TRABAJO DE GRADO

SISTEMA DE INVENTARIOS PARA LA ORGANIZACION "LA CAJA DEL AMOR"

ESTEBAN OCHOA MEJIA

Código: 199927540010

UNIVERSIDAD EAFIT

DEPARTAMENTO DE INGENIERIA DE SISTEMAS

MEDELLIN

2006

TRABAJO DE GRADO
SISTEMA DE INVENTARIOS PARA LA ORGANIZACION "LA CAJA DEL AMOR"

ESTEBAN OCHOA MEJIA

Propuesta de proyecto de grado.

Asesor
Juan Guillermo Lalinde Pulido.
Ingeniero de Sistemas.

Código: 199927540010

UNIVERSIDAD EAFIT
DEPARTAMENTO DE INGENIERIA DE SISTEMAS
MEDELLIN
2006

“Solo es útil el conocimiento que nos hace mejores”

AGRADECIMIENTOS

Quiero expresar todo mi agradecimiento a mi familia por todo el esfuerzo que han hecho para que yo haya podido tener una educación, la cual me ha abierto y me seguirá abriendo una cantidad de puertas. Por todo el apoyo que he tenido por parte de ellos en momentos difíciles y la compañía en momentos agradables.

Agradezco también a mi asesor Juan Guillermo Lalinde por guiarme y aclararme dudas en la realización de este proyecto.

Y por último quiero agradecerle a la organización la caja del amor por haberme abierto sus puertas y permitirme trabajar con la población más afectada que tiene la ciudad, los cuales a cambio de mi trabajo me han dado su más sincero afecto, el cual ha tenido, tiene y tendrá un valor invaluable para mi vida, y de donde he aprendido lecciones que de otra manera no hubiera sido posible aprenderlas.

CONTENIDO

INTRODUCCION

La Caja del Amor, una organización sin ánimo de lucro, la cual fue creada en la ciudad de Medellín en el año 2001 y se ha ido expandiendo a otras ciudades como Cali, Bogotá y Barranquilla.

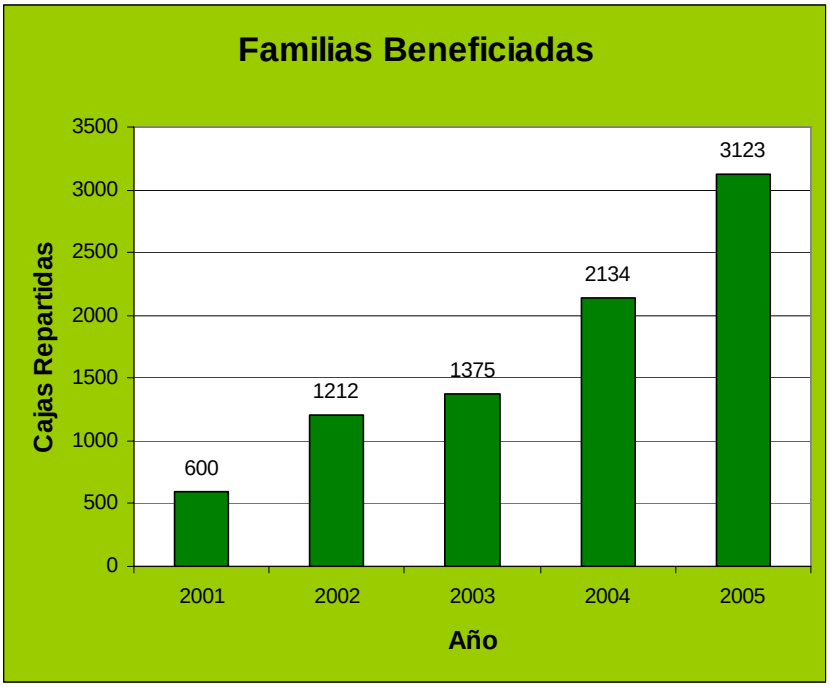
La caja del amor es un regalo que una familia pudiente prepara para una familia de escasos recursos con el objetivo de llevar un mensaje de amor, paz y esperanza durante la época de navidad a estas familias desamparadas.

Su funcionamiento es bastante simple, la familia generosa adquiere una caja con los nombres y edades de las personas de la familia necesitada y la llena con un mercado básico, según sus posibilidades le adicionan juguetes y regalos para las personas asignadas. Una vez hecho esto, la lleva a uno de los centros de acopio y ya posteriormente la organización se encarga de hacerla llegar a la familia previamente seleccionada.

Actualmente opera en los siguientes barrios de Medellín:

- El Edén.
- Villa hermosa.
- 12 de Octubre.
- Iguaná.
- Robledo.
- Mirador de Calasanz.
- Carpinelo.
- La Esperanza.
- Aldea Pablo VI.
- Manrique Oriental.
- Santo Domingo Savio.
- La Estrella.
- Limonar.
- La Gabriela Bello.
- Carambolas.
- Bello Oriente.
- La Avanzada.
- Moravia.

Este gráfico puede ayudar a clarificar un poco más cuál ha sido el impacto que la organización ha tenido en la ciudad de Medellín:



El programa ha ido ganando una aceptación y tomando más popularidad cada año que pasa si se tiene en cuenta que cuando empezó el número de cajas que se vendió fue de 600 y el año pasado (2005) se alcanzó un número de 3123 cajas, traducido en un regalo de navidad para 3123 familias. Esto incrementa todo el trabajo de logística requerido para llevar a cabo el funcionamiento, desde el momento en que se comienza a realizar el censo para ubicar las familias que participarán en el programa por barrio hasta la venta de todas y cada una de las cajas y posteriormente su repartición a cada una de las familias.

Durante los años 2001, 2002, 2003 y 2004 todo este proceso era llevado a cabo manualmente con la ayuda de Excel, teniendo como resultado que información relevante no era centralizada, y donde extraer información era algo tedioso y complicado, ya que sólo era posible hacerlo desde un computador y solamente una persona entendía la manera como tenía organizada la información.

Para el año 2005 no había al interior de la organización una aplicación que se encargara de canalizar todo el flujo de información de una manera centralizada y que permitiera tener un seguimiento completo del estado de cada una de las cajas que estaba en circulación ya que este era bastante precario, lo cual traía como consecuencia en algunas veces la perdida de cajas o entregas tardías.

Por otro lado, el historial que se estaba llevando de las familias, tanto beneficiadas por el programa como de las familias generosas, no era el mejor debido a que el formato de éste hacia bastante complicada su consulta. Sólo estaba disponible en un computador de manera que, si se realizaban cambios en otro equipo, estos sólo se verían reflejados localmente y no en el depósito central de información. Adicionalmente,

la información recogida en años anteriores no está siendo utilizada para las campañas de los años posteriores.

Se planteó la idea de desarrollar un sistema de información basado en Web que le diera solución a este problema, el cual, mediante la ayuda de la codificación por código de barras, haría mucho más fácil todo el proceso de logística y distribución reduciendo tiempos de horas a minutos en cuanto a conteo de inventarios se refiere, al mismo tiempo permitiendo saber la ubicación de una caja en un momento dado, llevando un detallado historial de todas las actividades que se tenían con una familia en particular (familia generosa o de escasos recursos), número de cajas en circulación, en bodega y entregadas.
Al mismo tiempo, se desea hacer posible la venta de estas cajas por Internet teniendo como sistema de pago “Todo1”, el cual permite hacer pagos mediante cuentas CONAVI y BANCOLOMBIA, con el objetivo de masificar la venta de las cajas y no limitar la venta de estas a solo ciudades como Medellín, Barranquilla, Cali y Bogotá sino el resto de ciudades de Colombia. Es importante aclarar que en el presente trabajo no se presentarán resultados del funcionamiento y/o masificación que pueda tener la venta de cajas por Internet.

Este sistema de información sería de gran ayuda para la organización en su proceso de crecimiento. A medida que pasa el tiempo se adquieren más voluntarios, como lo muestra la gráfica se adquiere más participación de familias en el programa y por consiguiente se incrementa la dificultad de llevar un seguimiento detallado de todas las actividades de logística y manejo de información.

Es importante llevar un registro de la información relativa a cada una de estas cajas en el sistema, como la fecha de compra de la caja, número de serie, familia destinataria (integrantes y edades de cada miembro), familia que colaboró con esta, etc. De la misma manera, así como los datos generales de las cajas son importantes, también lo son los datos detallados.

Se planteo que este fuera desarrollado bajo un modelo de 3 capas (Interfaz gráfica, lógica de negocio y base de datos) utilizando como herramienta principal C#, ASP.NET 2.0, SQL Server 2000 y como herramienta principal Visual Studio 2005. Con el fin de acelerar un poco mas el desarrollo de la herramienta se decidió trabajar con unos tipos de objetos de Visual Studio .Net llamados DataSet´s que básicamente hacen un mapeo de la base de datos en clases para que posteriormente el desarrollador haga uso de estos.

OBJETIVOS

Este proyecto tiene por objetivo principal ayudar a las personas menos favorecidas y de escasos recursos no solo al momento de entregar este proyecto sino por muchos años más, es por esto que se le dio enfoque social apoyando a la organización sin ánimo de lucro “La Caja del Amor”.

- Objetivo General: Apoyar el proceso de manejo de inventarios con una aplicación basada en Web.

    - Objetivo Específico: Desarrollar un sistema que facilite parte del proceso de manejo de inventarios (En este caso cajas) al interior de la organización “La Caja del Amor” el cual permita conocer el estado de todas y cada una de las cajas mediante un sistema de código de barras.

- Objetivo General: Masificar la venta de cajas mediante la creación de una página que permita el pago de estas mediante la entidad de pagos TODO1.

    - Objetivo Específico: Desarrollar un sistema Web que promueva y masifique la venta de cajas del amor a través de Internet haciendo uso del módulo virtual de pagos de CONAVI y BANCOLOMBIA.

- Objetivo General: Crear un framework de trabajo que encapsule ciertos servicios requeridos para la elaboración de este sistema con el fin de facilitar y agilizar el desarrollo de este sistema web.

- Objetivo Específico: El framework a crear contará con los siguientes servicios: Acceso a datos, manejo de errores y grabación de mensajes de la aplicación.

ALCANCE

Lo que se busca con el sistema Web a desarrollar es apoyar el proceso de manejo de inventarios mas no sustituirlo. Este estará enfocado principalmente a colaborar en el conteo de cajas en la bodega, permitir saber cual es el estado de una caja específica en un momento en particular, es decir, saber si la caja está en la bodega o si todavía no ha sido entregada por el donante o saber en que estado se encuentra de los que se definan por el usuario (Ejemplo: En transito, entregada, lista para recoger, en centro de acopio.).

Para ello contará con los siguientes módulos:

Módulo Seguimiento Cajas: Proporciona herramientas para realizar un seguimiento detallado de cada una de las cajas del sistema.

También cuenta con un área de impresión de información relativa a cada una de las cajas.

Módulo Reportes: Permite conocer el estado actual de las cajas en circulación filtrando la información por cada uno de los estados definidos por el usuario y también permite generar un reporte de cajas ubicadas por barrio de destino.

Módulo Administrativo: Permite la creación y edición de los siguientes ítems:

-Barrios.

-Cajas.

-Donantes.

-Estados Cajas.

-Familias Beneficiarias.

-Responsables Cajas.

-Ítems tienda virtual.

Este sistema no incluirá en ningún momento planeación de entrega de cajas ni tampoco cálculo de tiempos o vías optimas para la entrega de las mismas en los diferentes barrios que opera la organización “La Caja del Amor”.

El módulo de pagos a través de CONAVI y BANCOLOMBIA incluirá un módulo para manejar los productos a vender donde se podrá añadir nuevos ítems y editar la información de los ya existentes.
Esta fase del proyecto no contará con un generador de informes de ventas, ya que esto es proporcionado por CONAVI y BANCOLOMBIA.

El Framework a crear expondrá los siguientes servicios:
- Acceso a datos.
- Manejo de errores al interior de la aplicación.
- Grabación de mensajes de la aplicación.

Este estará basado en ciertos componentes ya existentes y de alta aceptación en el mercado a nivel industrial como lo son: Data Access Application Block de Microsoft (Componentes de acceso a datos. Para mas información visitar este link: http://www.microsoft.com/downloads/details.aspx?FamilyID=f63d1f0a-9877-4a7b-88ec-0426b48df275&DisplayLang=en) y Log4Net (Componente de logeo de errores, para mas información visitar este link: http://logging.apache.org/log4net/) el cual ha sido creado por la fundación de “Open-Source” Software: Apache.

DESCRIPCION DEL PROBLEMA

A continuación se hace una descripción paso a paso de cómo es el funcionamiento al interior de la organización “La Caja del Amor” para cada uno de los siguientes procesos:

-Creación de censos de familias beneficiarias.

-Venta.

-Recolección de cajas.

-Distribución de cajas.

Creación de censos de familias beneficiarias:

Al interior de la organización hay una fuerte conexión con centros solidarios, centros comunales y de ayuda social en los barrios más marginados de la ciudad los cuales tienen conocimiento de cuales son las familias más necesitadas en cada uno de estos sectores. De acuerdo al número de cajas previstas para cada año, a cada uno de estos centros se le asigna un cupo de familias disponibles para inscribir. Cada centro llena un formato de Excel (Ver Anexo #3) previamente enviado por la organización “La Caja del Amor” con la cabeza e integrantes de cada una de las familias incluyendo las edades, parentescos y sexo de cada uno de los integrantes de la familia. Es importante aclarar que estos formatos existen, pero la forma como estos son diligenciados no siempre es consistente, ya que en algunos casos se escribe en el contenido del campo sexo la palabra “hombre” y en otros casos “masculino”. Lo mismo sucede con los parentescos, edades de niños los cuales algunas veces son dados en meses y otras veces en años. Esto

trayendo como consecuencia la inhabilidad para hacer un buen análisis de la información.

Una vez estos formatos han sido diligenciados con las familias inscritas en el programa, son enviados de regreso en algunos casos por correo electrónico, en otros casos impresos a la oficina central, donde toda la información es consolidada en una sola hoja de cálculo.

Proceso de Venta:

La caja puede ser comprada en un centro comercial, o mediante algunos de los miembros de la caja del amor (Agentes multiplicadores), la cual tiene un precio de $10,000 (Por este valor se hace entrega de una caja de cartón como la que se muestra en la figura 1, la cual tiene el logo impreso de la caja del amor. Esta viene completamente vacía) los cuales tienen por objetivo sostener los cotos de logística.

(Figura 1)

La familia se lleva esta caja vacía para su casa donde la idea es que esta sea llenada en familia (en caso de que se tenga una) con un mercado básico y de acuerdo a los integrantes de la familia y sus edades agregarle regalos para los integrantes.

Al momento de vender una caja, un desprendible es llenado (ver anexo 1), el cual tiene los siguientes campos:

Información Promotor:

- Nombre.
- Teléfono 1.
- Teléfono 2.
- Email.

Información del Donante:

- Nombre.
- Teléfono 1.
- Teléfono 2.
- Email.

Sitio de Acopio:

- Laureles.
- Poblado.
- Santa Mónica.

Tiene un campo extra donde dice si la caja ya fue pagada o se debe y un número de serie de caja.

Proceso de Entrega:

Una vez llenada la caja con el mercado, la familia puede optar por llevar la caja a uno de los diferentes centros de acopio en la ciudad o llamar directamente a la organización para que esta sea recogida dado el caso de que la familia no tenga la posibilidad de llevarla. Cabe anotar que la fecha limite para devolver la caja o llamar a la organización para que esta la recoja es el 1 de octubre de cada año.

Proceso de Inventario:

Una vez la caja es entregada, esta es llevada a una bodega central (La cual generalmente ha sido el colegio “San José de las Vegas”) donde se hace al final de cada día un conteo manual y en una hoja impresa con el listado de todas las cajas se tachan las que ya se encuentran allí. Con base en esto se hace un listado para ver que cajas faltan por entregar y cuales faltan por recoger. Este listado sirve para planificar las actividades y labores del día siguiente.

Proceso de Distribución:

Una vez las cajas se tienen en la bodega y se han clasificado por barrios (destino final), éstas son recogidas por unos camiones y llevadas a los diferentes barrios, donde posteriormente serán entregadas. Cabe anotar que la salida de cada una de las cajas no es registrada en ninguna parte.

ANALISIS DEL PROBLEMA

A continuación se presenta un análisis que se realizó sobre cada proceso donde se listan las principales falencias que presenta:

Censo de familias:

La organización tiene un formato para la recolección de censos de familias lo cual es bueno (Ver anexo 3), el problema está en que la información aquí recolectada no está estandarizada, es decir, una persona puede ingresar como sexo la palabra “hombre” y posteriormente ingresar “masculino”, lo mismo puede suceder con la palabra “mujer” y “femenino”, lo cual si no es estandarizado a la hora de sacar un reporte por sexo de las personas que se tienen matriculadas en el programa, este arrojará información falsa. Similar sucede con los parentescos al interior de las familias, unas veces se escribe “padre” ó “papa”, “mama” ó “madre”, “abuelo” ó “abuelito”, “abuela” ó “abuelita”, y en otros casos la información no es suministrada en su totalidad, por ejemplo el campo “edad” es necesario y requerido, ya que este va impreso con el nombre de cada uno de los integrantes en cada una de las cajas con el objetivo de que la familia que está llenando la caja, en caso de que pueda le anexe un regalo para esta persona acorde con su sexo y edad, lo cual trae como consecuencia reportes que no van a ser confiables desde ningún punto de vista.

Otro problema que se identificó aquí, es que el formato de censo muchas veces es enviado impreso (debido a desconocimiento), lo cual para ser solucionado tiene dos posibles caminos: el traslado físico hasta el sitio donde fue impreso el formato y extraerlo del computador ó

digitarlo nuevamente en el listado consolidado. Cualquiera de las dos alternativas presenta un costo no sólo en dinero sino también en tiempo.

Proceso de Venta:

La información de venta de cajas (proveniente de los desprendibles, para mas información ver el anexo 3) es consolidada al final de cada semana, lo que no permite saber con exactitud cuantas cajas se han vendido, cuantas de estas han sido pagadas y cuantas se deben. Luego hay que sumarle a esto que la información solo la tendrá una persona, ya que esta no está centralizada.

La consolidada de la información no siempre es fácil, ya que no existe un formato preestablecido, y la información de las cajas vendidas por cada uno de los miembros de la organización no siempre es entregada en el mismo formato. Adicionalmente al consolidar esta información en una sola hoja de cálculo central, la información tarda mucho tiempo en ser ingresada, lo cual no permite tener reportes precisos y confiables.

Muchas veces se encuentran desprendibles sin diligenciar completamente, como por ejemplo sin ningún teléfono, el cual es un medio indispensable para poder contactar a la familia que tiene la caja ya sea para recogerla o para recordarle la fecha de entrega en caso de que la haya olvidado.

Nuevamente se encuentra como problema la falta de historial de la información de una manera centralizada.

Proceso de Entrega de cajas en los centros de acopio:

Una vez la caja es entregada en algún centro de acopio ésta sólo es reportada a la bodega principal al finalizar la semana que es cuando se pasa un reporte de las cajas que llegaron. Esto no permite tener una información actualizada y confiable de cual es el estado actual de todas las cajas en circulación.

Una vez las cajas son ingresadas a la bodega, es necesario buscar en un listado caja por caja hasta encontrar el número de serie buscado y tacharla, con el fin de indicar que ya está en la bodega. Esto, dado el caso de que entre una gran cantidad de cajas, se vuelve un cuello de botella a parte de que al ser un proceso manual es propenso a errores y por ende la información consultada no es totalmente confiable.

Proceso de Inventario:

Cada día es impreso un listado con todos los números de series de las cajas, luego de esto al finalizar el día se recorren todas las cajas y con base en sus números de serie estas son tachadas de la lista con el fin de obtener cuantas cajas hay en bodega. Se han dado casos donde este conteo se ha tenido que repetir más de una vez debido a que ha habido errores en la cuenta. Esta información es bastante propensa a errores y toma mucho tiempo llevar a cabo el proceso de obtenerla. Por ende los reportes generados no son confiables (Cuantas cajas hay en total, cuantas faltan por recoger, cuantas han salido de bodega).

Dado el caso de que se quiera saber cual es el estado o ubicación de una caja en particular es necesario buscar en el listado principal de la bodega para saber si esta se encuentra allí, en caso contrario llamar a cada centro de acopio y averiguar si esta se encuentra allí, de lo

contrario se concluye de que no ha sido aún entregada. Nuevamente este es un proceso bastante manual, que toma una gran cantidad de tiempo y recursos para llevarla a cabo y que no es preciso.

Proceso de Distribución:

Al salir las cajas de la bodega, éstas son nuevamente tachadas en otra lista completamente distinta a la empleada para el inventario para indicar que ya han salido de la bodega, pero no se registra ni la fecha, ni la hora, ni a quién fue entregada, ni por quien fue entregada cada una de las cajas, por tanto generar reportes a partir de la información es bastante complicado.

El proceso de sacar las cajas de la bodega llega a puntos en que se convierte un cuello de botella, ya que al salir cada caja hay que buscarla en el listado y tacharla, lo cual es un proceso demorado y poco fiable.

SOLUCION PROPUESTA AL PROBLEMA

Luego de haber determinado los principales problemas en el proceso, se identificó cuales serían las principales características requeridas por el sistema de información a desarrollar y a continuación se presenta un listado de las mismas clasificadas por procesos:

Censo de familias:

- Estandarizar el formato requerido para la captura de datos de las familias.
- Estandarizar las posibles respuestas tanto en campos de selección múltiple como en campos de única selección.
- Los campos requeridos para el formato serían los siguientes:
    - Nombre
    - Apellidos
    - Edad
    - Sexo
    - Parentesco
- El ingreso de la información se debería hacer mediante Internet (Esta decisión se tomo teniendo en cuenta de que en todos los centros donde la información es tomada hay acceso a Internet) y se requiere que la información quede almacenada de una forma centralizada.
- Se debe permitir la generación de reportes con base en parámetros ingresados por el usuario.

Proceso de Venta:

- Se encontró la necesidad de que la información debe ser posible consolidarla al menos una vez al día y que esta pudiera ser ingresada simultáneamente por más de una persona. Esta debe permanecer centralizada.
- Se definieron campos requeridos para este proceso, que en este caso serían:
    - Nombre donante.
    - Apellido donante.
    - Teléfono de contacto donante.
    - Estado de la caja (Si se pagó o se quedó debiendo).
    - Nombre y apellidos promotor (Quién vendió la caja).
    - Teléfono promotor.
- Se debe permitir la generación de reportes con base en parámetros ingresados por el usuario.
- Cada caja debe llevar un código de barras que permita un manejo más fácil y rápido a la hora de realizar la venta y hacerle su respectivo seguimiento.
- Para todos estos puntos la información debe quedar centralizada con el fin de poder obtener reportes confiables.

Proceso de Entrega de cajas en los centros de acopio:

- Es necesario que los centros de acopio reporten al menos una vez al día los número de serie de las cajas que han recibido con el fin de poder llevar un control preciso de las mismas. Esta información debe ser posible ingresarla simultáneamente por cada centro de acopio a través de Internet y que resida de una manera

centralizada con el fin de que ésta pueda ser consultada por los distintos miembros del equipo de trabajo.

- Debe ser posible consultar el estado de una caja por su número de serie y obtener cual es su ubicación exacta en ese preciso momento.
- Debe ser posible registrar las cajas que entran de una manera rápida y ágil. Se propone usar código de barras para este fin.
- Se debe permitir la generación de reportes con base en parámetros ingresados por el usuario.
- Para todos estos puntos la información debe quedar centralizada con el fin de poder obtener reportes confiables.

Proceso de Inventario:

- Se debe poder hacer un conteo de inventario de una manera ágil y precisa, con el fin de poder saber al fin de cada día cual es el estado de todas las cajas (Código de barras).
- Dado un número de serie es preciso conocer su ubicación de una manera rápida y exacta.
- Para ambos puntos la información debe quedar centralizada con el fin de poder obtener reportes confiables.
- Se debe permitir la generación de reportes con base en parámetros ingresados por el usuario.

Proceso de Distribución:

- Es preciso conocer cuantas cajas han salido, en que fecha lo han hecho, quien gestionó la salida de bodega de cada una de estas y para que barrio van dirigidas.
- El proceso de extracción de cajas de la bodega se debe hacer de una manera rápida y precisa (Código de barras).
- La información debe estar centralizada.
- Se debe permitir la generación de reportes con base en parámetros ingresados por el usuario.

CASOS DE USO

Con el fin de darle un poco más de claridad a la aplicación a desarrollar, se crearon unos casos de uso con base en los puntos anteriormente expuestos. Estos pueden ser vistos en el Anexo #4.

ARQUITECTURA DE LA APLICACIÓN

Toda aplicación desarrollada tiene unas bases, similares a las que se crean al momento de la construcción de un edificio. Una vez creadas, esta construcción tendrá unos planos eléctricos, unos planos de acueducto, de alcantarillado, de iluminación, seguridad, entre otros que les dirán a los constructores como se llevará a cabo la edificación de estos, que juntos al final componen el 100% de la construcción.
Lo mismo sucede entonces con el desarrollo de este sistema Web, el cual a diferencia de los planos de un edificio tiene otros tipos, como por ejemplo: Diagrama de clases, modelo de datos, modelo de base de datos, modelo de acceso a datos, modelo de reporte de errores y mensajes de la aplicación, modelo de seguridad y autenticación.

A continuación se hace entonces una descripción de cada una de las partes que componen toda la arquitectura bajo la cual se desarrollo el presente sistema Web.

Hydra (Framework de Trabajo).

Que pasa si cada vez que un pintor va a dibujar un cuadro utiliza un pincel y oleos totalmente distintos a los usados en el cuadro pintado inmediatamente anterior? O que pasa si un jugador de tenis utiliza una nueva raqueta cada vez que juega? O que pasa si un chef utiliza cada vez que cocina productos de distintas marcas?
No hace falta pensarlo mucho para darse cuenta de que los resultados no van a ser constantes ni similares a través del tiempo, es decir, van a presentar variaciones, en algunos casos no tan notables como en otros, influyendo esto entonces no solo en la calidad del producto sino en todos los factores que componen el desarrollo del mismo.

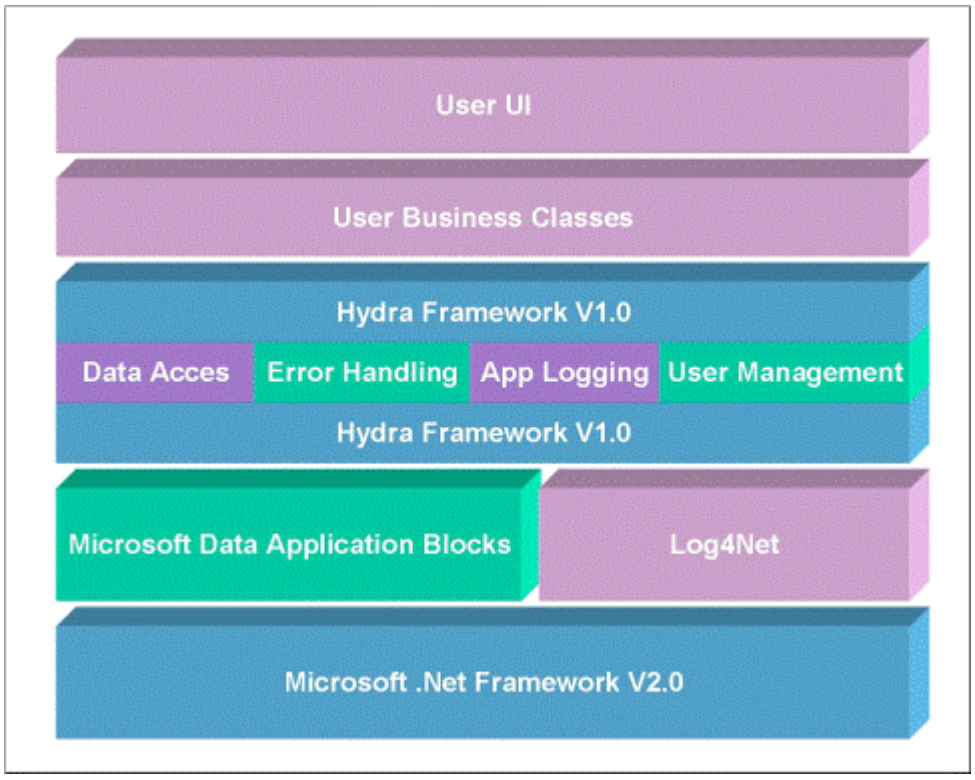
Hydra es una plataforma de desarrollo construida en Microsoft Visual Studio .NET Framework 2.0 enfocada en el desarrollo de aplicaciones web.
Su objetivo principal es facilitar y acelerar el proceso de desarrollo soportando las siguientes áreas:

a. Data Access.
b. Error Handling.
c. Application Logging.

Descripción

Hydra fue desarrollado en c# y se apoya completamente sobre el Framework de Microsoft .NET V2.0, el cual le provee una gran cantidad de funcionalidades y servicios.

Adicionalmente, es soportado por dos assemblies principales los cuales a su vez están completamente soportados por el framework de .Net V2.0, el primero usado para todo lo relacionado con Acceso a Datos (Microsoft Data Application Blocks) el cual es actualmente catalogado por Microsoft como "Best Practices", y el segundo es log4net el cual fue desarrollado por Apache Software Foundation y provee una variada gama de funcionalidades en cuanto a logeo de errores y mensajes de aplicación.



Configuración

Hydra está conformado por 3 DLL´s que son:

1. Framework.Net.dll: Este es el ensamblado resultante luego de la compilación del framework Hydra.

2. Log4net.dll: Este ensamblado provee todas las funcionalidades de logeo al interior del framework y puede ser descargado gratuitamente la siguiente dirección: www.apache.org.

3. Microsoft.ApplicationBlocks.Data.dll: Provee todas las funcionalidades de acceso a datos. Puede ser descargado gratuitamente desde la siguiente dirección: www.gotdotnet.com

Configurando el Logger:

1. Es necesario agregar una referencia a "Framework.Net.dll" en el proyecto donde va a usarse y asegurarse de que una copia de Log4net.dll y Microsoft.ApplicationBlocks.Data.dll se encuentren en la carpeta "Bin" del proyecto.

2. Se debe configurar el archivo web.config donde se especifica cual es la configuración básica de los "loggers" de la siguiente manera:

   <configSections>

```
        <section name="log4net"
        type="log4net.Config.Log4NetConfigurationSectionHandler,
log4net" />
        </configSections>
        <log4net>
              <appender name="LogFileAppender"
type="log4net.Appender.FileAppender">
                    <param name="File" value="logFile.txt" />
                    <param name="AppendToFile" value="true" />
                    <layout type="log4net.Layout.PatternLayout">
                          <param name="Header" value="----------
------------------------------\r\n" />
                          <param name="Footer" value="----------
------------------------------\r\n" />
                          <param name="ConversionPattern"
value="%d [%t] %-5p %c %m%n" />
                    </layout>
                    <filter type="log4net.Filter.LevelRangeFilter">
                          <levelMax value="WARN" />
                    </filter>
              </appender>

        <appender name="ErrorFileAppender"
type="log4net.Appender.FileAppender">
              <param name="File" value="errorFile.txt" />
              <param name="AppendToFile" value="true" />
              <layout type="log4net.Layout.PatternLayout">
```

```
                <param name="Header" value="----------
----------\r\n" />
                <param name="Footer" value="----------
----------\r\n" />
                <param name="ConversionPattern" value="%d
[%t] %-5p %c %m%n" />
            </layout>
            <filter type="log4net.Filter.LevelRangeFilter">
                <levelMin value="ERROR" />
            </filter>
        </appender>

        <appender name="ConsoleAppender"
type="log4net.Appender.ConsoleAppender" >
            <layout type="log4net.Layout.PatternLayout">
                <param name="Header" value="[Header]\r\n"
/>
                <param name="Footer" value="[Footer]\r\n" />
                <param name="ConversionPattern" value="%d
[%t] %-5p %c %m%n" />
            </layout>
        </appender>

        <root>
            <level value="ALL" />
            <appender-ref ref="LogFileAppender" />
            <appender-ref ref="ErrorFileAppender" />
        </root>
```

```
</log4net>
<appSettings>
    <add key="fileLogErrors" value="AppLogError.txt"/>
    <add key="fileLogInfo" value="AppLogInfo.txt"/>
</appSettings>
```

Nota: Este fragmento de código se debe pegar luego de que la directiva <configuration> ha sido declarada.

La configuración inicial aquí mostrada tiene configurados 2 loggers, uno llamado "ErrorFileAppender" el cual logea todos los errores a un archivo con nombre "errorFile.txt" y otro que logea eventos especificados por el programador llamado "LogFileAppender" el cual logea en un archivo con nombre "logFile.txt".

Por defecto, ambos loggers están activados mediante la siguiente directiva:

```
<root>
    <level value="ALL" />
    <appender-ref ref="LogFileAppender" />
    <appender-ref ref="ErrorFileAppender" />
</root>
```

En caso de querer desactivar uno de ellos o ambos, basta con eliminar la línea "<appender-ref ref="LogFileAppender" />" para

el caso del logger encargado de los logs indicados por el desarrollador.

Si se desea cambiar el nombre del archivo en donde esta siendo enviado el log, se puede modificar el nombre del archivo como en las líneas que aparecen abajo:

```
<appender name="ErrorFileAppender"
type="log4net.Appender.FileAppender">
        <param name="File" value="errorFile.txt" />
```

3. Luego de hacer esta referencia, se agrega una referencia en el codebehind de donde se desea usar referenciado el namespace Framework.net de la siguiente manera:

   "using Framework.Net;"

4. Luego de tener configurados los loggers, se crea una instancia de la clase loggers de la siguiente manera en el código donde se quiera usar:

   ILogger _objLogger = new Log4Net(typeof(Tipo de la pagina));

   Ejemplo: El siguiente logger se crea en la pagina con nombre Default.aspx
   ILogger _objLogger = new Log4Net(typeof(_Default));

Luego se invoca alguno de estos métodos, dependiendo del tipo de log que se desee crear:

Estos métodos son considerados log de error:

```
_objLog4Net.fxn_debug("error description goes here as string");
_objLog4Net.fxn_error("error description goes here as string");
_objLog4Net.fxn_fatal("error description goes here as string");
```

Estos métodos son considerados log de información de aplicación:

```
_objLog4Net.fxn_info("App description goes here as string");
_objLog4Net.fxn_warn("App description goes here as string");
```

Configurando el Acceso a Datos:

Actualmente Hydra solo tiene implementado funcionalidades de acceso a datos con SQL Server, para llevarla a cabo se debe de declarar la conexión en el web.config dentro del statement "appSettings" de la siguiente manera:

```
<appSettings>
    <add key="ConnectionString"
value="server=DMG036;database=dbNameHere;uid=sa;password=123
;"/>
</appSettings>
```

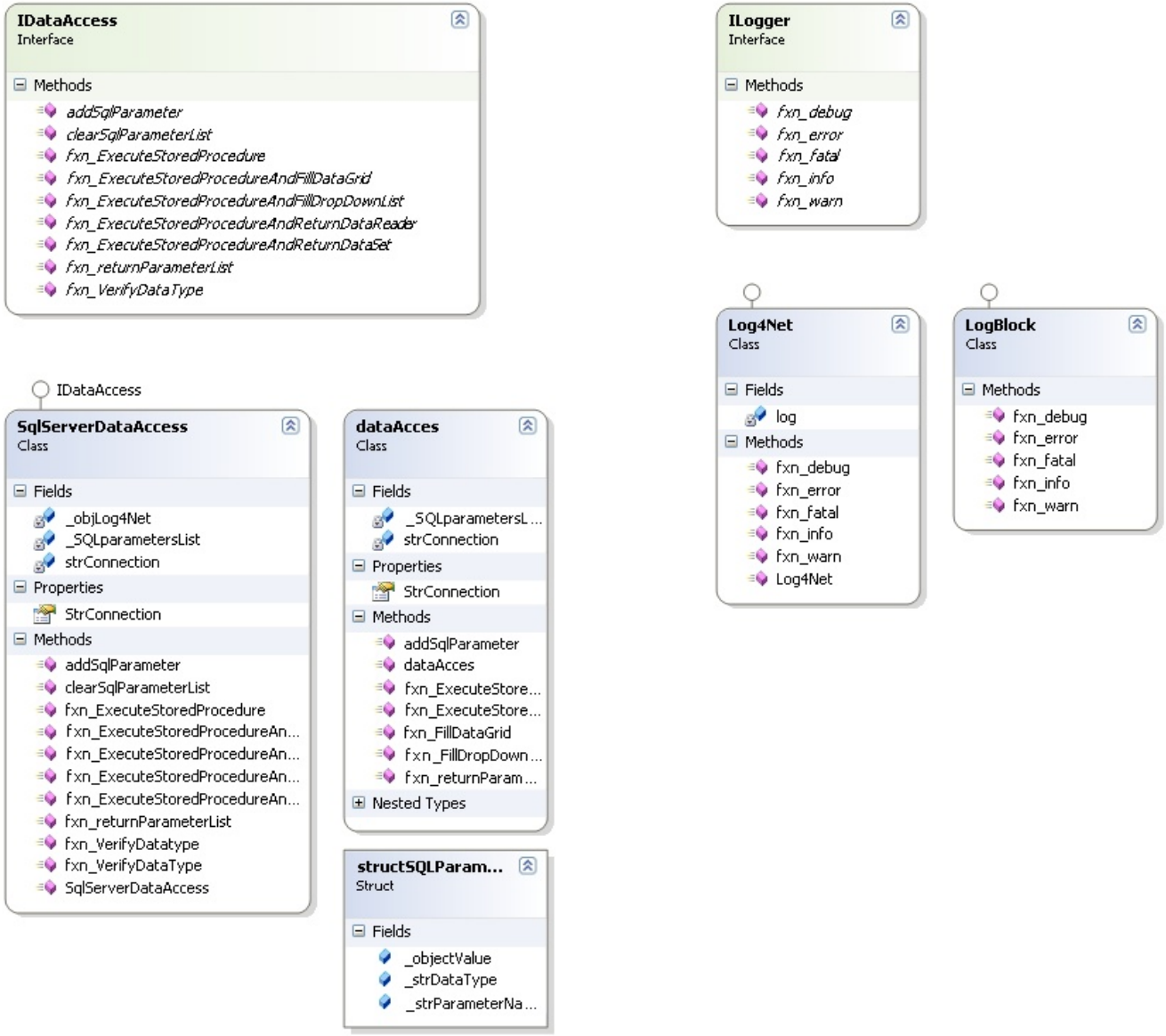
Luego de especificar los parametros de la conexión en el web.config podemos proceder a utilizar las funcionalidades de acceso a datos de la siguiente manera:

Creamos un objeto de la interfaz IDataAccess y de tipo SqlServerDataAccess como se muestra a continuación:

```
IDataAccess _objDataAccess = new SqlServerDataAccess();
```

Una vez hecho esto se puede acceder a todos los métodos que ofrece Hydra de acceso a datos.

A continuación se presenta un diagrama que ilustra la arquitectura empleada para la construcción de este framework:


IDataAccess
Interface
Methods
addSqlParameter
clearSqlParameterList
fxn_ExecuteStoredProcedure
fxn_ExecuteStoredProcedureAndFillDataGrid
fxn_ExecuteStoredProcedureAndFillDropDownList
fxn_ExecuteStoredProcedureAndReturnDataReader
fxn_ExecuteStoredProcedureAndReturnDataSet
fxn_returnParameterList
fxn_VerifyDataType
ILogger
Interface
Methods
fxn_debug
fxn_error
fxn_fatal
fxn_info
fxn_warn
Log4Net
Class
Fields
log
Methods
fxn_debug
fxn_error
fxn_fatal
fxn_info
fxn_warn
Log4Net
LogBlock
Class
Methods
fxn_debug
fxn_error
fxn_fatal
fxn_info
fxn_warn
IDataAccess
SqlServerDataAccess
Class
Fields
_objLog4Net
_SQLparametersList
strConnection
Properties
StrConnection
Methods
addSqlParameter
clearSqlParameterList
fxn_ExecuteStoredProcedure
fxn_ExecuteStoredProcedureAn...
fxn_ExecuteStoredProcedureAn...
fxn_ExecuteStoredProcedureAn...
fxn_ExecuteStoredProcedureAn...
fxn_returnParameterList
fxn_VerifyDatatype
fxn_VerifyDataType
SqlServerDataAccess
dataAcces
Class
Fields
_SQLparametersL...
strConnection
Properties
StrConnection
Methods
addSqlParameter
dataAcces
fxn_ExecuteStore...
fxn_ExecuteStore...
fxn_FillDataGrid
fxn_FillDropDown...
fxn_returnParam...
Nested Types
structSQLParam...
Struct
Fields
_objectValue
_strDataType
_strParameterNa...

Usando el Modelo de Datos de ASP.NET 2.0 en SQL SERVER 2000

Dado el caso de que se quiera usar una base de datos distinta a SQLSERVER EXPRESS para el manejo de membership, roles y demás, se debe correr el siguiente comando contra la base de datos a usar: aspnet_regsql.exe con el fin de que crear las tablas y stored procedures requeridos por el provider de acceso a datos de ASP.NET 2.0.

Luego, se añade una connectionString como en el paso anterior.

Posteriormente se configura el role provider agregando la siguiente línea de configuración en el web.config:

```
<roleManager enabled="true" defaultProvider="SqlRoleProvider">
        <providers>
                <add name="SqlRoleProvider"
type="System.Web.Security.SqlRoleProvider"
connectionStringName="ConnectionString"/>
        </providers>
</roleManager>
```

Y por ultimo, para configurar el Membership, se añade la siguiente línea también en el web.config:

```
<membership defaultProvider="SqlProvider"
userIsOnlineTimeWindow="15">
        <providers>
                <clear/>
```

```
            <add name="SqlProvider"
type="System.Web.Security.SqlMembershipProvider"
connectionStringName="ConnectionString" applicationName="Volar y
Servir" enablePasswordRetrieval="false" enablePasswordReset="true"
requiresQuestionAndAnswer="false" requiresUniqueEmail="true"
passwordFormat="Hashed"/>
        </providers>
</membership>
```

Nota: Ambos fragmentos de configuración van dentro del tag <system.web> </system.web>.

Tener en cuenta de que esta base de datos debe de ser SQL SERVER 2000 ó 2005.

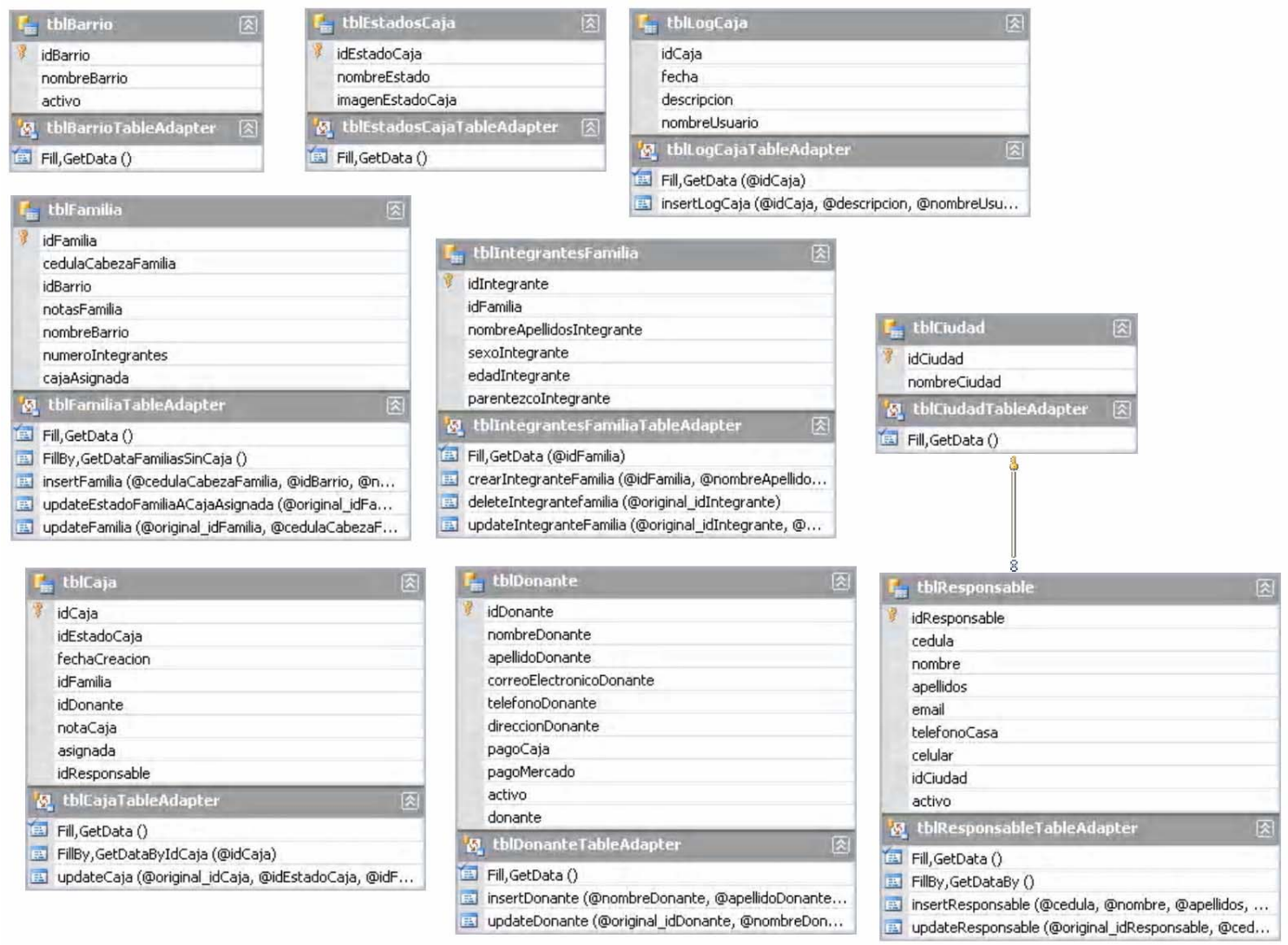
Dado el caso de que también se quiera cambiar el provider de datos para SLQ SERVER 2000 en este caso, basta con añadir las siguientes líneas en el web.config:

```
<profile enabled="true" defaultProvider="ProfileProvider">
        <providers>
            <add name="ProfileProvider"
type="System.Web.Profile.SqlProfileProvider"
connectionStringName="MySqlConnection" />
        </providers>
        <properties>
            <add name="Country" type="string"/>
            <add name="Gender" type="string"/>
```

```
            <add name="Age" type="Int32"/>
        </properties>
</profile>
```

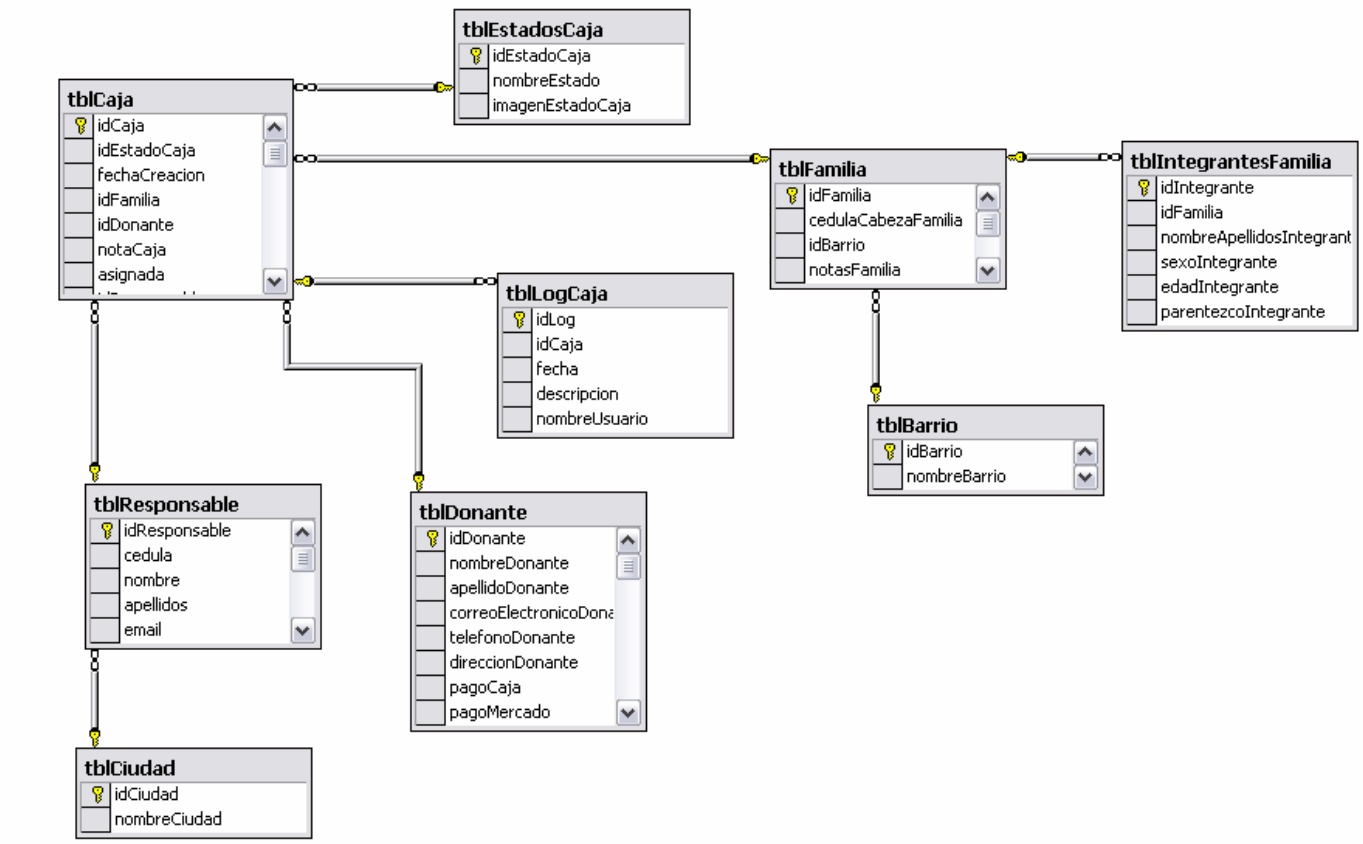
Modelo de Datos (Lógica)

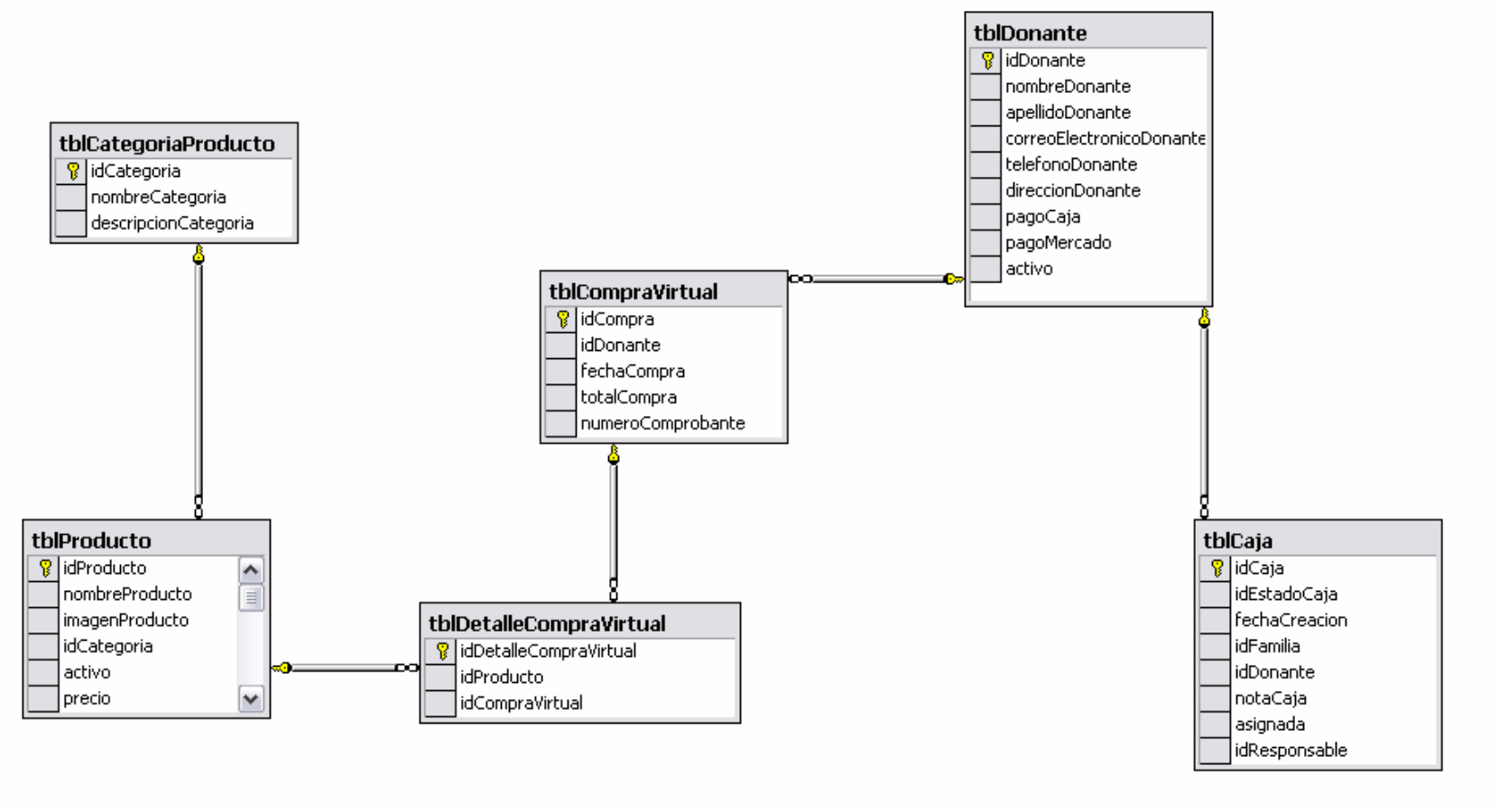
En su gran mayoría, la aplicación fue trabajada con Datasets, un nuevo tipo de dato utilizado por Visual Studio 2005 en donde las clases que antes eran creadas por el desarrollador ahora son generadas por el Framework V 2.0 de Microsoft mapeando directamente a la base de datos con base en las tablas provenientes de la base de datos de la caja del amor lo cual permite agilizar el desarrollo de la aplicación.


tblBarrio
idBarrio
nombreBarrio
activo
tblBarrioTableAdapter
Fill,GetData ()
tblEstadosCaja
idEstadoCaja
nombreEstado
imagenEstadoCaja
tblEstadosCajaTableAdapter
Fill,GetData ()
tblLogCaja
idCaja
fecha
descripcion
nombreUsuario
tblLogCajaTableAdapter
Fill,GetData (@idCaja)
insertLogCaja (@idCaja, @descripcion, @nombreUsu...
tblFamilia
idFamilia
cedulaCabezaFamilia
idBarrio
notasFamilia
nombreBarrio
numeroIntegrantes
cajaAsignada
tblFamiliaTableAdapter
Fill,GetData ()
FillBy,GetDataFamiliasSinCaja ()
insertFamilia (@cedulaCabezaFamilia, @idBarrio, @n...
updateEstadoFamiliaACajaAsignada (@original_idFa...
updateFamilia (@original_idFamilia, @cedulaCabezaF...
tblIntegrantesFamilia
idIntegrante
idFamilia
nombreApellidosIntegrante
sexoIntegrante
edadIntegrante
parentezcoIntegrante
tblIntegrantesFamiliaTableAdapter
Fill,GetData (@idFamilia)
crearIntegranteFamilia (@idFamilia, @nombreApellido...
deleteIntegranteFamilia (@original_idIntegrante)
updateIntegranteFamilia (@original_idIntegrante, @...
tblCiudad
idCiudad
nombreCiudad
tblCiudadTableAdapter
Fill,GetData ()
tblCaja
idCaja
idEstadoCaja
fechaCreacion
idFamilia
idDonante
notaCaja
asignada
idResponsable
tblCajaTableAdapter
Fill,GetData ()
FillBy,GetDataByIdCaja (@idCaja)
updateCaja (@original_idCaja, @idEstadoCaja, @idF...
tblDonante
idDonante
nombreDonante
apellidoDonante
correoElectronicoDonante
telefonoDonante
direccionDonante
pagoCaja
pagoMercado
activo
donante
tblDonanteTableAdapter
Fill,GetData ()
insertDonante (@nombreDonante, @apellidoDonante...
updateDonante (@original_idDonante, @nombreDon...
tblResponsable
idResponsable
cedula
nombre
apellidos
email
telefonoCasa
celular
idCiudad
activo
tblResponsableTableAdapter
Fill,GetData ()
FillBy,GetDataBy ()
insertResponsable (@cedula, @nombre, @apellidos, ...
updateResponsable (@original_idResponsable, @ced...

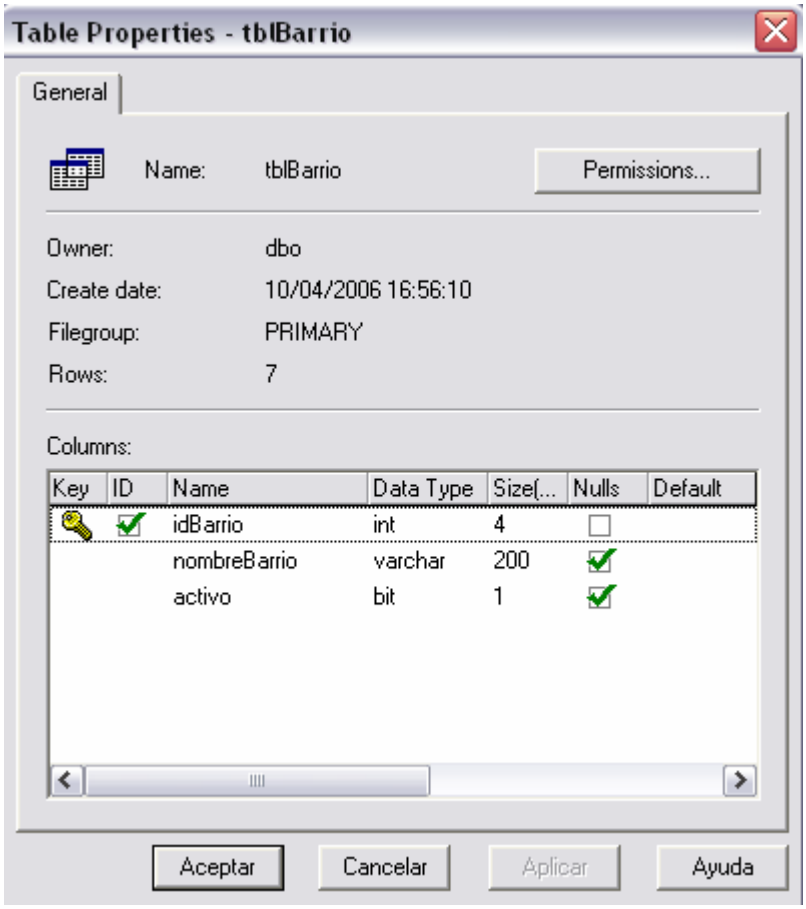
Modelo Base de Datos

A continuación se presenta un diagrama general de cómo están relacionadas las tablas que componen la base de datos, luego se presenta una detallada descripción por cada una de las tablas existentes:




tblCategoriaProducto
idCategoria
nombreCategoria
descripcionCategoria
tblDonante
idDonante
nombreDonante
apellidoDonante
correoElectronicoDonante
telefonoDonante
direccionDonante
pagoCaja
pagoMercado
activo
tblCompraVirtual
idCompra
idDonante
fechaCompra
totalCompra
numeroComprobante
tblProducto
idProducto
nombreProducto
imagenProducto
idCategoria
activo
precio
tblDetalleCompraVirtual
idDetalleCompraVirtual
idProducto
idCompraVirtual
tblCaja
idCaja
idEstadoCaja
fechaCreacion
idFamilia
idDonante
notaCaja
asignada
idResponsable

Descripción detallada: tblBarrio



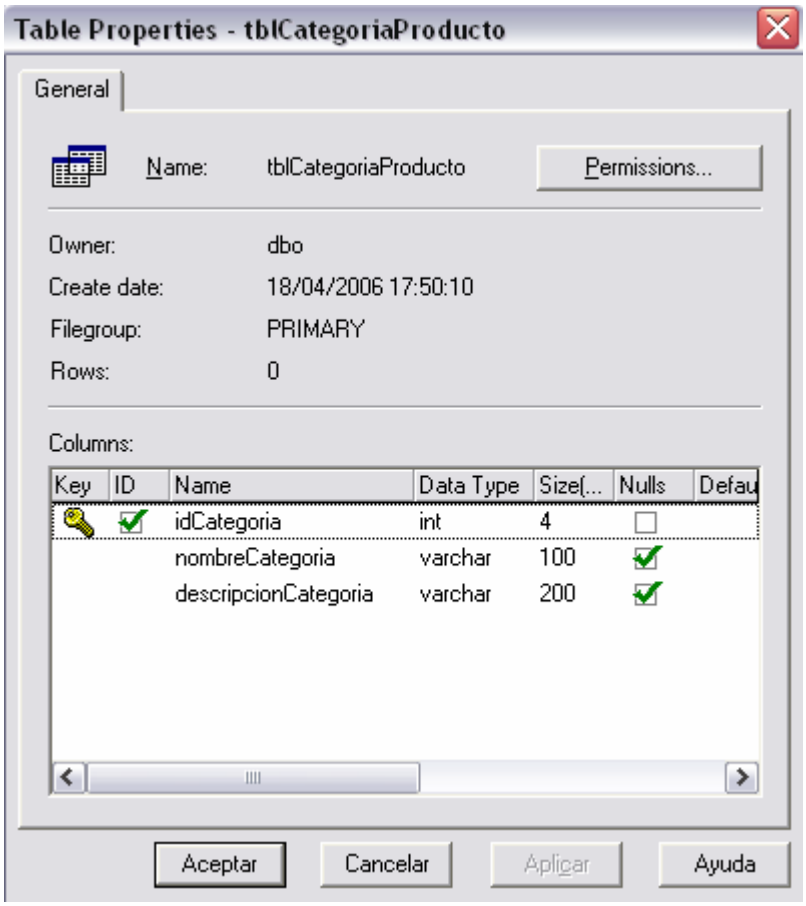
La tabla tiene por objetivo almacenar todos los barrios disponibles en la aplicación y está conformada por un campo idBarrio el cual es un auto numérico para cada registro que se crea en la tabla. Adicionalmente tiene un campo llamado nombreBarrio de tipo varChar con una longitud de 200 caracteres el cual almacenará el nombre del barrio ingresado. Por último encontramos un campo llamado “activo” de tipo bit, el cual dependiendo de su estado indica si el barrio esta como su nombre lo indica activo o inactivo.

Descripción detallada: tblCaja

	Column Name	Data Type	Length	Allow Nulls
🔑	idCaja	bigint	8	
	idEstadoCaja	int	4	✓
	fechaCreacion	datetime	8	✓
	idFamilia	bigint	8	✓
	idDonante	bigint	8	✓
	notaCaja	varchar	1000	✓
	asignada	bit	1	✓
	idResponsable	bigint	8	✓

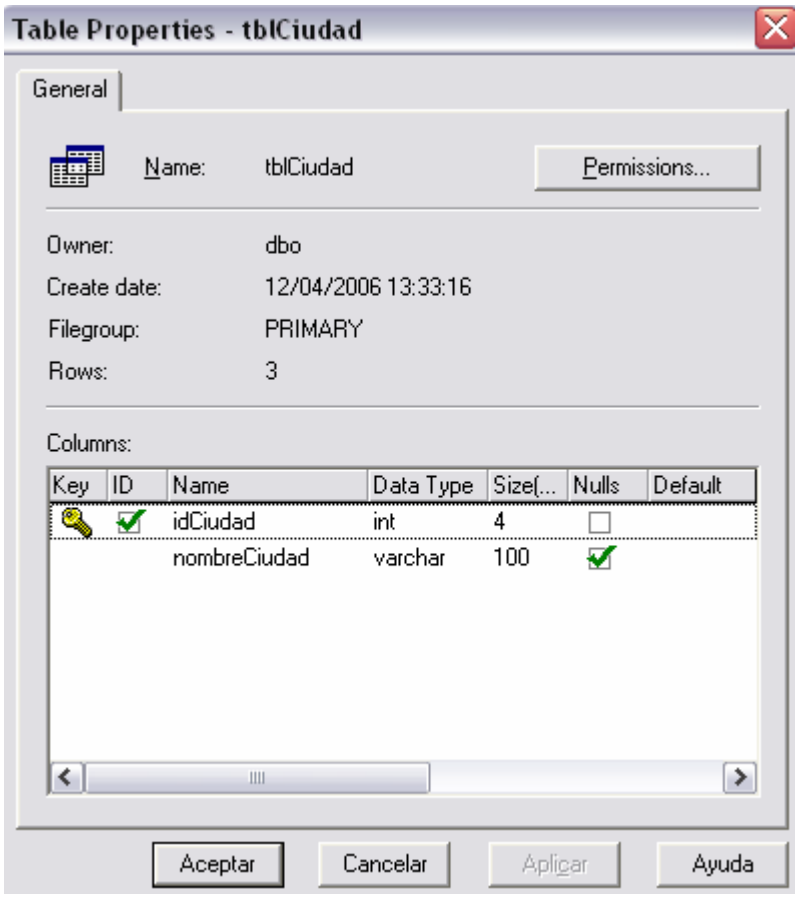
La tabla tiene por objetivo almacenar todas las cajas existentes en el sistema y está conformada por un campo IdCaja el cual es un auto numérico para cada registro ingresado en la tabla el cual servirá a su vez como clave primaria, tiene un idEstadoCaja el cual actúa como clave foránea de la tabla "tblEstadosCaja" la cual será descrita posteriormente. El campo "fechaCreacion" indica la fecha en la cual fue creada la caja, "idFamilia" actúa también como clave foránea de la tabla "tblFamilia" e indica a que familia ha sido asignada esta caja, lo mismo sucede con el campo "idDonante" el cual actúa como clave foránea de la tabla "tblDonante" y especifica a su vez quién fue el donante de esta caja. Luego se encuentra el campo "notaCaja" en el cual se pueden hacer anotaciones relativas a la caja, "asignada" indica si esta caja ya ha sido asignada a una familia dada, y por último encontramos el campo "idResponsable" que actúa como clave primaria de la tabla "tblResponsable" e indica quién es el encargado de la venta de esta caja.

Descripción detallada: tblCategoriaProducto



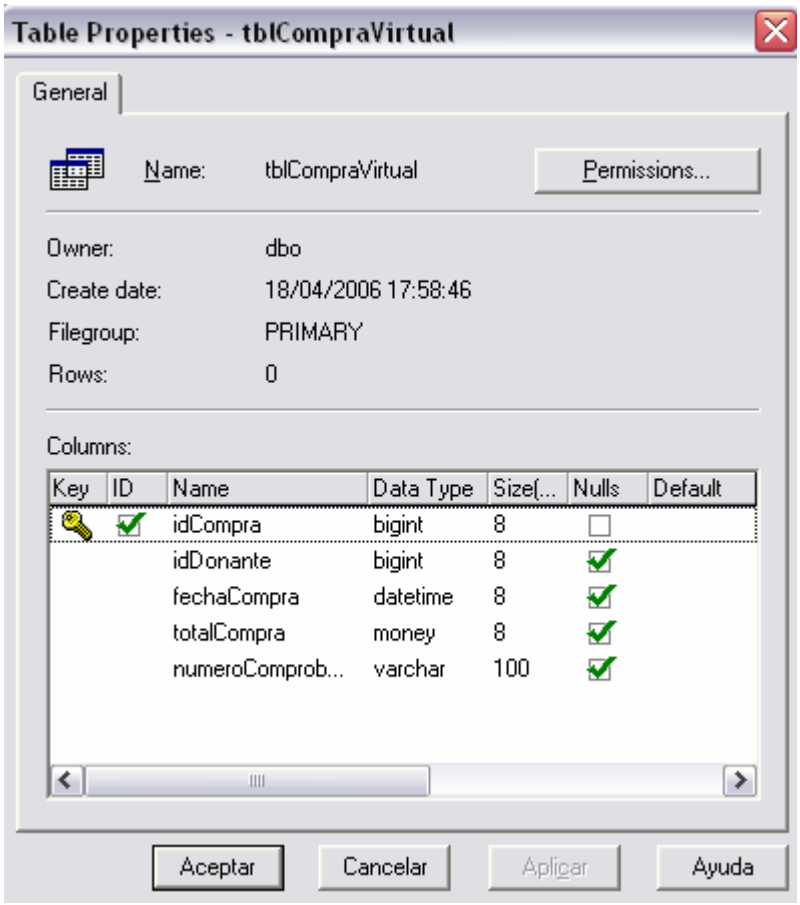
La tabla tiene por objetivo clasificar los productos existentes en la tienda virtual provenientes de la tabla “tblProducto” en diferentes categorías. Está conformada por un campo “idCategoria” el cual es un campo auto numérico y que actúa como clave primaria, “nombreCategoria” como su nombre lo indica lleva el nombre de la categoría y por último encontramos el campo “descripciónCategoria” el cual lleva una descripción de la categoría en caso de que la necesite.

Descripción detallada: tblCiudad



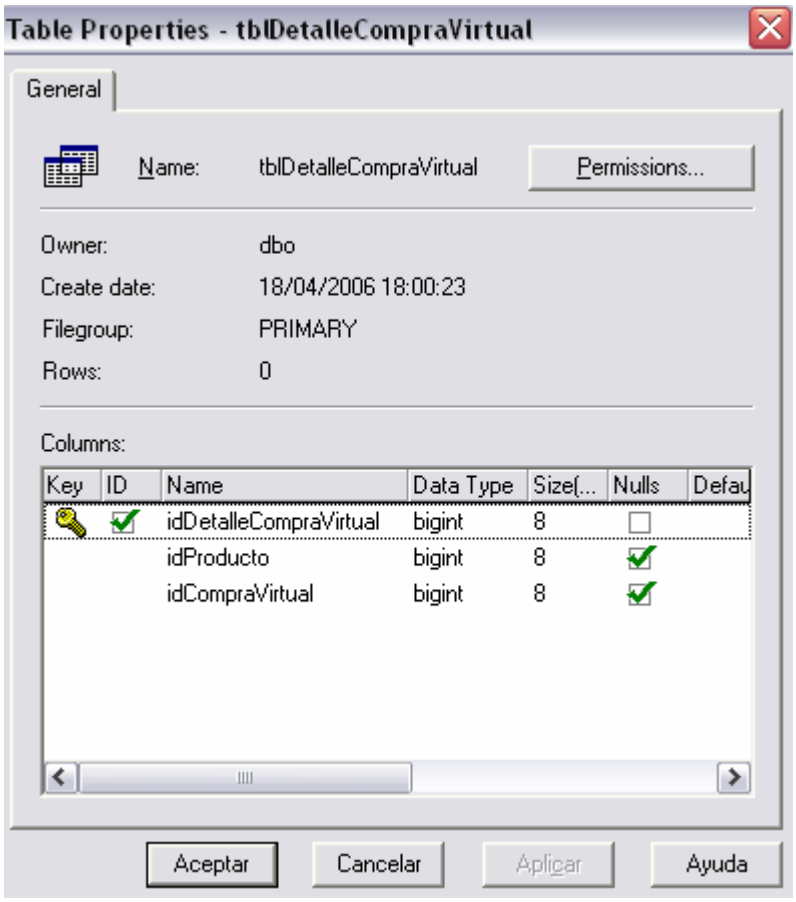
La tabla tiene por objetivo tener un listado de todas las ciudades donde hay responsables de cajas del amor. Conformada por campos como "idCiudad" el cual es un campo auto numérico y actúa como clave primaria, "nombreCiudad" el cual como su nombre lo indica almacena el nombre del la ciudad ingresada.

Descripción Detallada: tblCompraVirtual



Esta tabla tiene por objetivo almacenar las compras hechas mediante el módulo de E-commerce en la aplicación. Conformada por un campo “idCompra” el cual actúa como campo auto numérico, “idDonante” almacena quién fue la persona que compró la caja, “fechaCompra” almacena la fecha en la cual fue hecha la compra, “totalCompra” almacena el valor en dinero de la compra y por ultimo “numeroComprobante” es el número que le asigna el banco como comprobante de la transacción.

Descripción Detallada: tblDetalleCompraVirtual



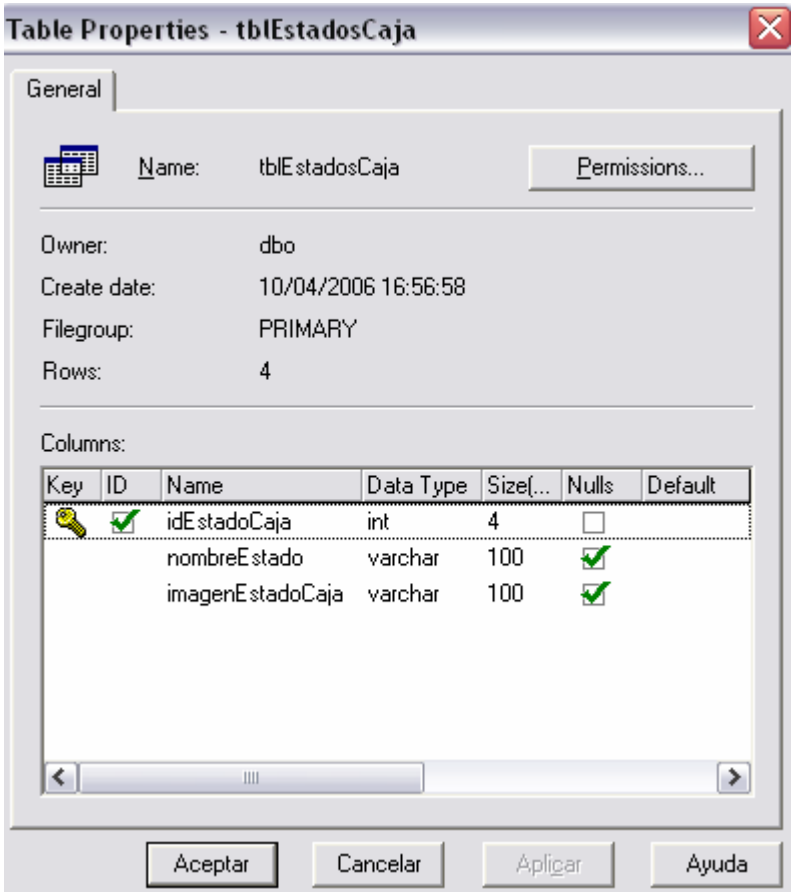
Esta tabla tiene por objetivo almacenar todos los detalles de cada una de las compras realizadas por medio del módulo E-commerce del sitio. Conformada por campos como “idDetalleCompraVirtual” el cual actúa como campo auto numérico y clave primaria, “idCompraVirtual” el cual actúa como clave foránea de la tabla “tblCompraVirtual” para indicar a que compra hace referencia el registro ingresado y por último encontramos “idProducto” el cual actúa como clave foránea de la tabla “tblProducto” para indicar que producto fue comprado en cada detalle de la compra virtual.

Descripción Detallada: tblDonante

	Column Name	Data Type	Length	Allow Nulls
🔑	idDonante	bigint	8	
	nombreDonante	varchar	100	✓
	apellidoDonante	varchar	100	✓
	correoElectronicoDona	varchar	100	✓
	telefonoDonante	varchar	20	✓
	direccionDonante	varchar	500	✓
	pagoCaja	varchar	50	✓
	pagoMercado	varchar	50	✓
	activo	bit	1	✓

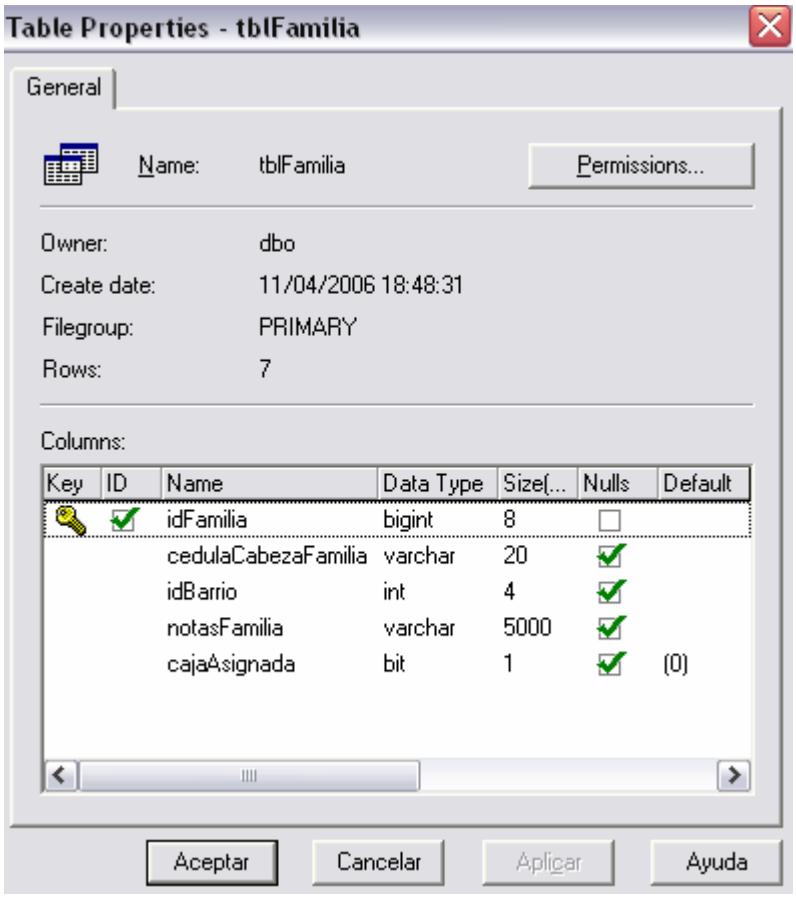
Esta tabla tiene por objetivo almacenar todos los donantes de cajas al interior del sistema, conformada por los siguientes campos:
“idDonante” el cual es un campo auto numérico y actúa como clave primaria, “nombreDonante” como su nombre lo indica almacena el nombre del donante, lo mismo sucede con el campo “apellidoDonante”, “correoElectronicoDonante”, “telefonoDonante”, “direccionDonante”. El campo “pagoCaja” indica si la persona que compro la caja la pagó al momento de hacer la compra o no, igual sucede con el campo “pagoMercado” ya que hay personas que solo hacen la colaboración monetaria mas no están interesadas en llenarlas ellas mismas, y por ultimo encontramos el campo “activo” el cual especifica si el donante está activo o no en el sistema.

Descripción Detallada: tblEstadosCaja



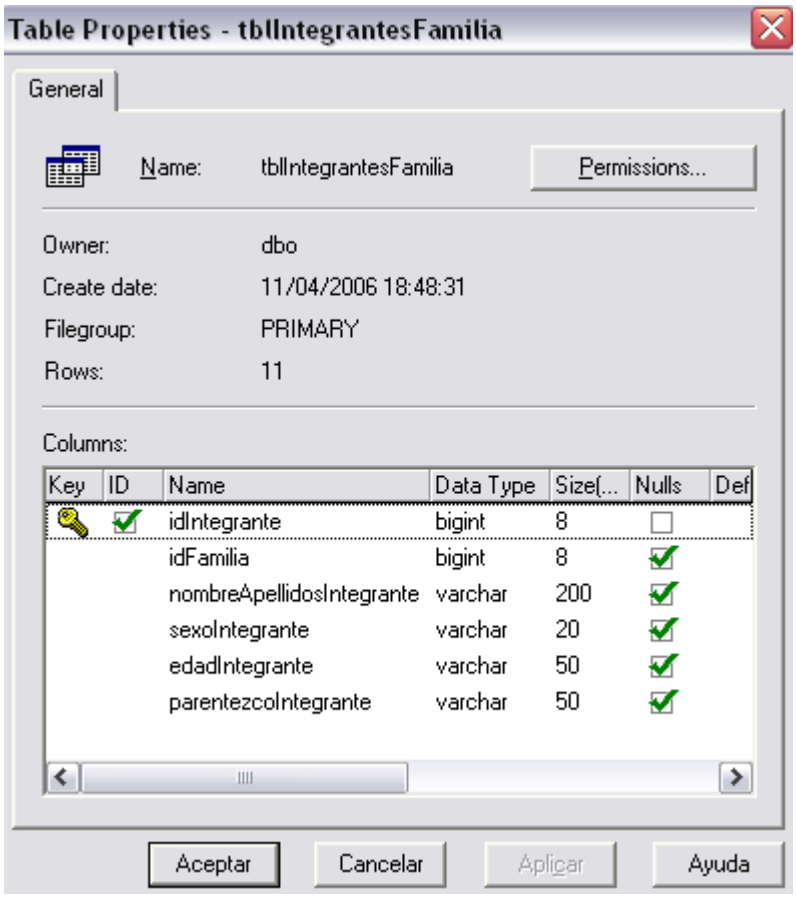
Esta tabla tiene por objetivo almacenar los posibles estados en los cuales se pueda encontrar una caja en un momento determinado. “idEstadoCaja” actúa como campo auto numérico y clave primaria, “nombreEstado” como su nombre lo indica contiene el nombre del estado, y por ultimo “imagenEstadoCaja” almacena una imagen asociada al estado ingresado.

Descripción Detallada: tblFamilia



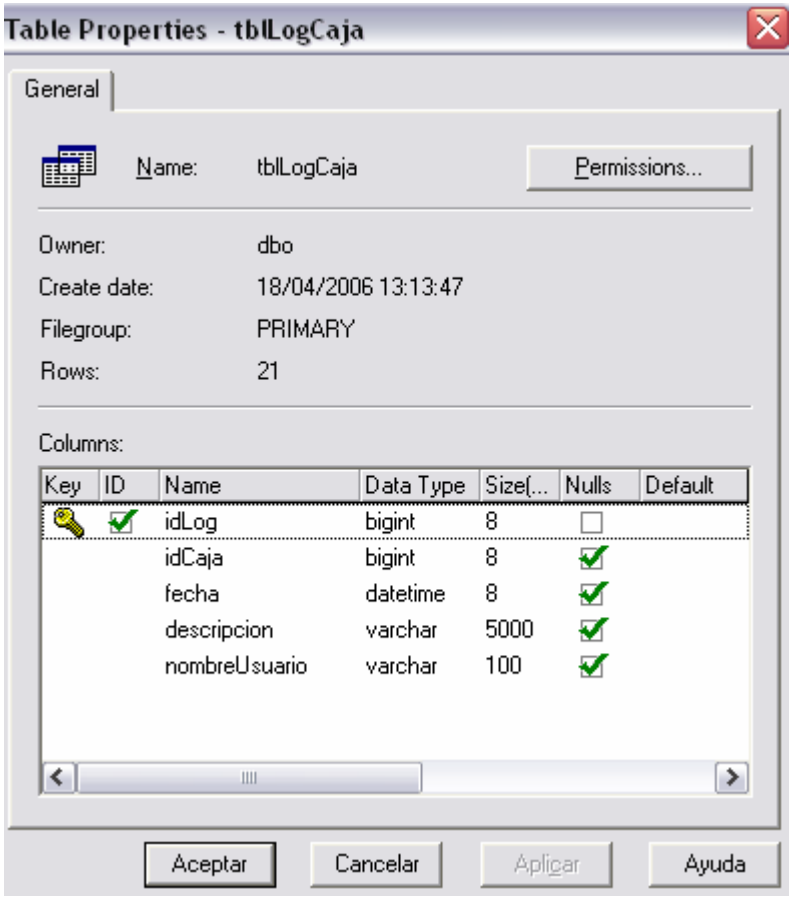
Esta tabla tiene por objetivo almacenar los datos de las familias existentes en el sistema. Conformada por “idFamilia” el cual es un campo auto numérico y que actúa como clave primaria, “cedulaCabezaFamilia” como su nombre lo indica contiene la cédula del integrante cabeza de la familia ingresada, “idBarrio” actúa como clave foránea de la tabla “tblBarrio” y almacena el id del barrio al cual la familia pertenece, “notasFamilia” contiene notas relacionadas a la familia en ingreso y por ultimo “cajaAsignada” indica si esta familia ya tiene o no una caja asignada.

Descripción Detallada: tblIntegrantesFamilia



Esta tabla tiene por objetivo almacenar todos los integrantes relacionados a una familia dada. "idIntegrante" actúa como clave primaria y es un campo auto numérico, "idFamilia" actúa como clave foránea de la tabla "tblFamilia" para indicar a que familia pertenece el integrante que está siendo ingresado, "nombreApellidosIntegrante" almacena como su nombre lo dice los nombres y apellidos del integrante, lo mismo sucede para los campos "sexoIntegrante", "edadIntegrante" y "parentezcoIntegrante".

Descripción Detallada: tblLogCaja



Esta tabla tiene por objetivo almacenar todos los cambios que pueda tener una caja dada en el sistema como sus cambios de estados, cambios de ubicaciones, etc.

“idLog” es un campo auto numérico y actúa como clave primaria, “idCaja” relaciona el log a una caja en particular y este campo actúa como clave foránea de la tabla “tblCaja”, “fecha” indica la fecha en que fue ingresado el log de la caja, “descripción” puede llevar como su nombre lo indica una descripción de lo sucedido, y por último almacena en el campo “nombreUsuario” el nombre del usuario que efectuó la operación con el objetivo de llevar un control.

Descripción Detallada: tblProducto

	Column Name	Data Type	Length	Allow Nulls
🔑	idProducto	bigint	8	
	nombreProducto	varchar	100	✓
	imagenProducto	varchar	100	✓
	idCategoria	int	4	✓
	activo	bit	1	✓
	precio	money	8	✓
	paquete	bit	1	✓
	descripcionProducto	varchar	200	✓

Esta tabla tiene por objetivo almacenar todos los productos existentes de la tienda virtual. “idProducto” es un campo auto numérico que actúa como clave primaria en la tabla, “nombreProducto” como su nombre lo indica lleva el nombre del producto ingresado, lo mismo sucede con “imagenProducto”, “idCategoria” actúa como clave foránea de la tabla “tblCategoriaProducto” y tiene por objetivo clasificar el producto en una de las varias categorías existentes, “activo” indica si el producto estará o no disponible para ser comprado, “precio” indica cual es el precio del producto, “paquete” indica si el producto es un combo, es decir, si es una caja prepago que ya viene con el mercado incluido, y por último encontramos el campo “descripcionProducto” que lleva una descripción para el producto siendo ingresado.

Descripción Detallada: tblResponsable

	Column Name	Data Type	Length	Allow Nulls
🔑	idResponsable	bigint	8	
	cedula	varchar	20	✓
	nombre	varchar	100	✓
	apellidos	varchar	100	✓
	email	varchar	80	✓
	telefonoCasa	varchar	20	✓
	celular	varchar	20	✓
	idCiudad	int	4	✓
	activo	bit	1	✓

Esta tabla tiene por objetivo almacenar toda la información relativa a los responsables de las cajas del amor, es decir, las personas que se encargan de distribuirlas, venderlas y ubicarlas en los diferentes sitios de distribución.

Conformada por campos como “idResponsable” el cual actúa como clave primaria en la tabla y es un auto numérico, “cedula”, “nombre”, “apellidos”, “email”, “telefonoCasa”, “celular” son campos relativos a la información del responsable como sus nombres lo indican. “idCiudad” actúa como clave foránea de la tabla “tblCiudad” e indica a que ciudad pertenece el responsable, y por último tenemos el campo “activo” el cual indica si el responsable esta actualmente activo o no en el sistema.

Pagos en Línea

Con el objetivo de masificar la venta de cajas a través de Internet, se creo un módulo que permitiera la compra de cajas a través de una tienda virtual, es por esto que se buscó que bancos y entidades prestaran el servicio de pagos mediante Internet, entre estos se encontraron empresas privadas donde el servicio que se prestaba era a través de Bancolombia y Conavi (Todo1) en la mayoría de los casos, debido a esto se decidió obviar estos terceros y hablar directamente con Todo1.

Es importante aclarar que el sistema de pagos que se buscaba debía de ser principalmente desde cuentas de ahorros/corriente y posiblemente pero no necesariamente con tarjetas de crédito.

Dado el caso que se quisiera implementar el pago con tarjetas de crédito se encontró que para permitir pagos con tarjetas VISA hay que contratar el servicio directamente con Redeban Multicolor y para tarjetas MasterCard a Credibanco, lo que implica la configuración extra de la tienda virtual con dos proveedores mas de pagos en donde los protocolos de comunicación varían notablemente e incurren en mayores gastos no solo de tiempo de desarrollo sino de dinero a la hora de contratar el servicio.

A continuación se hace una breve descripción de lo que ofrece el servicio de la empresa TODO1, como funciona, y como se ha de integrar al sistema.

Descripción general del servicio TODO1:

Para que una tienda virtual pueda ofrecer el servicio, se debe cumplir con todo el proceso de vinculación de la entidad financiera (BANCOLOMBIA y/o CONAVI), el cual incluye el diligenciamiento de la respectiva solicitud y la firma del convenio “e-pagos” por el representante legal de la empresa dueña de la tienda virtual que ofrecerá el servicio.

Es importante aclarar que este servicio no incluye pagos con tarjetas de crédito Visa o Mastercard.

Este servicio de e-pagos esta concebido como una solución de pagos en línea B2C, es decir, para que empresas puedan recibir pagos de personas naturales.

Este servicio permite al establecimiento de comercio vender en su sitio WEB y obtener pagos (confirmación y abono del dinero) en tiempo real.

El medio de pago opera bajo el esquema de transferencia de fondos mediante un débito directo a una cuenta de depósitos BANCOLOMBIA o CONAVI (cuenta corriente y ahorros) del cliente pagador y un crédito a la cuenta de depósitos del cliente recaudador (facturador o Tienda Virtual), teniendo en cuenta que los pagos desde cuentas BANCOLOMBIA solo son abonados en cuentas BANCOLOMBIA y los pagos desde cuentas CONAVI solo son abonados en cuentas CONAVI.

Al momento de contratar el servicio con TODO1, se exige que la tienda virtual cumpla con las siguientes condiciones:

- La tienda debe ser un sitio "Seguro".
- La tienda tiene que generar un número de referencia (o número de factura) para identificar el pago.
- Es un recaudo con validación en el que la tienda suministra la información básica del pago (referencia y valor) mediante un enlace dinámico, generado con la firma digital proporcionada por TODO1.
- La transacción de pago se realiza directamente en los servidores de TODO1. Estos servidores cuentan con la aprobación y certificación de BANCOLOMBIA y CONAVI para su funcionamiento.
- La confirmación del pago por parte de TODO1 se realiza mediante un enlace dinámico a la tienda.
- TODO1 habilita gratuitamente un servicio de consulta vía Web, para que la tienda virtual pueda hacer seguimiento a los pagos recibidos.
- También a través del sistema *Enlínea BANCOLOMBIA y de la *Sucursal Virtual de Empresas BANCOLOMBIA, se puede confirmar el abono de los pagos en tiempo real, consultando los movimientos de la respectiva cuenta recaudadora en BANCOLOMBIA.
- Igualmente, a través del sistema *Gerencia Electrónica CONAVI o de CONAVI.com*, se puede confirmar el abono de los pagos en tiempo real, consultando los movimientos de la respectiva cuenta recaudadora en CONAVI.

Proceso de pago

Se ingresa a la tienda virtual, selecciona el(los) articulo(s) a comprar, selecciona la opción de e-pagos BANCOLOMBIA, la cual envía el usuario a la sección de pagos de TODO1, en la que el cliente ingresa su número de identificación y la clave de servicios electrónicos. Una vez aprobada la validación de identificación, selecciona la cuenta desde donde desea realizar el pago, verifica los datos (referencia y valor) que le suministra la pantalla de confirmación del pago y lo efectúa. Una vez realizado el pago, presiona un hipervínculo que lo regresa a la tienda virtual. El proceso es similar cuando se realiza el pago con CONAVI, excepto que en vez de seleccionar la opción “e-pagos BANCOLOMBIA” se selecciona “e-pagos CONAVI”.

Requisitos Generales

Tienda Virtual (Cliente recaudador)

- Para recibir pagos de clientes BANCOLOMBIA, debe tener cuenta corriente BANCOLOMBIA.
- Para recibir pagos de clientes CONAVI, debe tener cuenta de ahorros CONAVI.

Recomendación: Ser un sitio seguro (Certificado SSL). No es imprescindible, pero se recomienda para mejorar la seguridad.

Actualmente el sitio Web de la caja del amor desde donde se realizarán las ventas no cuenta con un certificado digital, pero la idea es adquirir uno a principios del año 2007.

Nota: Toda la transacción monetaria en los servidores de TODO1 sí se efectúa bajo SSL y fuertes esquemas de seguridad.

Usuario (Cliente-pagador)

De BANCOLOMBIA:

- Tener una cuenta de depósitos, corriente o de ahorros BANCOLOMBIA (No requiere tarjeta débito o crédito).
- Tener habilitada la clave para los servicios electrónicos BANCOLOMBIA (Cajeros, Sucursal Telefónica, Sucursal Virtual (para las empresas, etc.)

De CONAVI:

- Tener una cuenta de ahorros CONAVI
- Haber inscrito la cuenta de ahorros al servicio de Pagos Virtuales CONAVI.
- Tener habilitada la clave de su tarjeta débito.

Seguridad

TODO1 proporciona a la tienda virtual una firma digital. Ella permite realizar firmas digitales sobre documentos que se encuentren en forma digital. Esta clave es usada en el proceso de generación de la firma digital sobre la factura a ser pagada por su cliente. La única utilidad que tiene es la de garantizar la integridad y la autenticidad de las ordenes de compra generadas por el sistema de la tienda y redireccionar las transacciones al sistema de TODO1.

- La privacidad de la información se garantiza mediante el uso del estándar de seguridad SSL.
- La autenticidad del recaudador y la integridad de la información se asegura por medio de la firma digital.
- Los bancos asociados aseguran el no repudio de la transacción, exigiendo al usuario (Cliente-pagador) en el momento del pago, la respectiva validación con la clave para servicios electrónicos.
- El cliente pagador BANCOLOMBIA tiene la opción de seleccionar los topes máximos de transferencias de fondos en BANCOLOMBIA.
- El cliente pagador CONAVI tiene la opción de habilitar/deshabilitar su cuenta para realizar pagos por Internet.

Ventajas

- Fácil implementación del esquema para la tienda.
- Sencilla utilización del servicio por parte de los clientes pagadores.
- Disponibilidad del Servicio 7 días, 24 horas y en cualquier lugar del mundo.
- Alternativa adicional de recaudo.
- Fácil masificación del servicio.
- Transferencias de fondos en tiempo real.
- Confirmación de la transacción en tiempo real.
- La entidad financiera responde por el no repudio de la transacción a diferencia de los pagos con Tarjeta de Crédito.
- Acceso a los clientes que realizan transacciones por la Sucursal Virtual BANCOLOMBIA.
- Acceso a los clientes que realizan transacciones por CONAVI.com

Tarifa del Servicio

El banco por medio del cual se efectúe el pago, cobrará una comisión del 1.5% del valor total de la transacción, con un mínimo que lo define el banco. Sobre el costo de la comisión, se cobrará el IVA del 16%.

Ejemplo: Valor del pago $100.000

Valor de la comisión $100.000x1.5%= $1.500 +IVA

Valor del pago $20.000

Valor de la comisión $20.000x1.5%= $300 + IVA

Si el mínimo establecido por el banco es mayor a los $300, se cobrará esa tarifa del banco.

No hay costos fijos mensuales, pero TODO1 cobrará $50.000 + IVA por la activación del servicio con cada banco y por cualquier modificación hecha por el comercio que implique la activación de una nueva llave digital.

NOTA: No hay ningún costo para el cliente pagador.

Proceso de Instalación

Para ver en detalles cuales son los pasos requeridos para la instalación del módulo de e-pagos de TODO1 por favor referirse al Anexo #5.

CONCLUSIONES Y RECOMENDACIONES

La implementación de un sistema de información es un gran avance para una organización sin ánimo de lucro que basa sus procesos y tareas en manejo de información.

Se desarrolló un sistema de información con base en las especificaciones presentadas en este documento, el cual cumplió satisfactoriamente las necesidades de la organización la caja del amor, facilitando el proceso de manejo de inventarios.

El sistema de e-pagos aunque ya está implementado solo se verá sus resultados en el programa del presente año de la caja del amor (Diciembre de 2006), de donde se podrán posteriormente con base en los datos recolectados sacar interesantes conclusiones.

El personal de la caja del amor estuvo presente durante toda la construcción de este sistema de información, lo cual fue clave en el éxito del mismo, lo cual demuestra que involucrar al cliente en la construcción del software de una manera proactiva y no reactiva permite obtener grandes resultados.

El sistema de información de es funcional y totalmente aplicable para la organización la caja del amor. Su estructura de página Web permite que su acceso sea a través de cualquier equipo al interior y exterior de la organización.

En el conteo de cajas al interior de la bodega con código de barras disminuyó tiempos que antes tomaban mas de 5 horas a solo 30 ~ 45 minutos, lo cual muestra que la tecnología al ser bien aplicada trae resultados beneficiosos no solo en dinero y tiempo sino también en la actitud de las personas en cuanto al trabajo y dedicación.

Los tiempos para realizar reportes se redujeron drásticamente de tiempos que antes tomaban 2 horas a solo un par de minutos, lo que permite una toma de decisiones mucho más rápida y acertada.

La disponibilidad de la información al estar centralizada permitió que cualquier miembro del equipo la pudiera consultar en cuestión de minutos y no en días como se hacia anteriormente ya que como solo una persona tenía acceso a la información había que esperar a que esta tuviera tiempo disponible para sacar el reporte solicitado.

BIBLIOGRAFIA

- Microsoft MSDN.
- Application Blocks.
- Log 4 net.
- ICONTEC: Normas colombianas para la presentación de tesis, trabajos de grado y otros trabajos de investigación. Quinta actualización. Santa fe de Bogotá D.C. ICONTEC, 2002, NTC 1486.
- http://ccc.inaoep.mx/~dtapia/francisco/UML%20--%20Diagramas%20de%20Casos%20de%20Uso.htm
- www.dcc.uchile.cl/~psalinas/uml/casosuso.html
- http://www.vico.org/MuestrarioDiagCU.pdf

ANEXOS

Anexo 1.

Desprendible usado al momento de vender una caja del amor.

Anexo 2.

Manual de Usuario aplicación Web manejo de inventarios.

Anexo 3.

Formato de censos familias beneficiarias (Formato Excel).

Anexo 4.

Casos de Uso.

Anexo 5.

Manual de instalación módulo e-pagos TODO1.