ingresados

```
CREATE PROCEDURE [datamart].[ConsultaClientesIngresados]
(
      @i_estado                         VARCHAR(15) = NULL,
      @i_sucursal                       VARCHAR(100) = NULL,
      @i_ciudad                         VARCHAR(100) = NULL,
      @i_fecha_desde                    DATE = NULL,
      @i_fecha_hasta                    DATE = NULL
)
AS

DECLARE @w_anio_inicio      SMALLINT,
              @w_anio_fin         SMALLINT,
              @w_mes_inicio TINYINT,
              @w_mes_final  TINYINT

SET @w_anio_inicio = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_desde)))
SET @w_anio_fin = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_hasta)))
SET @w_mes_inicio = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_desde)))
SET @w_mes_final = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_hasta)))

IF (@i_ciudad IS NOT NULL)
```

```
BEGIN

	IF(@i_ciudad = 'Todos')
		SET @i_ciudad = NULL

	SELECT
			df.Anio
			,'NombreMes' =
				CASE df.Mes
					WHEN 1 THEN 'Enero'
					WHEN 2 THEN 'Febrero'
					WHEN 3 THEN 'Marzo'
					WHEN 4 THEN 'Abril'
					WHEN 5 THEN 'Mayo'
					WHEN 6 THEN 'Junio'
					WHEN 7 THEN 'Julio'
					WHEN 8 THEN 'Agosto'
					WHEN 9 THEN 'Septiembre'
					WHEN 10 THEN 'Octubre'
					WHEN 11 THEN 'Nomviembre'
					WHEN 12 THEN 'Diciembre'
				END
			,ds.Provincia
			,ds.Ciudad
			,'' AS 'Sucursal'
			,count(*) AS 'Aperturados'
			,@i_estado AS 'EstadoRpt'
	FROM	datamart.HechosAperturarClientesCuentas h
	JOIN	datamart.DimCliente dc ON h.IdCliente = dc.IdCliente
	JOIN	datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
	JOIN	datamart.DimFecha df ON h.IdFecha = df.IdFecha
	WHERE	ds.Ciudad = ISNULL(@i_ciudad, ds.Ciudad)
	AND		df.Anio BETWEEN ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
	AND		df.Mes BETWEEN ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)
	GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad
END
ELSE
BEGIN

	IF (@i_sucursal = 'Todos')
		SET @i_sucursal = NULL

	SELECT
			df.Anio
			,'NombreMes' =
				CASE df.Mes
					WHEN 1 THEN 'Enero'
					WHEN 2 THEN 'Febrero'
					WHEN 3 THEN 'Marzo'
					WHEN 4 THEN 'Abril'
					WHEN 5 THEN 'Mayo'
					WHEN 6 THEN 'Junio'
					WHEN 7 THEN 'Julio'
					WHEN 8 THEN 'Agosto'
					WHEN 9 THEN 'Septiembre'
					WHEN 10 THEN 'Octubre'
					WHEN 11 THEN 'Nomviembre'
					WHEN 12 THEN 'Diciembre'
```

```
                        END
                    ,ds.Provincia
                    ,ds.Ciudad
                    ,ds.Sucursal
                    ,count(*) AS 'Aperturados'
                    ,@i_estado AS 'EstadoRpt'
      FROM   datamart.HechosAperturarClientesCuentas h
      JOIN   datamart.DimCliente dc ON h.IdCliente = dc.IdCliente
      JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
      JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
      WHERE  ds.Sucursal = ISNULL(@i_sucursal, ds.Sucursal)
      AND          df.Anio BETWEEN ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
      AND          df.Mes BETWEEN ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)
      GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad, ds.Sucursal
END

RETURN 0
```

Consulta Clientes Pasivos

```
CREATE PROCEDURE [datamart].[ConsultaClientesPasivos]
(
      @i_estado                        VARCHAR(15) = NULL,
      @i_sucursal                      VARCHAR(100) = NULL,
      @i_ciudad                        VARCHAR(100) = NULL,
      @i_fecha_desde                   DATE = NULL,
      @i_fecha_hasta                   DATE = NULL
)
AS

DECLARE @w_anio_inicio     SMALLINT,
            @w_anio_fin          SMALLINT,
            @w_mes_inicio TINYINT,
            @w_mes_final  TINYINT

SET @w_anio_inicio = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_desde)))
SET @w_anio_fin = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_hasta)))
SET @w_mes_inicio = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_desde)))
SET @w_mes_final = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_hasta)))

IF (@i_ciudad IS NOT NULL)
BEGIN

      IF(@i_ciudad = 'Todos')
            SET @i_ciudad = NULL

      SELECT
                  df.Anio
                  ,'NombreMes' =
                        CASE df.Mes
                              WHEN 1 THEN 'Enero'
                              WHEN 2 THEN 'Febrero'
                              WHEN 3 THEN 'Marzo'
                              WHEN 4 THEN 'Abril'
                              WHEN 5 THEN 'Mayo'
                              WHEN 6 THEN 'Junio'
                              WHEN 7 THEN 'Julio'
```

```
                                WHEN 8 THEN 'Agosto'
                                WHEN 9 THEN 'Septiembre'
                                WHEN 10 THEN 'Octubre'
                                WHEN 11 THEN 'Nomviembre'
                                WHEN 12 THEN 'Diciembre'
                        END
                ,ds.Provincia
                ,ds.Ciudad
                ,'' AS 'Sucursal'
                ,count(*) AS 'NumeroClientesPasivos'
                ,@i_estado AS 'EstadoRpt'
        FROM   datamart.HechosAperturarClientesCuentas h
        JOIN   datamart.DimCliente dc ON h.IdCliente = dc.IdCliente
        JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
        JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
        WHERE  ds.Ciudad = ISNULL(@i_ciudad, ds.Ciudad)
        AND           df.Anio BETWEEN ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
        AND           df.Mes BETWEEN ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)
        AND           dc.FechaModificacion IS NULL
        GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad
END
ELSE
BEGIN

        IF (@i_sucursal = 'Todos')
                SET @i_sucursal = NULL

        SELECT
                df.Anio
                ,'NombreMes' =
                        CASE df.Mes
                                WHEN 1 THEN 'Enero'
                                WHEN 2 THEN 'Febrero'
                                WHEN 3 THEN 'Marzo'
                                WHEN 4 THEN 'Abril'
                                WHEN 5 THEN 'Mayo'
                                WHEN 6 THEN 'Junio'
                                WHEN 7 THEN 'Julio'
                                WHEN 8 THEN 'Agosto'
                                WHEN 9 THEN 'Septiembre'
                                WHEN 10 THEN 'Octubre'
                                WHEN 11 THEN 'Nomviembre'
                                WHEN 12 THEN 'Diciembre'
                        END
                ,ds.Provincia
                ,ds.Ciudad
                ,ds.Sucursal
                ,count(*) AS 'NumeroClientesPasivos'
                ,@i_estado AS 'EstadoRpt'
        FROM   datamart.HechosAperturarClientesCuentas h
        JOIN   datamart.DimCliente dc ON h.IdCliente = dc.IdCliente
        JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
        JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
        WHERE  ds.Sucursal = ISNULL(@i_sucursal, ds.Sucursal)
        AND           df.Anio BETWEEN ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
        AND           df.Mes BETWEEN ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)
```

```
	AND		dc.FechaModificacion IS NULL
	GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad, ds.Sucursal
END

RETURN 0
```

Consulta Clientes Todos los estados

```
CREATE PROCEDURE [datamart].[ConsultaClientesTodos]
(
	@i_estado                                               VARCHAR(15) = NULL,
	@i_sucursal                                          VARCHAR(100) = NULL,
	@i_ciudad                                               VARCHAR(100) =
NULL,
	@i_tipo_subproducto		VARCHAR(30) = NULL,
	@i_fecha_desde                                  DATE = NULL,
	@i_fecha_hASta                                  DATE = NULL
)
AS

DECLARE @w_anio_inicio	SMALLINT,
			@w_anio_fin		SMALLINT,
			@w_mes_inicio TINYINT,
			@w_mes_final  TINYINT

SET @w_anio_inicio   = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_desde)))
SET @w_anio_fin                  = CONVERT(SMALLINT, (DATENAME(YEAR,
@i_fecha_hASta)))
SET @w_mes_inicio		= CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_desde)))
SET @w_mes_final		= CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_hASta)))

DECLARE @w_tmp_clientes TABLE
(	Anio
	INT,
	NombreMes
	VARCHAR(20),
	Provincia
	VARCHAR(40),
	Ciudad
	VARCHAR(40),
	Sucursal
	VARCHAR(40),
	Aperturados                                                     INT,
	NumeroClientesActivos                   INT,
	NumeroClientesPASivos                   INT,
	NumeroCuentASAperturadAS	INT,
	NumeroCuentASActivAS		INT,
	NumeroCuentASCerradAS                   INT,
	TipoSubproducto                                         VARCHAR(40),
	EstadoRpt
	VARCHAR(20),
	FechaInicio                                                     DATE,
	FechaFin                                                                DATE,
	FechaModificacion                               DATE
)

SET NOCOUNT ON
```

```
IF     (@i_estado = 'Todos')
       SET @i_estado = NULL

IF (@i_sucursal IS NOT NULL)
BEGIN

       IF     (@i_sucursal = 'Todos')
       SET @i_sucursal = NULL

              INSERT INTO @w_tmp_clientes
              SELECT
                     df.Anio
                                          AS 'Anio'
                     ,df.NombreMes
                     AS 'NombreMes'
                     ,ds.Provincia
                     AS 'Provincia'
                     ,ds.Ciudad
                            AS 'Ciudad'
                     ,ds.Sucursal
                     AS 'Sucursal'
                     ,0
                                                 AS 'Aperturados'
                     ,0
                                                 AS 'NumeroClientesActivos'
                     ,0
                                                 AS 'NumeroClientesPASivos'
                     ,0
                                                 AS 'NumeroCuentASAperturadAS'
                     ,0
                                                 AS 'NumeroCuentASActivAS'
                     ,0
                                                 AS 'NumeroCuentASCerradAS'
                     ,''
                                                 AS 'TipoSubproducto'
                     ,@i_estado
                            AS 'EstadoRpt'
                     ,convert(DATE, @i_fecha_desde)     AS 'FechaInicio'
                     ,convert(DATE, @i_fecha_hASta)     AS 'FechaFin'
                     ,dc.FechaModificacion AS 'FechaModificacion'
              FROM datamart.HechosAperturarClientesCuentAS h
              JOIN datamart.DimCliente dc ON h.IdCliente = dc.IdCliente
              JOIN datamart.DimFecha df ON h.IdFecha = df.IdFecha
              JOIN datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
              WHERE ds.Sucursal = ISNULL(@i_sucursal, ds.Sucursal)
              AND df.Anio between ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
              AND df.Mes between ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)

END

DECLARE @w_clientesactivos int,
              @w_clientespASivos int

              --SELECT * FROM @w_tmp_clientes
--Aperturados
SELECT totalaperturados = count(*),
          Sucursal AS 'SucursalAperturados',
               NombreMes AS 'NombreMesAperturados'
```

```
INTO #tmp_clientes_aperturados
FROM @w_tmp_clientes
GROUP BY Sucursal, NombreMes

UPDATE @w_tmp_clientes
SET Aperturados =  totalaperturados
FROM #tmp_clientes_aperturados
WHERE SucursalAperturados = Sucursal
  AND NombreMesAperturados = NombreMes

--Activos
SELECT totalactivos = count(*),
          Sucursal AS 'SucursalActivos',
               NombreMes AS 'NombreMesActivos'
INTO #tmp_clientes_activos
FROM @w_tmp_clientes
WHERE FechaModificacion IS NOT NULL
GROUP BY Sucursal, NombreMes

UPDATE @w_tmp_clientes
SET NumeroClientesActivos =  totalactivos
FROM #tmp_clientes_activos
WHERE SucursalActivos = Sucursal
AND NombreMesActivos = NombreMes

--PASivos
SELECT totalpASivos = count(*),
          Sucursal AS 'SucursalPASivos',
               NombreMes AS 'NombreMesPASivos'
INTO #tmp_clientes_pASivos
FROM @w_tmp_clientes
WHERE FechaModificacion IS NULL
GROUP BY Sucursal, NombreMes

UPDATE @w_tmp_clientes
SET NumeroClientesPASivos =  totalpASivos
FROM #tmp_clientes_pASivos
WHERE SucursalPASivos = Sucursal
AND NombreMesPASivos = NombreMes

--Select Final
SELECT Anio,  NombreMes,    Provincia,    Ciudad,       Sucursal,     Aperturados,
       NumeroClientesActivos,      NumeroClientesPASivos
FROM @w_tmp_clientes
GROUP BY Anio,       NombreMes,    Provincia,    Ciudad,       Sucursal,
       Aperturados,  NumeroClientesActivos,      NumeroClientesPASivos


RETURN 0
```

Consulta Cuentas Aperturadas

```
CREATE PROCEDURE [datamart].[ConsultaCuentasAperturadas]
(
       @i_sucursal                          VARCHAR(100) = NULL,
       @i_estado                            VARCHAR(15) = NULL,
       @i_tipo_subproducto          VARCHAR(30) = NULL,
       @i_fecha_desde                       DATE = NULL,
```

```
	@i_fecha_hasta                    DATE = NULL,
	@i_edad_desde              SMALLINT = NULL,
	@i_edad_hasta              SMALLINT = NULL,
	@i_sexo                                VARCHAR(15) = NULL
)
AS

DECLARE @w_anio_inicio      SMALLINT,
              @w_anio_fin          SMALLINT,
              @w_mes_inicio TINYINT,
              @w_mes_final  TINYINT

SET @w_anio_inicio = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_desde)))
SET @w_anio_fin = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_hasta)))
SET @w_mes_inicio = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_desde)))
SET @w_mes_final = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_hasta)))

IF (@i_sucursal = 'Todos')
	SET @i_sucursal  = NULL

IF (@i_tipo_subproducto='Todos')
	SET @i_tipo_subproducto = NULL

SELECT
              df.Anio
              ,'NombreMes' =
                     CASE df.Mes
                            WHEN 1 THEN 'Enero'
                            WHEN 2 THEN 'Febrero'
                            WHEN 3 THEN 'Marzo'
                            WHEN 4 THEN 'Abril'
                            WHEN 5 THEN 'Mayo'
                            WHEN 6 THEN 'Junio'
                            WHEN 7 THEN 'Julio'
                            WHEN 8 THEN 'Agosto'
                            WHEN 9 THEN 'Septiembre'
                            WHEN 10 THEN 'Octubre'
                            WHEN 11 THEN 'Nomviembre'
                            WHEN 12 THEN 'Diciembre'
                     END
              ,ds.Provincia
              ,ds.Ciudad
              ,ds.Sucursal
              ,dc.Nombre AS 'TipoSubproducto'
              ,count(*) AS 'NumeroCuentasAperturadas'
              ,@i_estado AS 'EstadoRpt'
FROM   datamart.HechosAperturarClientesCuentas h
JOIN   datamart.DimCliente dcl ON h.IdCliente = dcl.IdCliente
JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
JOIN   datamart.DimCuenta dc ON h.IdCuenta = dc.IdCuenta
JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
where  ds.Sucursal = ISNULL(@i_sucursal, ds.Sucursal)
AND           dc.Nombre = ISNULL(@i_tipo_subproducto, dc.Nombre)
AND df.Anio between ISNULL(@w_anio_inicio, df.Anio) and ISNULL(@w_anio_fin,
df.Anio)
AND df.Mes between ISNULL(@w_mes_inicio, df.Mes) and ISNULL(@w_mes_final, df.Mes)
AND DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()) >= ISNULL(@i_edad_desde,
DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()))
AND DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()) <= ISNULL(@i_edad_hasta,
DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()))
```

```
AND dcl.Sexo = ISNULL(@i_sexo, dcl.Sexo)
GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad, ds.Sucursal,dc.Nombre
```

Consulta Cuentas Por Estado

```
CREATE PROCEDURE [datamart].[ConsultaCuentasPorEstado]
(
	@i_sucursal                         VARCHAR(100) = NULL,
	@i_estado                           VARCHAR(15) = NULL,
	@i_tipo_subproducto         VARCHAR(30) = NULL,
	@i_fecha_desde                      DATE = NULL,
	@i_fecha_hasta                      DATE = NULL,
	@i_edad_desde               SMALLINT = NULL,
	@i_edad_hasta               SMALLINT = NULL,
	@i_sexo                                     VARCHAR(15) = NULL
)
AS

DECLARE @w_anio_inicio      SMALLINT,
		@w_anio_fin         SMALLINT,
		@w_mes_inicio TINYINT,
		@w_mes_final  TINYINT

SET @w_anio_inicio = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_desde)))
SET @w_anio_fin = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_hasta)))
SET @w_mes_inicio = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_desde)))
SET @w_mes_final = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_hasta)))

IF (@i_estado = 'Activa')

	BEGIN

	IF (@i_sucursal = 'Todos')
		SET @i_sucursal  = NULL

	IF (@i_tipo_subproducto='Todos')
		SET @i_tipo_subproducto = NULL

		SELECT
				df.Anio
				,'NombreMes' =
					CASE df.Mes
						WHEN 1 THEN 'Enero'
						WHEN 2 THEN 'Febrero'
						WHEN 3 THEN 'Marzo'
						WHEN 4 THEN 'Abril'
						WHEN 5 THEN 'Mayo'
						WHEN 6 THEN 'Junio'
						WHEN 7 THEN 'Julio'
						WHEN 8 THEN 'Agosto'
						WHEN 9 THEN 'Septiembre'
						WHEN 10 THEN 'Octubre'
						WHEN 11 THEN 'Nomviembre'
						WHEN 12 THEN 'Diciembre'
					END
				,ds.Provincia
				,ds.Ciudad
				,ds.Sucursal
```

```
                        ,dc.Nombre AS 'TipoSubproducto'
                        ,COUNT(*) AS 'NumeroCuentasActivas'
                        ,@i_estado AS 'EstadoRpt'
            FROM   datamart.HechosAperturarClientesCuentas h
            JOIN   datamart.DimCliente dcl ON h.IdCliente = dcl.IdCliente
            JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
            JOIN   datamart.DimCuenta dc ON h.IdCuenta = dc.IdCuenta
            JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
            JOIN   datamart.DimEstadoCuenta de ON h.IdEstadoCuenta =
de.IdEstadoCuenta
            WHERE  ds.Sucursal = ISNULL(@i_sucursal, ds.Sucursal)
            AND           dc.Nombre = ISNULL(@i_tipo_subproducto, dc.Nombre)
            AND           df.Anio BETWEEN ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
            AND           df.Mes BETWEEN ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)
            AND           DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()) >=
ISNULL(@i_edad_desde, DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()))
            AND           DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()) <=
ISNULL(@i_edad_hasta, DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()))
            AND           dcl.Sexo = ISNULL(@i_sexo, dcl.Sexo)
            AND           de.Nombre = ISNULL(@i_estado, de.Nombre)
            GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad,
ds.Sucursal,dc.Nombre
      END

ELSE IF (@i_estado = 'Liquidada')
      BEGIN

            IF (@i_sucursal = 'Todos')
                  SET @i_sucursal  = NULL

            IF (@i_tipo_subproducto='Todos')
                  SET @i_tipo_subproducto = NULL

            SELECT
                        df.Anio
                        ,'NombreMes' =
                              CASE df.Mes
                                    WHEN 1 THEN 'Enero'
                                    WHEN 2 THEN 'Febrero'
                                    WHEN 3 THEN 'Marzo'
                                    WHEN 4 THEN 'Abril'
                                    WHEN 5 THEN 'Mayo'
                                    WHEN 6 THEN 'Junio'
                                    WHEN 7 THEN 'Julio'
                                    WHEN 8 THEN 'Agosto'
                                    WHEN 9 THEN 'Septiembre'
                                    WHEN 10 THEN 'Octubre'
                                    WHEN 11 THEN 'Nomviembre'
                                    WHEN 12 THEN 'Diciembre'
                              END
                        ,ds.Provincia
                        ,ds.Ciudad
                        ,ds.Sucursal
                        ,dc.Nombre AS 'TipoSubproducto'
                        ,COUNT(*) AS 'NumeroCuentasLiquidadas'
                        ,@i_estado AS 'EstadoRpt'
            FROM   datamart.HechosAperturarClientesCuentas h
            JOIN   datamart.DimCliente dcl ON h.IdCliente = dcl.IdCliente
```

```
              JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
              JOIN   datamart.DimCuenta dc ON h.IdCuenta = dc.IdCuenta
              JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
              JOIN   datamart.DimEstadoCuenta de ON h.IdEstadoCuenta =
de.IdEstadoCuenta
              WHERE  ds.Sucursal = ISNULL(@i_sucursal, ds.Sucursal)
              AND           dc.Nombre = ISNULL(@i_tipo_subproducto, dc.Nombre)
              AND           df.Anio BETWEEN ISNULL(@w_anio_inicio, df.Anio) AND
ISNULL(@w_anio_fin, df.Anio)
              AND           df.Mes BETWEEN ISNULL(@w_mes_inicio, df.Mes) AND
ISNULL(@w_mes_final, df.Mes)
              AND           DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()) >=
ISNULL(@i_edad_desde, DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()))
              AND           DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()) <=
ISNULL(@i_edad_hasta, DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()))
              AND           dcl.Sexo = ISNULL(@i_sexo, dcl.Sexo)
              AND           de.Nombre = ISNULL(@i_estado, de.Nombre)
              GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad,
ds.Sucursal,dc.Nombre
       END

RETURN 0
```

Anexo 5: Lógica para la historia H2 (A6)

Modelo

```
using Denarius.CoreBanking.Shared.Cumplimiento.Model;
using Denarius.CoreBanking.Shared.Model;
using Denarius.CoreBanking.Web.Mvc.ViewModel.ComunesModel;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Denarius.CoreBanking.Web.Mvc.ViewModel.CumplimientoModel
{
    public class ConsultaIndicadoresView
    {
        public Mensaje Mensaje { get; set; }
        public string Estado { get; set; }
        public DateTime FechaDesde { get; set; }
        public DateTime FechaHasta { get; set; }
        public DateTime FechaGeneracion { get; set; }
        public ReportViewerView Reporte { get; set; }
        public bool TipoLocalizacion { get; set; }
        public int IdSucursal { get; set; }
        public string Sucursal { get; set; }
        public string Ciudad { get; set; }
        public string SubProducto { get; set; }
        public string Sexo { get; set; }
        public int? EdadDesde { get; set; }
        public int? EdadHasta { get; set; }
        public bool FiltroFechas { get; set; }
        public bool FiltroSexo { get; set; }
        public bool FiltroEdad { get; set; }
        public bool MostarChartIngresadosSucursal { get; set; }
```

```
        public bool MostarChartIngresadosCiudad { get; set; }
        public List<DatosApertura> ListaDatosApertura {get; set;}
    }
}
```

Vista

```
@model
Denarius.CoreBanking.Web.Mvc.ViewModel.CumplimientoModel.ConsultaIndicadoresView
@{
    var estado = ViewData.TemplateInfo.GetFullHtmlFieldId("Estado");
    var sucursal = ViewData.TemplateInfo.GetFullHtmlFieldId("Sucursal");
    var mostarChartIngresadosSucursal =
ViewData.TemplateInfo.GetFullHtmlFieldId("MostarChartIngresadosSucursal");
}

<script type="text/javascript">

    $('#chkFiltroFechas').bind('click', function () {
        if ($('#chkFiltroFechas').is(":checked")) {
            $('#divFechas').show();
        }
        else {
            $('#divFechas').hide();
        }
    });

    $('#TipoLocalizacionCiudad').change(function () {
        $('#TipoCiudad').show();
        $('#TipoSucursal').hide();
    });
    $('#TipoLocalizacionSucursal').change(function () {
        $('#TipoCiudad').hide();
        $('#TipoSucursal').show();
    });

</script>

<div id="FormContainer">
    @using (@Html.BeginForm("GenerarRptIndClienteActivos", "Cumplimiento"))
    {
        <text>
        @{ Html.RenderPartial("_EditarGenerarReporteIndicador", Model); }
        </text>

        <text>
        <table>
            <tr>
                <td>Estado:
                </td>
                <td>
                    @(
                        Html.Kendo().DropDownListFor(model => model.Estado)
                        .BindTo(new List<SelectListItem> {
                        new SelectListItem { Value="Activo", Text = "Activo" },
                    }))
                </td>
            </tr>
```

```
            <tr>
                <td>
                    Sucursal:
                </td>
                <td >
                    @(
                        Html.Kendo().DropDownListFor(model => model.Sucursal)
                        .OptionLabel("Todos")
                        .DataValueField("Sucursal")
                        .DataTextField("Sucursal")
                        .DataSource(
                                    db => db.Read(r =>
r.Action("BuscarSucursales", "Cumplimiento"))
                                        .ServerFiltering(true)
                            )
                     )
                </td>
            </tr>
        </table>

        <table>
            <tr>
                <td>Fechas:
                </td>
                <td>
                @Html.CheckBoxFor(model => model.FiltroFechas, new { id =
"chkFiltroFechas" })
                </td>
            </tr>
        </table>

        <div id="divFechas">
            <table>
                <tr>
                    <td>Desde:</td>
                    <td>
                        @Html.Kendo().DatePickerFor(model =>
model.FechaDesde).Start(CalendarView.Year).Depth(CalendarView.Year).Format("yyyy
- MM").Footer(false)
                        @Html.ValidationMessageFor(model => model.FechaDesde)
                    </td>
                    <td>Hasta:</td>
                    <td>
                        @Html.Kendo().DatePickerFor(model =>
model.FechaHasta).Start(CalendarView.Year).Depth(CalendarView.Year).Format("yyyy
- MM").Footer(false)
                        @Html.ValidationMessageFor(model => model.FechaHasta)
                    </td>
                </tr>
            </table>
        </div>
        <br />

        <table>
            <tr>
                <td>
                    <input class="button" type="submit" value="Generar Reporte"
id="GenerarReporte"
onclick="submitForm('@Url.Action("GenerarRptIndClienteActivos", "Cumplimiento")
', this);"/>
```

```
                    <input class="button" type="button" value="Limpiar"
id="Limpiar"
onclick="submitFormNoValidation('@Url.Action("RptIndicadoreClienteActivo",
"Cumplimiento")    ', this)"/>
                </td>
            </tr>
        </table>
        <br />

        <div id="divChartIngresadosSucursal"
style="display:@(Model.MostarChartIngresadosSucursal ? string.Empty : "none")">
            <table>
                <tr>
                    <td>

@(Html.Kendo().Chart<Denarius.CoreBanking.Shared.Cumplimiento.Model.DatosApertura
>()
                            .Name("chart")
                            .Title("Indicador de Clientes Activos")
                            .Legend(legend => legend
                                .Visible(true)
                            )
                            .ChartArea(chartArea => chartArea
                                .Background("transparent")
                            )
                            .DataSource(db => db
                                .Read(read =>
read.Action("MostraIndicadorCliente", "Cumplimiento"))
                                .Group(group => group.Add(model =>
model.Sucursal))
                            )
                            .Series(series =>
                            {
                                series.Column(model =>
model.NumeroClientesActivos).Name("Activos").Color("#bb6e36");
                            })
                            .CategoryAxis(axis => axis
                                .Name("series-axis")
                                .Line(line => line.Visible(false))
                             )
                             .CategoryAxis(axis => axis
                                 .Categories(model => model.NombreMes)
                                 .Labels(labels => labels.Rotation(-90))
                             )
                             .ValueAxis(axis => axis.Numeric()
                                 .Labels(labels => labels.Format("{0:N0}"))
                                 .MajorUnit(25)
                             )
                            .Tooltip(tooltip => tooltip
                                .Visible(true)
                                .Template("#= series.name #: #= value #")
                             )
                        )
                    </td>
                </tr>
            </table>
        </div>

        @{
           if (Model.Reporte != null)
```

```
            {
                Html.RenderPartial("ReportViewer", Model.Reporte);
            }
        }
        @Html.MensajeFor(model => model.Mensaje)
        @Html.HiddenFor(model => model.Estado)
        @Html.HiddenFor(model => model.Sucursal)
        @Html.HiddenFor(model => model.MostarChartIngresadosSucursal)

     </text>
    }
</div>

<script type="text/javascript">

    if ($('#chkFiltroFechas').is(":checked")) {
        $('#divFechas').show();
    }
    else {
        $('#divFechas').hide();
    }

</script>
```

Controlador

```
        public List<DatosApertura> ConsultaClientesIngresados(string estado,
string sucursal, string ciudad, DateTime? fechaDesde, DateTime? fechaHasta)
        {
            List<DatosApertura> cabeceras = new List<DatosApertura>();

            using (dao)
            {
                dao.CadenaDeConexion = strConexion[1];
                SqlParametro[] parParametro =
                {
                    new SqlParametro("ReturnValue", SqlDbType.Int, 4),
                    new SqlParametro("@i_estado", SqlDbType.VarChar, 40, (estado
== "" )? null: estado),
                    new SqlParametro("@i_sucursal", SqlDbType.VarChar, 40,
(sucursal == "" )? null: sucursal),
                    new SqlParametro("@i_ciudad", SqlDbType.VarChar, 40, (ciudad
== "" )? null: ciudad),
                    new SqlParametro("@i_fecha_desde", SqlDbType.Date, 3,
fechaDesde),
                    new SqlParametro("@i_fecha_hasta", SqlDbType.Date, 3,
fechaHasta),
                };

                parParametro[0].Direction = ParameterDirection.ReturnValue;

                cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaClientesIngresados", parParametro,
"ClientesIngresados");

                if (dao.ReturnParametro != 0)
                    throw new Error(dao.ReturnParametro);
            }
```

```
            return cabeceras;
        }

        public List<DatosApertura> ConsultaClientesActivos(string estado, string
sucursal, string ciudad, DateTime? fechaDesde, DateTime? fechaHasta)
        {
            List<DatosApertura> cabeceras = new List<DatosApertura>();

            using (dao)
            {
                dao.CadenaDeConexion = strConexion[1];
                SqlParametro[] parParametro =
                {
                    new SqlParametro("ReturnValue", SqlDbType.Int, 4),
                    new SqlParametro("@i_estado", SqlDbType.VarChar, 40, (estado
== "" )? null: estado),
                    new SqlParametro("@i_sucursal", SqlDbType.VarChar, 40,
(sucursal == "" )? null: sucursal),
                    new SqlParametro("@i_ciudad", SqlDbType.VarChar, 40, (ciudad
== "" )? null: ciudad),
                    new SqlParametro("@i_fecha_desde", SqlDbType.Date, 3,
fechaDesde),
                    new SqlParametro("@i_fecha_hasta", SqlDbType.Date, 3,
fechaHasta),
                };

                parParametro[0].Direction = ParameterDirection.ReturnValue;

                cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaClientesActivos", parParametro,
"ClientesActivos");

                if (dao.ReturnParametro != 0)
                    throw new Error(dao.ReturnParametro);
            }

            return cabeceras;
        }

        public List<DatosApertura> ConsultaClientesPasivos(string estado, string
sucursal, string ciudad, DateTime? fechaDesde, DateTime? fechaHasta)
        {
            List<DatosApertura> cabeceras = new List<DatosApertura>();

            using (dao)
            {
                dao.CadenaDeConexion = strConexion[1];
                SqlParametro[] parParametro =
                {
                    new SqlParametro("ReturnValue", SqlDbType.Int, 4),
                    new SqlParametro("@i_estado", SqlDbType.VarChar, 40, (estado
== "" )? null: estado),
                    new SqlParametro("@i_sucursal", SqlDbType.VarChar, 40,
(sucursal == "" )? null: sucursal),
                    new SqlParametro("@i_ciudad", SqlDbType.VarChar, 40, (ciudad
== "" )? null: ciudad),
                    new SqlParametro("@i_fecha_desde", SqlDbType.Date, 3,
fechaDesde),
                    new SqlParametro("@i_fecha_hasta", SqlDbType.Date, 3,
fechaHasta),
```

```
                };

                parParametro[0].Direction = ParameterDirection.ReturnValue;

                cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaClientesPasivos", parParametro,
"ClientesPasivos");

                if (dao.ReturnParametro != 0)
                    throw new Error(dao.ReturnParametro);
            }

            return cabeceras;
        }

        public List<DatosApertura> ConsultaClientesTodos(string estado, string
sucursal, string ciudad, DateTime? fechaDesde, DateTime? fechaHasta)
        {
            List<DatosApertura> cabeceras = new List<DatosApertura>();

            using (dao)
            {
                dao.CadenaDeConexion = strConexion[1];
                SqlParametro[] parParametro =
                {
                    new SqlParametro("ReturnValue", SqlDbType.Int, 4),
                    new SqlParametro("@i_estado", SqlDbType.VarChar, 40, (estado
== "" )? null: estado),
                    new SqlParametro("@i_sucursal", SqlDbType.VarChar, 40,
(sucursal == "" )? null: sucursal),
                    new SqlParametro("@i_ciudad", SqlDbType.VarChar, 40, (ciudad
== "" )? null: ciudad),
                    new SqlParametro("@i_fecha_desde", SqlDbType.Date, 3,
fechaDesde),
                    new SqlParametro("@i_fecha_hasta", SqlDbType.Date, 3,
fechaHasta),
                };

                parParametro[0].Direction = ParameterDirection.ReturnValue;

                cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaClientesTodos", parParametro,
"ClientesTodos");

                if (dao.ReturnParametro != 0)
                    throw new Error(dao.ReturnParametro);
            }

            return cabeceras;
        }

        public List<DatosApertura> ConsultaCuentasAperturadas(string estado,
string sucursal, string ciudad, string tipoSubProducto, DateTime? fechaDesde,
DateTime? fechaHasta, int? edadDesde, int? edadHasta, string sexo)
        {
            List<DatosApertura> cabeceras = new List<DatosApertura>();

            using (dao)
            {
                dao.CadenaDeConexion = strConexion[1];
```

```
                SqlParametro[] parParametro =
                {
                    new SqlParametro("ReturnValue", SqlDbType.Int, 4),
                    new SqlParametro("@i_estado", SqlDbType.VarChar, 40, (estado
== "" )? null: estado),
                    new SqlParametro("@i_sucursal", SqlDbType.VarChar, 40,
(sucursal == "" )? null: sucursal),
                    new SqlParametro("@i_tipo_subproducto", SqlDbType.VarChar,
40, (tipoSubProducto == "" )? null: tipoSubProducto),
                    new SqlParametro("@i_fecha_desde", SqlDbType.Date, 3,
fechaDesde),
                    new SqlParametro("@i_fecha_hasta", SqlDbType.Date, 3,
fechaHasta),
                    new SqlParametro("@i_edad_desde", SqlDbType.Int, 4,
edadDesde),
                    new SqlParametro("@i_edad_hasta", SqlDbType.Int, 4,
edadHasta),
                    new SqlParametro("@i_sexo", SqlDbType.VarChar, 40,(sexo ==
"" )? null:  sexo),
                };

                parParametro[0].Direction = ParameterDirection.ReturnValue;

                cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaCuentasAperturadas", parParametro,
"CuentasAperturadas");

                if (dao.ReturnParametro != 0)
                    throw new Error(dao.ReturnParametro);
            }

            return cabeceras;
        }

        public List<DatosApertura> ConsultaCuentasPorEstado(string estado, string
sucursal, string ciudad, string tipoSubProducto, DateTime? fechaDesde, DateTime?
fechaHasta, int? edadDesde, int? edadHasta, string sexo)
        {
            List<DatosApertura> cabeceras = new List<DatosApertura>();

            using (dao)
            {
                dao.CadenaDeConexion = strConexion[1];
                SqlParametro[] parParametro =
                {
                    new SqlParametro("ReturnValue", SqlDbType.Int, 4),
                    new SqlParametro("@i_estado", SqlDbType.VarChar, 40, (estado
== "" )? null: estado),
                    new SqlParametro("@i_sucursal", SqlDbType.VarChar, 40,
(sucursal == "" )? null: sucursal),
                    new SqlParametro("@i_tipo_subproducto", SqlDbType.VarChar,
40, (tipoSubProducto == "" )? null: tipoSubProducto),
                    new SqlParametro("@i_fecha_desde", SqlDbType.Date, 3,
fechaDesde),
                    new SqlParametro("@i_fecha_hasta", SqlDbType.Date, 3,
fechaHasta),
                    new SqlParametro("@i_edad_desde", SqlDbType.Int, 4,
edadDesde),
                    new SqlParametro("@i_edad_hasta", SqlDbType.Int, 4,
edadHasta),
```

```
                    new SqlParametro("@i_sexo", SqlDbType.VarChar, 40,(sexo ==
"" )? null:  sexo),
                };

                parParametro[0].Direction = ParameterDirection.ReturnValue;

                if(estado == "Activa")
                    cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaCuentasPorEstado", parParametro,
"CuentasActivas");

                else if (estado == "Liquidada")
                    cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaCuentasPorEstado", parParametro,
"CuentasLiquidadas");

                if (dao.ReturnParametro != 0)
                    throw new Error(dao.ReturnParametro);
            }

            return cabeceras;
        }

        public List<DatosApertura> ConsultaCuentasTodas(string estado, string
sucursal, string ciudad, string tipoSubProducto, DateTime? fechaDesde, DateTime?
fechaHasta)
        {
            List<DatosApertura> cabeceras = new List<DatosApertura>();

            using (dao)
            {
                dao.CadenaDeConexion = strConexion[1];
                SqlParametro[] parParametro =
                {
                    new SqlParametro("ReturnValue", SqlDbType.Int, 4),
                    new SqlParametro("@i_estado", SqlDbType.VarChar, 40, (estado
== "" )? null: estado),
                    new SqlParametro("@i_sucursal", SqlDbType.VarChar, 40,
(sucursal == "" )? null: sucursal),
                    new SqlParametro("@i_tipo_subproducto", SqlDbType.VarChar,
40, (tipoSubProducto == "" )? null: tipoSubProducto),
                    new SqlParametro("@i_fecha_desde", SqlDbType.Date, 3,
fechaDesde),
                    new SqlParametro("@i_fecha_hasta", SqlDbType.Date, 3,
fechaHasta),
                };

                parParametro[0].Direction = ParameterDirection.ReturnValue;

                cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaCuentasTodas", parParametro,
"CuentasTodas");

                if (dao.ReturnParametro != 0)
                    throw new Error(dao.ReturnParametro);
            }

            return cabeceras;
        }
```

Anexo 6: MVC para la historia H2 (A7)

```
       #region Indicadores
        public PartialViewResult Index()
        {
            ConsultaIndicadoresView datosConsulta = new
ConsultaIndicadoresView();
            try
            {
                return PartialView("_EditarGenerarReporteIndicador",
datosConsulta);
            }
            catch (Exception ex)
            {
                string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                datosConsulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores", mensaje, tipoMensaje:
TipoMensaje.Error);

                return PartialView("_ReportesIndicadores", datosConsulta);
            }
        }

        public JsonResult BuscarSucursales()
        {
            List<DatosSucursal> sucursales =
CumplimientoAgent.ConsultarSucursales();

            if (sucursales == null)
                this.RetornarErrorJsonResult("No se encontró sucursales");

            return Json(sucursales, JsonRequestBehavior.AllowGet);
        }

        public ActionResult MostraIndicadorCliente()
        {
            List<DatosApertura> datosConsultaCliente =
Sesion.GetFromSession<List<DatosApertura>>("ListaDatosCliente");

            if (datosConsultaCliente != null)
                return Json(datosConsultaCliente);

            return Json((new List<DatosApertura>()));
        }

        #region Gestion
        //Clientes Ingresados
        public ActionResult ReportesIndicadores()
        {
            ConsultaIndicadoresView indicadoresView = new
ConsultaIndicadoresView();
            indicadoresView.FechaDesde = DateTime.Now;
            indicadoresView.FechaHasta = DateTime.Now;
            indicadoresView.MostarChartIngresadosSucursal = false;
            Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", null);
            return PartialView("_ReportesIndicadores", indicadoresView);
        }
```

```
        public PartialViewResult
GenerarRptIndClienteIngresados(ConsultaIndicadoresView consulta)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    ModelState.Clear();
                    Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente",
null);
                    var tenant = Sesion.Usuario.TenantUsr;
                    string estado = "";
                    DateTime? fechaDesde = null;
                    DateTime? fechaHasta = null;
                    string sucursal = "";

                    estado = consulta.Estado;
                    sucursal = consulta.Sucursal;

                    if (consulta.FiltroFechas)
                    {
                        fechaDesde = consulta.FechaDesde;
                        fechaHasta = consulta.FechaHasta;
                    }

                    List<DatosApertura> datos = new List<DatosApertura>();

                    if (estado == "Ingresado")
                    {
                        datos =
CumplimientoAgent.ConsultaClientesIngresados(estado, sucursal, null, fechaDesde,
fechaHasta);
                        consulta.MostarChartIngresadosSucursal = true;
                    }

                    if (datos != null && datos.Count > 0)
                    {

Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", datos);
                        consulta.Reporte = new ReportViewerView();
                        consulta.Reporte.ReporteBook =
CumplimientoAgent.GenerarReporteProcesoApertura(datos, tenant);
                    }
                    else
                    {
                        consulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores - Consulta Clientes
Ingresados", "No se encuentraron registros.", tipoMensaje:
TipoMensaje.Informacion);
                        consulta.MostarChartIngresadosSucursal = false;
                    }
                }
                catch (Exception ex)
                {
                    string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                    consulta.Mensaje = new Mensaje("Error al Consultar
Indicadores", mensaje, tipoMensaje: TipoMensaje.Error);
                }
```

```
            }
            return PartialView("_ReportesIndicadores", consulta);
        }

        //Clientes Acitivos
        public ActionResult RptIndicadoreClienteActivo()
        {
            ConsultaIndicadoresView indicadoresView = new
ConsultaIndicadoresView();
            indicadoresView.FechaDesde = DateTime.Now;
            indicadoresView.FechaHasta = DateTime.Now;
            indicadoresView.MostarChartIngresadosSucursal = false;
            Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", null);
            return PartialView("_ConsultaClienteActivo", indicadoresView);
        }

        public PartialViewResult
GenerarRptIndClienteActivos(ConsultaIndicadoresView consulta)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    ModelState.Clear();
                    Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente",
null);
                    var tenant = Sesion.Usuario.TenantUsr;
                    string estado = "";
                    DateTime? fechaDesde = null;
                    DateTime? fechaHasta = null;
                    string sucursal = "";

                    estado = consulta.Estado;
                    sucursal = consulta.Sucursal;

                    if (consulta.FiltroFechas)
                    {
                        fechaDesde = consulta.FechaDesde;
                        fechaHasta = consulta.FechaHasta;
                    }

                    List<DatosApertura> datos = new List<DatosApertura>();

                    if (estado == "Activo")
                    {
                        datos = CumplimientoAgent.ConsultaClientesActivos(estado,
sucursal, null, fechaDesde, fechaHasta);
                        consulta.MostarChartIngresadosSucursal = true;
                    }

                    if (datos != null && datos.Count > 0)
                    {

Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", datos);
                        consulta.Reporte = new ReportViewerView();
                        consulta.Reporte.ReporteBook =
CumplimientoAgent.GenerarReporteProcesoApertura(datos, tenant);
                    }
                    else
                    {
```

```
                        consulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores - Consulta Clientes
Activos", "No se encuentraron registros.", tipoMensaje: TipoMensaje.Informacion);
                        consulta.MostarChartIngresadosSucursal = false;
                    }
                }
                catch (Exception ex)
                {
                    string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                    consulta.Mensaje = new Mensaje("Error al Consultar
Indicadores", mensaje, tipoMensaje: TipoMensaje.Error);
                }
            }
            return PartialView("_ConsultaClienteActivo", consulta);
        }

        //Clientes Pasivos
        public ActionResult RptIndicadoreClientePasivo()
        {
            ConsultaIndicadoresView indicadoresView = new
ConsultaIndicadoresView();
            indicadoresView.FechaDesde = DateTime.Now;
            indicadoresView.FechaHasta = DateTime.Now;
            indicadoresView.MostarChartIngresadosSucursal = false;
            Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", null);
            return PartialView("_ConsultaClientePasivo", indicadoresView);
        }

        public PartialViewResult
GenerarRptIndClientePasivo(ConsultaIndicadoresView consulta)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    ModelState.Clear();
                    Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente",
null);
                    var tenant = Sesion.Usuario.TenantUsr;
                    string estado = "";
                    DateTime? fechaDesde = null;
                    DateTime? fechaHasta = null;
                    string sucursal = "";

                    estado = consulta.Estado;
                    sucursal = consulta.Sucursal;

                    if (consulta.FiltroFechas)
                    {
                        fechaDesde = consulta.FechaDesde;
                        fechaHasta = consulta.FechaHasta;
                    }

                    List<DatosApertura> datos = new List<DatosApertura>();

                    if (estado == "Pasivo")
                    {
                        datos = CumplimientoAgent.ConsultaClientesPasivos(estado,
sucursal, null, fechaDesde, fechaHasta);
```

```
                        consulta.MostarChartIngresadosSucursal = true;
                    }

                    if (datos != null && datos.Count > 0)
                    {

Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", datos);
                        consulta.Reporte = new ReportViewerView();
                        consulta.Reporte.ReporteBook =
CumplimientoAgent.GenerarReporteProcesoApertura(datos, tenant);
                    }
                    else
                    {
                        consulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores - Consulta Clientes
Pasivos", "No se encuentraron registros.", tipoMensaje: TipoMensaje.Informacion);
                        consulta.MostarChartIngresadosSucursal = false;
                    }
                }
                catch (Exception ex)
                {
                    string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                    consulta.Mensaje = new Mensaje("Error al Consultar
Indicadores", mensaje, tipoMensaje: TipoMensaje.Error);
                }
            }
            return PartialView("_ConsultaClientePasivo", consulta);
        }

        //Clientes Todos
        public ActionResult RptIndicadoreClienteTodo()
        {
            ConsultaIndicadoresView indicadoresView = new
ConsultaIndicadoresView();
            indicadoresView.FechaDesde = DateTime.Now;
            indicadoresView.FechaHasta = DateTime.Now;
            indicadoresView.MostarChartIngresadosSucursal = false;
            Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", null);
            return PartialView("_ConsultaClienteTodo", indicadoresView);
        }

        public PartialViewResult GenerarRptIndClienteTodo(ConsultaIndicadoresView
consulta)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    ModelState.Clear();
                    Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente",
null);
                    var tenant = Sesion.Usuario.TenantUsr;
                    string estado = "";
                    DateTime? fechaDesde = null;
                    DateTime? fechaHasta = null;
                    string sucursal = "";

                    estado = consulta.Estado;
                    sucursal = consulta.Sucursal;
```

```
                    if (consulta.FiltroFechas)
                    {
                        fechaDesde = consulta.FechaDesde;
                        fechaHasta = consulta.FechaHasta;
                    }

                    List<DatosApertura> datos = new List<DatosApertura>();

                    if (estado == "Todos")
                    {
                        datos = CumplimientoAgent.ConsultaClientesTodos(estado,
sucursal, null, fechaDesde, fechaHasta);
                        consulta.MostarChartIngresadosSucursal = true;
                    }

                    if (datos != null && datos.Count > 0)
                    {

Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", datos);
                        consulta.Reporte = new ReportViewerView();
                        consulta.Reporte.ReporteBook =
CumplimientoAgent.GenerarReporteProcesoApertura(datos, tenant);
                    }
                    else
                    {
                        consulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores - Consulta Clientes", "No
se encuentraron registros.", tipoMensaje: TipoMensaje.Informacion);
                        consulta.MostarChartIngresadosSucursal = false;
                    }
                }
                catch (Exception ex)
                {
                    string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                    consulta.Mensaje = new Mensaje("Error al Consultar
Indicadores", mensaje, tipoMensaje: TipoMensaje.Error);
                }
            }
            return PartialView("_ConsultaClienteTodo", consulta);
        }

        //Cuentas Aperturadas
        public ActionResult ReptIndicadorCuentasAperturadas()
        {
            ConsultaIndicadoresView indicadoresView = new
ConsultaIndicadoresView();
            indicadoresView.MostarChartIngresadosSucursal = false;
            indicadoresView.FechaDesde = DateTime.Now;
            indicadoresView.FechaHasta = DateTime.Now;
            indicadoresView.TipoLocalizacion = true;
            Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", null);
            return PartialView("_ConsultaCuentaAperturada", indicadoresView);
        }

        public PartialViewResult
GenerarRptIndCuentaAperturada(ConsultaIndicadoresView consulta)
        {
            if (ModelState.IsValid)
```

```
            {
                try
                {
                    var tenant = Sesion.Usuario.TenantUsr;
                    string estado = "";
                    DateTime? fechaDesde = null;
                    DateTime? fechaHasta = null;
                    string sucursal = "";
                    string ciudad = "";
                    string tipoSubProducto = "";
                    string mes = "";
                    string sexo = "";
                    int? edadDesde = null;
                    int? edadHasta = null;

                    estado = consulta.Estado;
                    sucursal = consulta.Sucursal;
                    tipoSubProducto = consulta.SubProducto;

                    if (consulta.FiltroFechas)
                    {
                        fechaDesde = consulta.FechaDesde;
                        fechaHasta = consulta.FechaHasta;
                    }

                    if (consulta.FiltroEdad)
                    {
                        edadDesde = consulta.EdadDesde;
                        edadHasta = consulta.EdadHasta;
                    }

                    if (consulta.FiltroSexo)
                        sexo = consulta.Sexo;

                    if (consulta.TipoLocalizacion)
                        ciudad = consulta.Ciudad;
                    else
                        sucursal = consulta.Sucursal;

                    List<DatosApertura> datos = new List<DatosApertura>();


                    if (estado == "Aperturada")
                    {
                        datos =
CumplimientoAgent.ConsultaCuentasAperturadas(estado, sucursal, ciudad,
tipoSubProducto, fechaDesde, fechaHasta, edadDesde, edadHasta, sexo);
                        consulta.MostarChartIngresadosSucursal = true;
                    }

                    if (datos != null && datos.Count > 0)
                    {

Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", datos);
                        //consulta.Reporte = new ReportViewerView();
                        //consulta.Reporte.ReporteBook =
CumplimientoAgent.GenerarReporteProcesoAperturaCtas(datos, tenant);
                    }

                    else
```

```
                    {
                        consulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores - Consulta Cuentas
Aperturadas", "No se encuentraron registros.", tipoMensaje:
TipoMensaje.Informacion);
                        consulta.MostarChartIngresadosSucursal = false;
                    }
                }
                catch (Exception ex)
                {
                    string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                    consulta.Mensaje = new Mensaje("Error al Consultar
Indicadores", mensaje, tipoMensaje: TipoMensaje.Error);
                }
            }
            return PartialView("_ConsultaCuentaAperturada", consulta);
        }

        //Cuentas Activas
        public ActionResult ReptIndicadorCuentasActivas()
        {
            ConsultaIndicadoresView indicadoresView = new
ConsultaIndicadoresView();
            indicadoresView.MostarChartIngresadosSucursal = false;
            indicadoresView.FechaDesde = DateTime.Now;
            indicadoresView.FechaHasta = DateTime.Now;
            indicadoresView.TipoLocalizacion = true;
            Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", null);
            return PartialView("_ConsultaCuentaActiva", indicadoresView);
        }

        public PartialViewResult
GenerarRptIndCuentaActivas(ConsultaIndicadoresView consulta)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    var tenant = Sesion.Usuario.TenantUsr;
                    string estado = "";
                    DateTime? fechaDesde = null;
                    DateTime? fechaHasta = null;
                    string sucursal = "";
                    string ciudad = "";
                    string tipoSubProducto = "";
                    string mes = "";
                    string sexo = "";
                    int? edadDesde = null;
                    int? edadHasta = null;

                    estado = consulta.Estado;
                    sucursal = consulta.Sucursal;
                    tipoSubProducto = consulta.SubProducto;

                    if (consulta.FiltroFechas)
                    {
                        fechaDesde = consulta.FechaDesde;
                        fechaHasta = consulta.FechaHasta;
                    }
```

```
                    if (consulta.FiltroEdad)
                    {
                        edadDesde = consulta.EdadDesde;
                        edadHasta = consulta.EdadHasta;
                    }

                    if (consulta.FiltroSexo)
                        sexo = consulta.Sexo;

                    if (consulta.TipoLocalizacion)
                        ciudad = consulta.Ciudad;
                    else
                        sucursal = consulta.Sucursal;

                    List<DatosApertura> datos = new List<DatosApertura>();

                    if (estado == "Activa")
                    {
                        datos =
CumplimientoAgent.ConsultaCuentasPorEstado(estado, sucursal, ciudad,
tipoSubProducto, fechaDesde, fechaHasta, edadDesde, edadHasta, sexo);
                        consulta.MostarChartIngresadosSucursal = true;
                    }

                    if (datos != null && datos.Count > 0)
                    {

Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", datos);
                        //consulta.Reporte = new ReportViewerView();
                        //consulta.Reporte.ReporteBook =
CumplimientoAgent.GenerarReporteProcesoAperturaCtas(datos, tenant);
                    }
                    else
                    {
                        consulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores - Consulta Cuentas
Activas", "No se encuentraron registros.", tipoMensaje: TipoMensaje.Informacion);
                        consulta.MostarChartIngresadosSucursal = false;
                    }
                }
                catch (Exception ex)
                {
                    string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                    consulta.Mensaje = new Mensaje("Error al Consultar
Indicadores", mensaje, tipoMensaje: TipoMensaje.Error);
                }
            }
            return PartialView("_ConsultaCuentaActiva", consulta);
        }

        //Cuentas Liquidadas
        public ActionResult ReptIndicadorCuentasLiquidadas()
        {
            ConsultaIndicadoresView indicadoresView = new
ConsultaIndicadoresView();
            indicadoresView.MostarChartIngresadosSucursal = false;
            indicadoresView.FechaDesde = DateTime.Now;
            indicadoresView.FechaHasta = DateTime.Now;
```

```
            indicadoresView.TipoLocalizacion = true;
            Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", null);
            return PartialView("_ConsultaCuentaLiquidada", indicadoresView);
        }

        public PartialViewResult
GenerarRptIndCuentaLiquidadas(ConsultaIndicadoresView consulta)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    var tenant = Sesion.Usuario.TenantUsr;
                    string estado = "";
                    DateTime? fechaDesde = null;
                    DateTime? fechaHasta = null;
                    string sucursal = "";
                    string ciudad = "";
                    string tipoSubProducto = "";
                    string mes = "";
                    string sexo = "";
                    int? edadDesde = null;
                    int? edadHasta = null;

                    estado = consulta.Estado;
                    sucursal = consulta.Sucursal;
                    tipoSubProducto = consulta.SubProducto;

                    if (consulta.FiltroFechas)
                    {
                        fechaDesde = consulta.FechaDesde;
                        fechaHasta = consulta.FechaHasta;
                    }

                    if (consulta.FiltroEdad)
                    {
                        edadDesde = consulta.EdadDesde;
                        edadHasta = consulta.EdadHasta;
                    }

                    if (consulta.FiltroSexo)
                        sexo = consulta.Sexo;

                    if (consulta.TipoLocalizacion)
                        ciudad = consulta.Ciudad;
                    else
                        sucursal = consulta.Sucursal;

                    List<DatosApertura> datos = new List<DatosApertura>();

                    if (estado == "Liquidada")
                    {
                        datos =
CumplimientoAgent.ConsultaCuentasPorEstado(estado, sucursal, ciudad,
tipoSubProducto, fechaDesde, fechaHasta, edadDesde, edadHasta, sexo);
                        consulta.MostarChartIngresadosSucursal = true;
                    }

                    if (datos != null && datos.Count > 0)
                    {
```

```
Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", datos);
                        //consulta.Reporte = new ReportViewerView();
                        //consulta.Reporte.ReporteBook =
CumplimientoAgent.GenerarReporteProcesoAperturaCtas(datos, tenant);
                    }
                    else
                    {
                        consulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores - Consulta Cuentas
Liquidadas", "No se encuentraron registros.", tipoMensaje:
TipoMensaje.Informacion);
                        consulta.MostarChartIngresadosSucursal = false;
                    }
                }
                catch (Exception ex)
                {
                    string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                    consulta.Mensaje = new Mensaje("Error al Consultar
Indicadores", mensaje, tipoMensaje: TipoMensaje.Error);
                }
            }
            return PartialView("_ConsultaCuentaLiquidada", consulta);
        }
```

Anexo 7: Creación de procesos almacenados para la historia H3 (A9)

```
CREATE PROCEDURE [datamart].[ConsultaSaldoPromedio]
(
      @i_sucursal                       VARCHAR(100) = NULL,
      @i_estado                         VARCHAR(15) = NULL,
      @i_tipo_subproducto         VARCHAR(30) = NULL,
      @i_fecha_desde                    DATE = NULL,
      @i_fecha_hasta                    DATE = NULL,
      @i_edad_desde               SMALLINT = NULL,
      @i_edad_hasta               SMALLINT = NULL,
      @i_sexo                                 VARCHAR(15) = NULL
)
AS

DECLARE @w_anio_inicio      SMALLINT,
            @w_anio_fin         SMALLINT,
            @w_mes_inicio TINYINT,
            @w_mes_final  TINYINT

SET @w_anio_inicio = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_desde)))
SET @w_anio_fin = CONVERT(SMALLINT, (DATENAME(YEAR, @i_fecha_hasta)))
SET @w_mes_inicio = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_desde)))
SET @w_mes_final = CONVERT(TINYINT, (DATEPART(MONTH, @i_fecha_hasta)))

IF (@i_sucursal = 'Todos')
      SET @i_sucursal  = NULL

IF (@i_tipo_subproducto='Todos')
      SET @i_tipo_subproducto = NULL

SELECT
```

```
              df.Anio
              ,'NombreMes' =
                     CASE df.Mes
                            WHEN 1 THEN 'Enero'
                            WHEN 2 THEN 'Febrero'
                            WHEN 3 THEN 'Marzo'
                            WHEN 4 THEN 'Abril'
                            WHEN 5 THEN 'Mayo'
                            WHEN 6 THEN 'Junio'
                            WHEN 7 THEN 'Julio'
                            WHEN 8 THEN 'Agosto'
                            WHEN 9 THEN 'Septiembre'
                            WHEN 10 THEN 'Octubre'
                            WHEN 11 THEN 'Nomviembre'
                            WHEN 12 THEN 'Diciembre'
                     END
              ,ds.Provincia
              ,ds.Ciudad
              ,ds.Sucursal
              ,dc.Nombre AS 'TipoSubproducto'
              ,ROUND(AVG(SaldoApertura), 2,0) AS 'SaldoApertura'
              ,@i_estado AS 'EstadoRpt'
FROM   datamart.HechosAperturarClientesCuentas h
JOIN   datamart.DimCliente dcl ON h.IdCliente = dcl.IdCliente
JOIN   datamart.DimSucursales ds ON h.IdSucursal = ds.IdSucursal
JOIN   datamart.DimCuenta dc ON h.IdCuenta = dc.IdCuenta
JOIN   datamart.DimFecha df ON h.IdFecha = df.IdFecha
where  ds.Sucursal = ISNULL(@i_sucursal, ds.Sucursal)
AND           dc.Nombre = ISNULL(@i_tipo_subproducto, dc.Nombre)
AND df.Anio between ISNULL(@w_anio_inicio, df.Anio) and ISNULL(@w_anio_fin,
df.Anio)
AND df.Mes between ISNULL(@w_mes_inicio, df.Mes) and ISNULL(@w_mes_final, df.Mes)
AND DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()) >= ISNULL(@i_edad_desde,
DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()))
AND DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()) <= ISNULL(@i_edad_hasta,
DATEDIFF(yy,dcl.FechaNacimiento, GETDATE()))
AND dcl.Sexo = ISNULL(@i_sexo, dcl.Sexo)
GROUP BY df.Anio, df.Mes, ds.Provincia, ds.Ciudad, ds.Sucursal,dc.Nombre
```

Anexo 8: Lógica para la historia H3 (A10)

```
        public List<DatosApertura> ConsultaSaldoPromedio(string estado, string
sucursal, string ciudad, string tipoSubProducto, DateTime? fechaDesde, DateTime?
fechaHasta, int? edadDesde, int? edadHasta, string sexo)
        {
            List<DatosApertura> cabeceras = new List<DatosApertura>();

            using (dao)
            {
                dao.CadenaDeConexion = strConexion[1];
                SqlParametro[] parParametro =
                {
                    new SqlParametro("ReturnValue", SqlDbType.Int, 4),
                    new SqlParametro("@i_estado", SqlDbType.VarChar, 40, (estado
== "" )? null: estado),
                    new SqlParametro("@i_sucursal", SqlDbType.VarChar, 40,
(sucursal == "" )? null: sucursal),
```

```
                    new SqlParametro("@i_tipo_subproducto", SqlDbType.VarChar,
40, (tipoSubProducto == "" )? null: tipoSubProducto),
                    new SqlParametro("@i_fecha_desde", SqlDbType.Date, 3,
fechaDesde),
                    new SqlParametro("@i_fecha_hasta", SqlDbType.Date, 3,
fechaHasta),
                    new SqlParametro("@i_edad_desde", SqlDbType.Int, 4,
edadDesde),
                    new SqlParametro("@i_edad_hasta", SqlDbType.Int, 4,
edadHasta),
                    new SqlParametro("@i_sexo", SqlDbType.VarChar, 40,(sexo ==
"" )? null:  sexo),
                };

                parParametro[0].Direction = ParameterDirection.ReturnValue;

                cabeceras =
dao.TransmitirRpc<DatosApertura>("ConsultaSaldoPromedio", parParametro,
"SaldoPromedio");

                if (dao.ReturnParametro != 0)
                    throw new Error(dao.ReturnParametro);
            }

            return cabeceras;
        }
```

Anexo 9: MVC para la historia H3 (A11)

Modelo

```
using Denarius.CoreBanking.Shared.Cumplimiento.Model;
using Denarius.CoreBanking.Shared.Model;
using Denarius.CoreBanking.Web.Mvc.ViewModel.ComunesModel;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Denarius.CoreBanking.Web.Mvc.ViewModel.CumplimientoModel
{
    public class ConsultaIndicadoresView
    {
        public Mensaje Mensaje { get; set; }
        public string Estado { get; set; }
        public DateTime FechaDesde { get; set; }
        public DateTime FechaHasta { get; set; }
        public DateTime FechaGeneracion { get; set; }
        public ReportViewerView Reporte { get; set; }
        public bool TipoLocalizacion { get; set; }
        public int IdSucursal { get; set; }
        public string Sucursal { get; set; }
        public string Ciudad { get; set; }
        public string SubProducto { get; set; }
        public string Sexo { get; set; }
        public int? EdadDesde { get; set; }
        public int? EdadHasta { get; set; }
        public bool FiltroFechas { get; set; }
```

```
        public bool FiltroSexo { get; set; }
        public bool FiltroEdad { get; set; }
        public bool MostarChartIngresadosSucursal { get; set; }
        public bool MostarChartIngresadosCiudad { get; set; }
        public List<DatosApertura> ListaDatosApertura {get; set;}

    }
}
```

Clase DatosApertura:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Denarius.CoreBanking.Shared.Cumplimiento.Model
{
    public class DatosApertura
    {
        public int Anio { get; set; }
        public string Mes { get; set; }
        public string Provincia { get; set; }
        public string Ciudad { get; set; }
        public string Sucursal { get; set; }
        public int? NumeroClientesAperturados { get; set; }
        public int? NumeroClientesActivos { get; set; }
        public int? NumeroClientesPasivos { get; set; }
        public int? NumeroCuentasAperturadas { get; set; }
        public int? NumeroCuentasActivas { get; set; }
        public int? NumeroCuentasLiquidadas { get; set; }
        public string TipoSubproducto { get; set; }
        public string EstadoRpt { get; set; }
        public DateTime Fecha { get; set; }
        public string NombreMes { get; set; }
        public int Aperturados { get; set; }
        public char Sexo { get; set; }
        public int Edad { get; set; }
        public decimal SaldoApertura { get; set; }
    }
}
```

Vista

```
@model
Denarius.CoreBanking.Web.Mvc.ViewModel.CumplimientoModel.ConsultaIndicadoresView
@{
    var sucursal = ViewData.TemplateInfo.GetFullHtmlFieldId("Sucursal");
    var mostarChartIngresadosSucursal =
ViewData.TemplateInfo.GetFullHtmlFieldId("MostarChartIngresadosSucursal");
}

<script type="text/javascript">
    $('#chkFiltroFechas').bind('click', function () {
        if ($('#chkFiltroFechas').is(":checked")) {
            $('#divFechas').show();
        }
```

```
        else {
            $('#divFechas').hide();
        }
    });

    $('#chkFiltroSexo').bind('click', function () {
        if ($('#chkFiltroSexo').is(":checked")) {
            $('#divSexo').show();
        }
        else {
            $('#divSexo').hide();
        }
    });

    $('#chkFiltroEdad').bind('click', function () {
        if ($('#chkFiltroEdad').is(":checked")) {
            $('#divEdad').show();
        }
        else {
            $('#divEdad').hide();
        }
    });
</script>

<div id="FormContainer">
    @using (@Html.BeginForm("GenerarRptIndSaldoApertura", "Cumplimiento"))
    {
        <text>
        <table>
            <tr>
                <td>
                    Sucursal:
                </td>
                <td >
                    @(
                        Html.Kendo().DropDownListFor(model => model.Sucursal)
                        .OptionLabel("Todos")
                        .DataValueField("Sucursal")
                        .DataTextField("Sucursal")
                        .DataSource(
                                    db => db.Read(r =>
r.Action("BuscarSucursales", "Cumplimiento"))
                                        .ServerFiltering(true)
                            )
                     )
                </td>
            </tr>
            <tr>
                <td>SubProducto:
                </td>
                <td>
                    @( Html.Kendo().DropDownListFor(model => model.SubProducto)
                        .BindTo(new List<SelectListItem> {
                        new SelectListItem { Value="Cuenta De Ahorros", Text =
"Cuenta De Ahorros" },
                        new SelectListItem { Value="Certificados De Aportacion",
Text = "Certificados De Aportacion" },
                        new SelectListItem { Value="Ahorros Programados", Text =
"Ahorros Programados" },
```

```
                        new SelectListItem { Value="Cuenta Progresar Clientes",
Text = "Cuenta Progresar Clientes" },
                        new SelectListItem { Value="Cuenta Ahorro Progresar Yo
Soy", Text = "Cuenta Ahorro Progresar Yo Soy" },
                        new SelectListItem { Value="Cuenta Corporativa
Progresar", Text = "Cuenta Corporativa Progresar" },
                        new SelectListItem { Value="Todos", Text = "Todos" },

                    }))
                </td>
            </tr>
        </table>
        <br />

        <table>
            <tr>
                <td>Fechas:
                </td>
                <td>
                @Html.CheckBoxFor(model => model.FiltroFechas, new { id =
"chkFiltroFechas" })
                </td>
            </tr>
        </table>
        <div id="divFechas">
            <table>
                <tr>
                    <td>Desde:</td>
                    <td>
                        @Html.Kendo().DatePickerFor(model =>
model.FechaDesde).Start(CalendarView.Year).Depth(CalendarView.Year).Format("yyyy
- MM").Footer(false)
                    </td>
                    <td>Hasta:</td>
                    <td>
                        @Html.Kendo().DatePickerFor(model =>
model.FechaHasta).Start(CalendarView.Year).Depth(CalendarView.Year).Format("yyyy
- MM").Footer(false)
                    </td>
                </tr>
            </table>
        </div>
        <br />
        <table>
            <tr>
                <td>Edad:
                </td>
                <td>
                @Html.CheckBoxFor(model => model.FiltroEdad, new { id =
"chkFiltroEdad" })
                </td>
            </tr>
        </table>
        <div id="divEdad">
            <table>
                <tr>
                    <td>Desde:
                    </td>
                    <td>
                        @( Html.Kendo().DropDownListFor(model => model.EdadDesde)
```

```
                            .BindTo(new List<SelectListItem> {
                            new SelectListItem { Value="", Text = "" },
                            new SelectListItem { Value="20", Text = "20" },
                            new SelectListItem { Value="25", Text = "25" },
                            new SelectListItem { Value="30", Text = "30" },
                            new SelectListItem { Value="35", Text = "35" },
                            new SelectListItem { Value="40", Text = "40" },
                            new SelectListItem { Value="45", Text = "45" },
                            new SelectListItem { Value="50", Text = "50" },
                            new SelectListItem { Value="60", Text = "60" },
                            new SelectListItem { Value="60", Text = "60" },

                        }))
                    </td>

                    <td>Hasta:
                    </td>
                    <td>
                        @( Html.Kendo().DropDownListFor(model => model.EdadHasta)
                            .BindTo(new List<SelectListItem> {
                            new SelectListItem { Value="", Text = "" },
                            new SelectListItem { Value="20", Text = "20" },
                            new SelectListItem { Value="25", Text = "25" },
                            new SelectListItem { Value="30", Text = "30" },
                            new SelectListItem { Value="35", Text = "35" },
                            new SelectListItem { Value="40", Text = "40" },
                            new SelectListItem { Value="45", Text = "45" },
                            new SelectListItem { Value="50", Text = "50" },
                            new SelectListItem { Value="60", Text = "60" },
                        }))
                    </td>
                </tr>
            </table>
        </div>
        <br />
        <table>
            <tr>
                <td>Sexo:
                </td>
                <td>
                @Html.CheckBoxFor(model => model.FiltroSexo, new { id =
"chkFiltroSexo" })
                </td>
            </tr>
        </table>
        <div id="divSexo">
            <table>
                <tr>
                    <td>
                        @Html.RadioButtonFor(model => model.Sexo, "Masculino",
new { Checked = "checked" }) Masculino
                        <br />
                        @Html.RadioButtonFor(model => model.Sexo, "Femenino")
Femenino
                    </td>
                </tr>
            </table>
        </div>
        <br />
        <table>
```

```
            <tr>
                <td>
                    <input class="button" type="submit" value="Generar Reporte"
id="GenerarReporte"
onclick="submitForm('@Url.Action("GenerarRptIndSaldoApertura", "Cumplimiento")
', this);"/>
                    <input class="button" type="button" value="Limpiar"
id="Limpiar"
onclick="submitFormNoValidation('@Url.Action("ReptIndicadorSaldoApertura",
"Cumplimiento")    ', this)"/>
                </td>
            </tr>
        </table>
        <br />
        <div id="divChartIngresadosSucursal"
style="display:@(Model.MostarChartIngresadosSucursal ? string.Empty : "none")">
            <table>
                <tr>
                    <td>

@(Html.Kendo().Chart<Denarius.CoreBanking.Shared.Cumplimiento.Model.DatosApertura
>()
                            .Name("chart")
                            .Title("Indicador Saldo Apertura")
                            .Legend(legend => legend
                                .Visible(true)
                            )
                            .ChartArea(chartArea => chartArea
                                .Background("transparent")
                            )
                            .DataSource(db => db
                                .Read(read =>
read.Action("MostraIndicadorCliente", "Cumplimiento"))
                                .Group(group => group.Add(model =>
model.Sucursal))
                            )
                            .Series(series =>
                            {
                                series.Column(model =>
model.SaldoApertura).Name("Saldo Apertura").Color("#bb6e36");
                            })
                            .CategoryAxis(axis => axis
                                .Name("series-axis")
                                .Line(line => line.Visible(false))
                             )
                             .CategoryAxis(axis => axis
                                 .Categories(model => model.NombreMes)
                                 .Labels(labels => labels.Rotation(-90))
                             )
                             .ValueAxis(axis => axis.Numeric()
                                 .Labels(labels => labels.Format("{0:N0}"))
                                 .MajorUnit(25)
                             )
                            .Tooltip(tooltip => tooltip
                                .Visible(true)
                                .Template("#= series.name #: #= value #")
                             )
                        )
                    </td>
                </tr>
```

```
            </table>
        </div>

        @Html.MensajeFor(model => model.Mensaje)
        @Html.HiddenFor(model => model.Sucursal)
        @Html.HiddenFor(model => model.SubProducto)
        @Html.HiddenFor(model => model.MostarChartIngresadosSucursal)

       </text>
    }
</div>
<script type="text/javascript">

    if ($('#chkFiltroFechas').is(":checked")) {
        $('#divFechas').show();
    }
    else {
        $('#divFechas').hide();
    }

    if ($('#chkFiltroSexo').is(":checked")) {
        $('#divSexo').show();
    }
    else {
        $('#divSexo').hide();
    }

    if ($('#chkFiltroEdad').is(":checked")) {
        $('#divEdad').show();
    }
    else {
        $('#divEdad').hide();
    }
</script>
```

Controlador

```
        #region Saldo Apertura
        public ActionResult ReptIndicadorSaldoApertura()
        {
            ConsultaIndicadoresView indicadoresView = new
ConsultaIndicadoresView();
            indicadoresView.MostarChartIngresadosSucursal = false;
            indicadoresView.FechaDesde = DateTime.Now;
            indicadoresView.FechaHasta = DateTime.Now;
            indicadoresView.TipoLocalizacion = true;
            Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", null);
            return PartialView("_ConsultaSaldoApertura", indicadoresView);
        }

        public PartialViewResult
GenerarRptIndSaldoApertura(ConsultaIndicadoresView consulta)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    var tenant = Sesion.Usuario.TenantUsr;
```

```
                    string estado = "";
                    DateTime? fechaDesde = null;
                    DateTime? fechaHasta = null;
                    string sucursal = "";
                    string tipoSubProducto = "";
                    string sexo = "";
                    int? edadDesde = null;
                    int? edadHasta = null;

                    estado = consulta.Estado;
                    sucursal = consulta.Sucursal;
                    tipoSubProducto = consulta.SubProducto;

                    if (consulta.FiltroFechas)
                    {
                        fechaDesde = consulta.FechaDesde;
                        fechaHasta = consulta.FechaHasta;
                    }

                    if (consulta.FiltroEdad)
                    {
                        edadDesde = consulta.EdadDesde;
                        edadHasta = consulta.EdadHasta;
                    }

                    if (consulta.FiltroSexo)
                        sexo = consulta.Sexo;

                    List<DatosApertura> datos = new List<DatosApertura>();
                    datos = CumplimientoAgent.ConsultaSaldoPromedio(estado,
sucursal, null, tipoSubProducto, fechaDesde, fechaHasta, edadDesde, edadHasta,
sexo);

                    if (datos != null && datos.Count > 0)
                    {
                        consulta.MostarChartIngresadosSucursal = true;

Sesion.SetInSession<List<DatosApertura>>("ListaDatosCliente", datos);
                        //consulta.Reporte = new ReportViewerView();
                        //consulta.Reporte.ReporteBook =
CumplimientoAgent.GenerarReporteProcesoAperturaCtas(datos, tenant);
                    }

                    else
                    {
                        consulta.Mensaje = new
Denarius.CoreBanking.Shared.Model.Mensaje("Indicadores - Saldo Apertura", "No se
encuentraron registros.", tipoMensaje: TipoMensaje.Informacion);
                        consulta.MostarChartIngresadosSucursal = false;
                    }
                }
                catch (Exception ex)
                {
                    string mensaje = new
Error(Denarius.Shared.Error.DevolverExcepcion(ex)).MensajeSimple;
                    consulta.Mensaje = new Mensaje("Error al Consultar
Indicadores", mensaje, tipoMensaje: TipoMensaje.Error);
                }
            }
            return PartialView("_ConsultaSaldoApertura", consulta);
```

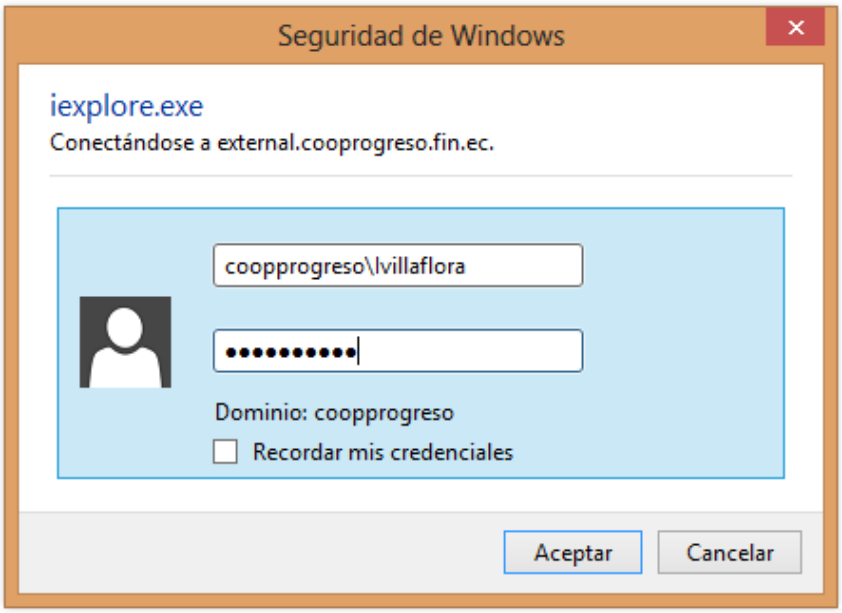
```
        }
        #endregion
```

Anexo 10: Ingreso al ambiente de pruebas.

a. Se ingresa a la página: https://cloud.denariusonline.com/web/login/cooprogreso, donde se muestra la siguiente página:

Dar click sobre la opción **Vaya a este sitio web (no recomendado)**

b. En el siguiente paso muestra la pantalla de autentificación como se muestra en la siguiente figura, se debe ingresar el usuario (coopprogreso\lvillaflora) y la contraseña (Progreso11):

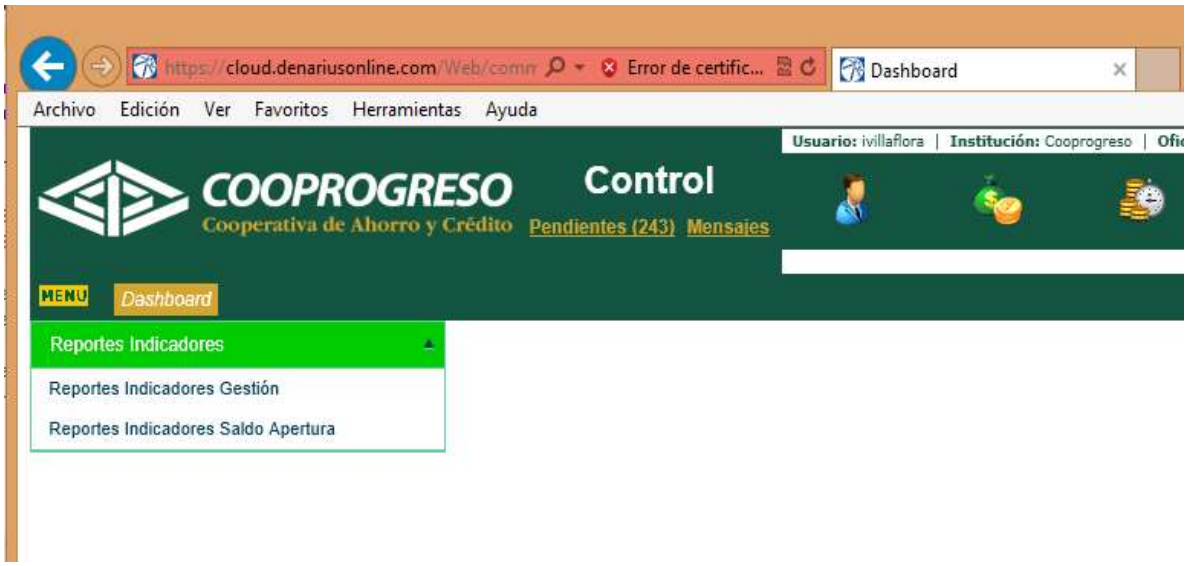


Dar click en aceptar.

c. Al ingresar a la página en la parte superior derecha nos muestra el menú principal: Clientes, Cuentas, Crédito, Inversiones, Contabilidad, BackOffice, Cumplimiento y Agregados, como se muestra en la siguiente figura:

Dar click en el menú cumplimiento.

d. Se desplegará un menú en la parte izquierda, donde nos muestra los submenús, como se muestra en la siguiente figura:



Se muestra en pantalla el menú Reporte Indicadores, conjuntamente con los submenús: Reportes Indicadores Gestión y Reportes Indicadores Saldo Apertura.

e. Se da click sobre cualquiera de los dos submenús y se desplegarán los filtros correspondientes del indicador.

Finalmente se muestran las pantallas de consultas iguales a la presentación del demo en la sección **2.5.2.4**

5.1 Glosario

BI.- Business Intelligence (inteligencia empresarial)

ETL.- Extracción, transformación y carga es un proceso periódico que se encarga de extraer información de las diferentes fuentes de datos con el fin de transformar y cargar la data en los modelos definidos en el Data Warehouse o Datamart.

Core.- Es el negocio desarrollado por una institución financiera.

TFS. - Team Foundation Server.

DW. - Data WareHouse

IT.- Information Technology (tecnología de la información)

Base de datos transaccional.- Tienen una gran cantidad de demanda de acceso a los registros de la base de datos ya sea para lectura o escritura.

Base de datos analítica.- Tienen una bajo e inexistente transacciones, está orientada a procesos analíticos.

Deploy.- Desplegar es la unificación del código que se pasa de un ambiente a otro.

Azure.- Es un servicio que Microsoft ofrece en la nube.

NET Framework. - Componente de Windows que admite la ejecución de aplicaciones**.**

IIS. - Internet Information Services, servicios para servidores de Microsoft.

GHz.- El gigahercio (GHz) es un múltiplo de la unidad de medida de frecuencia hercio (Hz) y equivale a 10^9 (1 000 000 000) Hz.

GB.- Un gigabyte es una unidad de almacenamiento de información cuyo símbolo es el GB, equivalente a 10^9 (1.000.000.000 -mil millones-) de bytes.

RPM.- Significa las revoluciones por minuto del disco duro de un computador, puede ser de lectura o escritura.

DirectX.- es una colección de API desarrolladas para facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y vídeo, en la plataforma Microsoft Windows.

SP.- Service Pack**,** es un conjunto de programas informáticos que consisten en un grupo de actualizaciones que corrigen y mejoran aplicaciones y sistemas operativos.

MB.- El Megabyte es una unidad de medida de cantidad de datos informáticos

AMD.- es el segundo proveedor de microprocesadores basados en la arquitectura x86 y también uno de los más grandes fabricantes de unidades de procesamiento gráfico.

EM64T.- Es un procesador que soporta sistemas operativos de 64 bits.

HTML.- es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet.

Sharepoint.- Es una plataforma de colaboración empresarial.

PowerShell.- Es una consola de sistema, un terminal o "CLI" bastante más avanzado y completo que MS-DOS o CMD desde el que podremos configurar completamente un equipo informático.

Xaml.- Es el lenguaje de formato para la interfaz de usuario para la Base de Presentación de Windows.

Data Warehouse.- Almacén de datos, se entiende como una base de datos que presenta información integrada y depurada de distintas fuentes de datos, en donde se realizarán los respectivos procesos que permita el análisis y consumo de la información por parte de la organización. Su principal enfoque es el de manejar la información en estructuras.

Data Marts.- Es un subconjunto temático de datos, orientados a un proceso o un área de negocio específica

MVC.- Modelo – Vista - Controlador (patrón de arquitectura)

SP.- Store Procedure