**Universidad Tecnológica Naciona**l

*Facultad Regional Córdoba*

Ingeniería en Sistemas de Información

Cátedra de Programación de Aplicaciones Visuales I

Documento de Arquitectura: Proyecto Agronegocios

Integrantes del Equipo (Grupo 9):

* Facundo Paz Fessia
* Nicolás Mingorance
* Neuen Actis

Métodos para compartir la Base de Datos y el Código

Para compartir la Base de Datos y el código de la solución se utilizó la herramienta de versionado de código y gestión de repositorios Github. Se utilizó SQL Server para generar la Base de Datos y se gestionó a través de la herramienta Microsoft SQL Server Management Studio. El script generador de la Base de Datos es el documento que se guarda en el repositorio Git y que cada integrante del grupo tiene disponible para generar la Base de Datos en su Computadora.

El código de la solución queda versionado también en el repositorio git y también queda accesible para cualquier integrante del grupo.

Acuerdos de Interfaz y Programación

Interfaces de ABMC:

Las interfaces para el manejo de la alta, baja, modificación y consulta, se componen, en una mitad de la pantalla, de una grilla que lista los objetos y los filtros para consultar dichos objetos, y del otro lado de la pantalla están los datos que se van a ingresar o modificar del objeto seleccionado en la grilla que se activa una vez presionado o el botón de “Nuevo” o “Modificar”.

Dentro del panel derecho se encuentran los botones “Aceptar” , “Limpiar”, “Finalizar”. Está pensado principalmente para una carga serial de datos, en donde se pueden cargar varios objetos seguidos. El botón “Aceptar” confirma la carga de un nuevo objeto o la modificación dependiendo si se seleccionó “Nuevo” o “Modificar”.

Por último se tiene en la parte inferior de la pantalla los botones “Nuevo” (Permite ingresar un nuevo objeto), “Modificar” (Permite modificar algún dato de un objeto seleccionado), “Eliminar” (Elimina el objeto seleccionado), y “Cerrar” (Cierra el formulario).

Para estas funcionalidades las clases Interfaz de Usuario (es decir los formularios), controlan el flujo de mensajes. Realizan las validaciones necesarias a los campos ingresados por el usuario y tienen referencias a objetos de la Capa de Lógica de Negocio y a Objetos que permiten encapsular los datos ingresados.

Las interfaces de ABMC implementadas son para Clientes, Transportes, Semillas y Empleados.

Interfaces de Transacciones (Facturación y Envíos):

Ambas interfaces tienen un diseño de tipo secuencial, es decir, los campos se van habilitando conforme se ingresan los datos. Ambas interfaces comienzan por buscar al Cliente por medio de su número de CUIT o CUIL. En caso de no encontrarlo se informa al usuario. Para el resto del diseño de la interfaz difieren en los datos necesarios para mostrar, ingresar o seleccionar. Para tipos de datos seleccionables se utilizaron campos de tipo ComboBox o DateTimePicker y para los datos ingresados desde teclado, se utilizaron campos de tipo TextBox. Para mostrar datos en general se utilizó Grillas.

Para la facturación, al igual que con los ABMC, es la interfaz la que controla el flujo de control y que tiene las referencias a objetos de la capa de Lógica de Negocio, referencias a objetos en sí, que encapsulan la información necesaria. Para el envío excepcionalmente se utilizó una clase GestorEnvio que maneja el flujo de control.

Las interfaces de Transacciones implementadas son la Facturación y la Asignación de Envíos.

Interfaces de Reportes y Estadísticas:

En la parte superior de la interfaz, con un Report Viewer se visualiza el Reporte o Estadística a generar y en la parte inferior están los campos que permiten filtrar la información que se va a generar en el Reporte. Como mínimo se debe ingresar las fechas límites del período que se quiere tener información.

Al igual que en los formularios de ABMC, es la interfaz la que maneja el Flujo de control y que realiza la llamada a los objetos en la capa de Lógica de Negocio.

Las interfaces de Reportes implementadas son: Antigüedad de Empleados, Detalles de Factura, Envíos por Empresa, Envíos por Provincia, Facturas por Cliente y Ventas por Empleado. Las interfaces de Estadísticas implementadas son: Top Clientes que más compraron en precios y en cantidad, Envios por empresa, Porcentaje de Ventas por Provincia, Ventas por Empleado y Cantidad de Semillas más vendidas.

Interfaces Principal y de Login:

La interfaz principal está compuesta de un MenuStrip y muestra el logo del sistema. El MenuStrip tiene el acceso a todos los formularios anteriormente mencionados y el Login (es el primer formulario que muestra el sistema) por medio de dos campos, nombre de usuario y contraseña, permite hacer la validación de el usuario que quiera ingresar al sistema.

Acuerdos Generales de Programación:

Se utilizó un framework de programación en capas con el objetivo de tener un sistema más modularizado dividido en capas de: Presentación (GUILayer), Lógica de Negocios (BusinessLayer) y Acceso a Datos (DataAccessLayer). Para las distintas funcionalidades implementadas, la capa de presentación se encarga de todo lo relacionado al control de mensajes, la capa de Lógica de Negocios se encarga de llevar el control para los datos que se quieran recuperar de la base de datos y la capa de acceso a datos se encarga de hacer las consultas e inserciones a la base de datos.

La Base de Datos utilizada es de tipo Relacional, manejada con sentencias SQL.

Aprendizajes

* Desarrollar un proyecto con el entorno Visual Studio, mediante el lenguaje C#

Creación de un nuevo proyecto de escritorio, asignación de nombre de proyecto con su Framework específico.

* Creación de Base de Datos

Se ejecutaron sentencias en SQL Server Management Studio para la creación de BD, con sus respectivas Claves primarias y foráneas.

* Conexión de Base de datos

Se creó una Clase dentro del proyecto denominada Data Manager, donde se usaron funciones de la biblioteca System.Data.SqlClient y asignamos a una variable nuestra cadena de conexión de la BD local. En nuestro caso en especial tuvimos una evolución con respecto a este tema, donde primero para el ingreso a nuestro sistema de software utilizamos una clase DBHelper y luego agregamos la mencionada anteriormente que se usa para el control de transacciones.

* Creación de Formularios

Agregamos a nuestro proyecto un nuevo elemento Formulario de tipo Windows Form, y diseñamos a partir de Herramientas, Propiedades y código la Interfaz y funcionalidades de nuestro sistema de Software.

* Altas, Bajas, Consultas de datos de un formulario

Se programó las Altas, Bajas y Modificaciones de datos del sistema mediante el cual las aplicaciones de bases de datos se mantienen actualizadas.

* Transacciones

Se programaron las Transacciones, donde generamos o modificamos a partir de este evento ciertos datos que se encuentran almacenados en nuestro sistema de información.

* Informes, Estadisticas, Graficos

Se programaron listados de ciertos datos del sistema filtrado por fechas, nombre, id, etc y se proporcionaron gráficos estadísticos que le permiten al usuario visualizar datos cuantitativos. Para esto tuvimos que importar un elemento denominado Reporte, y crear vistas en nuestra BD.

* Desarrollo del Proyecto en Capas

Desarrollamos el software en capas, para poder desacoplar nuestro sistema en en distintos módulos(Capa de Presentación, Capa lógica de Negocio, Capa de Acceso a Datos), de esta manera en el caso de que tengamos alguna modificación, solo afectara al nivel requerido sin tener que hacer cambios en el código fuente de otros módulos.

Problemas y su Solución

* Un problema en la implementación de la facturación fue que era necesario obtener el número de la última factura generada, lo cual al principio se codificó usando MAX(f.numero) lo cual trae problemas si es que se trabaja en múltiples terminales, entonces la solución fue usar @@IDENTITY el cual nos trae el numero de la factura generada en ese momento para posteriormente insertar todos los datos en la factura con ese número.
* Otro problema que tuvimos, fue con el Repositorio git, generalmente teníamos muchos conflictos cuando modificamos al mismo tiempo, la solución era hacer un pull antes de subir la modificación al repositorio y obtener la última versión, o cuando el problema se volvía muy complejo, directamente eliminamos nuestro proyecto y volvíamos a hacer un git clone y bajar el que estaba en la nube. En ciertos casos buscamos comandos nuevos como git Reset para resetear lo que habíamos modificado y volver a una versión anterior.
* Hubo problemas también con el registro de fechas en la base de datos, debido a que esta última no acepta el tipo de dato DateTime por lo que la solución fue pasarlo a tipo String con un formato especial en el orden de los días, meses y años.

Distribución de Roles

En general en el desarrollo de nuestro proyecto no se asignaron roles a cada uno en particular, sino que trabajamos de forma en que tal integrante se encargaba de programar mientras compartía pantalla, y los demás iban aportando ideas para la solución de problemas y el diseño de las funcionalidades.