

**SISTEMA DE GESTIÓN INTEGRAL DE SOPORTE INTERNO**

PROYECTO FINAL TC1/TD

**-GUÍA DE CONTENIDOS-**

ESTEBAN MAS (58223)

Sede Centro

Comisión A

Turno Mañana

2017

Docente: Esteban Calabria

**ÍNDICE**

1 Historial De Revisión 6

2 Introducción 7

3 Propósito 8

3.1 Propósito 8

3.2 Manejo Y Seguimiento De Los Tickets 8

3.3 Base De Datos Histórica De Incidentes (Base De Conocimiento) 9

3.4 Rápido Acceso A Estadísticas Y Métricas 9

3.5 Apoyar A La Gerencia En La Toma De Decisiones 9

4 Descripción Funcional Y Alcance 10

4.1 Funcionalidades 10

4.1.1 Gestión De Tickets 10

4.1.2 Gestión De Problemas 10

4.1.3 Gestión Del Cambio 10

4.1.4 Reportes 11

4.1.5 Autoservicio 11

4.1.6 Procesos 11

4.1.7 Gestión Del Nivel Del Servicio Y Sla 11

4.2 Alcance 12

5 Sentencia Que Define El Problema 12

6 Sentencia Que Define La Posición Del Producto 14

7 Definiciones, Acrónimos, Y Abreviaciones 14

7.1 Definiciones 14

7.2 Acrónimos 15

7.3 Abreviaturas 15

8 Diseño 16

8.1 Arquitectura 16

8.2 Incidentes 16

8.3 Flujo De Trabajo 17

9 Descripción De Stakeholders (Participantes En El Proyecto) Y Usuarios 18

9.1 Resumen De Stakeholders 18

9.2 Resumen De Usuarios 18

9.3 Entorno De Usuario 19

10 Perfil De Los Stakeholders 20

10.1 Representante Del Área Técnica Y Sistemas De Información 20

10.2 Perfiles De Usuario 20

10.2.1 Supervisor - Sectores 20

10.2.2 Jefe De Sector 21

10.2.3 Administrador - Sector 21

10.2.4 Responsable Del Sector Soporte 22

10.2.5 Administrador Del Sector Soporte 22

10.2.6 Cliente 22

11 Diagrama De Despliegue 24

11.1 Diagrama De Componentes (Arquitectura General) 25

12 Diagrama De Actividades Genérico 27

13 Modelo De Datos 28

14 Casos De Uso 29

14.1 Especificaciones, Diagramas De Clase Y Diagrama De Secuencia De Casos De Uso 30

14.1.1 Caso De Uso – AC001: Ingresar Al Sistema 30

14.1.1.1 Diagrama 30

14.1.1.2 Especificación 30

14.1.1.3 Diagrama De Secuencia 31

14.1.1.4 Diagrama De Clases 32

14.1.2 Caso De Uso – AC002: Salir Del Sistema 33

14.1.2.1 Diagrama 33

14.1.2.2 Especificación 33

14.1.2.3 Diagrama De Secuencia 34

14.1.2.4 Diagrama De Clases 35

14.1.3 Caso De Uso – AC003: Gestión De Errores 36

14.1.3.1 Diagrama 36

14.1.3.2 Especificación 36

14.1.3.3 Diagrama De Secuencia 38

14.1.3.4 Diagrama De Clases 39

14.1.4 Caso De Uso – FA001: Crear Familia 40

14.1.4.1 Diagrama 40

14.1.4.2 Especificación 40

14.1.5 Caso De Uso – CU002: Registro En Bitácora 42

14.1.5.1 Diagrama 42

14.1.5.2 Especificación 42

14.1.5.3 Diagrama De Secuencia 43

14.1.5.4 Diagrama De Clases 44

14.1.6 Caso De Uso – CU003: Consulta De Bitácora 45

14.1.6.1 Diagrama 45

14.1.6.2 Especificación 45

14.1.6.3 Diagrama De Secuencia 46

14.1.6.4 Diagrama De Clases 47

14.1.7 Caso De Uso - ID001: Crear Idioma 48

14.1.7.1 Diagrama 48

14.1.7.2 Especificación 48

14.1.7.3 Diagrama De Secuencia 49

14.1.7.4 Diagrama De Clases 50

14.1.8 Caso De Uso - ID002: Cambiar Idioma 51

14.1.8.1 Diagrama 51

14.1.8.2 Especificación 51

14.1.8.3 Diagrama De Secuencia 52

14.1.8.4 Diagrama De Clases 53

14.1.9 Caso De Uso – CU004: Resguardo De Datos 54

14.1.9.1 Diagrama 54

14.1.9.2 Especificación 54

14.1.9.3 Diagrama De Secuencia 55

14.1.9.4 Diagrama De Clases 56

14.1.10 Caso De Uso – Cu005: Recuperación De Datos 57

14.1.10.1 Diagrama 57

14.1.10.2 Especificación 57

14.1.10.3 Diagrama De Secuencia 58

14.1.10.4 Diagrama De Clases 59

14.1.11 Caso De Uso - Gestión De Encriptado 60

14.1.11.1 Diagrama 60

14.1.11.2 Especificación 60

14.1.11.3 Diagrama De Secuencia 61

14.1.11.4 Diagrama De Clases 61

14.2 Casos De Uso Relacionados 62

14.2.1 Cu001: Cargar Formulario Principal 62

14.2.2 Cu005: Suscribir Formulario 63

15 Diagrama De Clases De Negocio 64

16 Diagrama De Base De Datos 65

17 Diagrama De Paquetes (Arquitectura) 66

17.1.1 Capa De Acceso A Datos (DAL) 67

17.1.2 Capa De Servicios DAL 69

17.1.3 Capa De Estructura (BE) 71

17.1.4 Capa De Negocio (BLL) 74

17.1.5 Capa De Presentación (GUI) 75

17.1.6 Capa De Seguridad 78

18 Tutorial - ¿Cómo Crear Un Abm? 79

18.1 Creación De Estructura 79

18.2 Creación Del Orm (DAL) 82

18.3 Creación De Las Reglas De Negocio (BLL) 86

18.4 Creación De La Vista 87

19 Anexos 90

19.1 Prototipo De Pantalla 90

19.2 Prototipo De Buscador De Clientes 93

19.3 Validaciones 94

19.4 Servicio De Traducción 95

19.5 Login 99

19.6 Usuarios 100

19.7 Form Familias 102

19.7.1 Form Familia 104

19.8 Form Bitácora 106

20 Repositorio Github 107

21 Bibliografía 107

# HISTORIAL DE REVISIÓN

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Autor | Descripción |
| 09/04/2017 | 1.0 | ESTEBAN MAS | Generación del documento. |
| 15/04/2017 | 1.0.1 | ESTEBAN MAS | Introducción y Propósito |
| 16/04/2017 | 1.0.2 | ESTEBAN MAS | Descripción funcional y alcance |
| 23/04/2017 | 1.1 | ESTEBAN MAS | Sentencia que define el problema, definiciones, acrónimos y abreviaciones |
| 06/05/2017 | 1.2 | ESTEBAN MAS | Diseño y arquitectura |
| 25/05/2017 | 1.3 | ESTEBAN MAS | Descripción y perfil de Stakeholders |
| 11/07/2017 | 1.4 | ESTEBAN MAS | Diagrama de despliegue y de componentes |
| 20/06/2017 | 1.4.1 | ESTEBAN MAS | Diagrama de actividades genérico |
| 25/06/2017 | 1.4.2 | ESTEBAN MAS | Modelo de datos |
| 01/07/2017 | 1.5 | ESTEBAN MAS | Listado de caso de uso |
| 08/07/2017 | 1.5.1 | ESTEBAN MAS | Casos de uso AC001, AC002, AC003 |
| 09/07/2017 | 1.5.2 | ESTEBAN MAS | Casos de uso FA001, CU002,CU003 |
| 15/07/2017 | 1.5.3 | ESTEBAN MAS | Casos de uso ID001, ID002, CU004 |
| 17/07/2017 | 1.5.4 | ESTEBAN MAS | Casos de uso CU005, AC004 |
| 18/07/2017 | 1.5.5 | ESTEBAN MAS | Casos de uso Relacionados |
| 19/07/2017 | 1.6 | ESTEBAN MAS | Diagrama clase de negocio y de base de datos |
| 20/07/2017 | 1.6.1 | ESTEBAN MAS | Diagrama de paquetes, Capas DAL y Servicios DAL |
| 21/07/2017 | 1.7 | ESTEBAN MAS | Capas BE, BLL, GUI |
| 22/07/2017 | 1.7.1 | ESTEBAN MAS | Capa de Seguridad |
| 22/07/2017 | 1.7.2 | ESTEBAN MAS | Creación Reglas de Negocio y creación de la vista |
| 23/07/2017 | 1.8 | ESTEBAN MAS | Anexos: Prototipos de pantalla, buscador de clientes, Validaciones |
| 23/07/2017 | 1.8.1 | ESTEBAN MAS | Anexos: Prototipo de Servicio de traducción |
| 24/07/2017 | 1.8.2 | ESTEBAN MAS | Anexos: Prototipo pantalla Login, usuarios |
| 24/07/2017 | 1.8.3 | ESTEBAN MAS | Anexos: Prototipo de pantalla Forms Familia y Familias |
| 25/07/2017 | 1.8.4 | ESTEBAN MAS | Anexos: Prototipo de pantalla Form Bitácora |
| 25/07/2017 | 1.9 | ESTEBAN MAS | Repositorio GitHub |
| 29/07/2017 | 2.0 | ESTEBAN MAS | Bibliografía |
| 30/07/2017 | 2.1 | ESTEBAN MAS | Anexos: Prototipo de pantalla Forms Cliente, Clientes y Selecciona Cliente |

# INTRODUCCIÓN

Una Mesa de Ayuda le permite al departamento de IT responder incidentes inesperados dentro de la infraestructura IT o los servicios que provee. Una Mesa de Ayuda reacciona ante los incidentes y es usada para manejar problemas cuando los mismos surgen, permitiendo llevar un registro, control y métricas de los problemas que se presentan día a día en la organización. La mesa de ayuda se debe enfocar en facilitar la comunicación entre el equipo de soporte y los usuarios para ayudar a resolver los problemas, y su propósito es mantener activa la infraestructura y los servicios en el día a día, buscando trabajar más eficientemente y manejar mejor las solicitudes que se reciben.

# ****PROPÓSITO****

## Propósito

El propósito es brindar un sistema que nos permita poder gestionar, organizar, cuantificar y analizar el soporte tecnológico interno de la compañía, haciendo foco en los SLAs internos para mejorar los tiempos de resolución de incidentes, generando una base de conocimiento histórico, generar información útil a la gerencia.

## ****Manejo y seguimiento de los tickets****

En lo que respecta a dar respuesta a incidentes técnicos o solicitudes de servicio, la base de cualquier sistema de Help Desk debe ser el manejo eficiente y registro de tickets. Aunque suene obvio, la habilidad de organizar efectivamente y monitorear la carga de trabajo activo ofrece grandes beneficios en términos de rendimiento y tiempo de resolución, y al final, la eficiencia de toda la organización.

***Puntos clave a evaluar:***

* ¿Cómo es presentada la carga de trabajo actual? ¿Los ticktes están organizados de forma lógica, tomando en cuenta la prioridad de sus SLAs, y puedes ver esta información en un vistazo?
* ¿Los tickets que requieren nuestra atención directa, aprobación o asignación, están separados en un orden lógico? ¿Presenta el sistema detalles suficientes en el dashboard o en la primera vista del ticket, o necesitan detallar más a través de múltiples pantallas para encontrar la información que necesitas?
* ¿Qué tan dinámico es el proceso de reasignar o escalar tickets, puedes involucrar múltiples miembros del equipo en la resolución de un ticket? ¿Qué tan fácil es agrupar múltiples tickets con una misma causa raíz y mantener a múltiples usuarios finales en el ciclo mientras la resolución avanza?

## ****Base de datos histórica de incidentes (Base de conocimiento)****

Permite al equipo técnico compartir sus conocimientos colectivos mediante una base de conocimiento integrada lo que brindará un beneficio e impacto significativo en el tiempo de resolución y reducir la carga de trabajo.

***Puntos clave a evaluar:***

* ¿Qué tan fácil es crear y mantener artículos, navegar la base de conocimiento para encontrar la información relevante, y si la base de conocimiento presentada en el momento necesario?

## ****Rápido acceso a estadísticas y métricas****

El sistema permitirá el análisis, evaluación de las respuestas y tratamiento de todos los incidentes reportados. Por medio de lo que muestran los gráficos y estadísticas, se podrá medir la retroalimentación del cliente interno, evaluar el desempeño y rendimiento del equipo técnico.

## Apoyar a la gerencia en la toma de decisiones

Para que se pueda tomar una decisión, es necesario basarse en información actualizada y confiable.

Nuestro sistema ofrece las herramientas necesarias para generar dicha información, tanto textual como gráfica. Y en base a esto, poder tomar las decisiones necesarias para mejorar los la atención y resolución de tickets por parte de los grupos de soporte técnico.

# DESCRIPCIÓN FUNCIONAL Y ALCANCE

## FUNCIONALIDADES

### Gestión de tickets

Cuando un usuario final tiene un problema o una solicitud, es necesario poder darle respuesta lo más rápido posible. Con esta herramienta resulta fácil iniciar, gestionar y reportar las solicitudes de servicio y los problemas de TI que afectan a su empresa y su gente, no solo porque brinda una rápida resolución sino porque ofrece una buena experiencia al usuario final, además de la posibilidad de evaluar la satisfacción del cliente.

### Gestión de problemas

Dado el flujo incesante de incidentes hacia la mesa de ayuda, las organizaciones a menudo luchan por encontrar tiempo y recursos para realizar la gestión de problemas y eliminar los problemas recurrentes. El sistema de Mesa de Ayuda proporciona un número de funcionalidades de gestión de problemas que ayudan a ahorrar tiempo, además de permitir el proceso de avance durante el ciclo de vida del incidente, ayudando a llevar adelante la gestión de los mismos.

### Gestión del cambio

El cambio en las empresas modernas exige velocidad, pero la velocidad conlleva riesgos tanto en TI como en el negocio. Las funcionalidades de gestión del cambio ayudan a garantizar que todos los cambios de TI sean administrados eficientemente y que se haga un manejo eficaz del riesgo, de modo que se minimicen las fallas y demoras debidas a los cambios. El objetivo es facilitar la vida de mesa de servicio y mejorar la reputación de TI dentro de la empresa, minimizando el nivel de problemas relacionados con el cambio y su impacto negativo en las operaciones del negocio.

### [Reportes](http://www.invgate.com/es/service-desk/reporting-and-analytics/)

El sistema de Mesa de ayuda brinda reportes sobre el accionar de cada grupo de soporte y los tiempos de resolución de los tickets.

### Autoservicio

Las funcionalidades de autoservicio les permiten a los clientes proporcionar un canal de acceso y comunicación moderno, similar a los bienes de consumo; la autoayuda es a través de una base de conocimiento predictiva, funcionalidades de catálogo de servicios y registro de problemas del usuario. Además, por supuesto, el autoservicio también le quita presión al equipo de soporte de TI, ya que los usuarios finales pueden ayudarse a sí mismos, con el ahorro de costos asociado.

### Procesos

Se puede integrar completamente TI con los procesos de negocio y crear un proceso automatizado, independientemente del departamento. El sistema simplifica también el soporte de TI mediante el uso de automatización inteligente. Por ejemplo, el enrutamiento de los tickets de problemas al miembro del staff más adecuado para su resolución o las solicitudes dirigidas a los aprobadores, o la conversión de mensajes de correo electrónico de usuarios finales en tickets.

### Gestión del Nivel del Servicio y SLA

Priorizar las operaciones sobre la base del impacto y la urgencia, más las metas de nivel de servicio acordadas. Hacer el seguimiento de tickets hasta la resolución, recibir alertas cuando las metas de nivel de servicio corren el riesgo de no cumplirse o no se cumplen, y en definitiva ayudar a los individuos y el equipo a cumplir o superar los niveles de desempeño acordados. Diferentes niveles de servicio pueden ser aplicables a los incidentes y solicitudes de servicio, para los diferentes tipos de problema o petición y para diferentes lugares, equipos o roles.

## ALCANCE

El alcance del sistema está definido por los sectores que se ven afectados por la problemática interna del soporte técnico formal e informal que se viene prestando por cada uno de los sectores técnicos de la compañía.

# SENTENCIA QUE DEFINE EL PROBLEMA

|  |  |
| --- | --- |
| El problema de | Poder llevar una métrica de los casos de soporte que se atienden en cada sector y así poder cuantificar el tiempo que demanda esto, generando información que ayude a la toma de decisiones para saber cuándo es necesario incorporar personal y también para saber cuáles son los temas sobre los que hacer foco.  Gestionar los pedidos de soporte realizados por los clientes internos.  Gestionar los niveles de soporte. |
| afecta a | Gerencia de Desarrollo,  Gerencia de Operaciones,  Gerencia de Seguridad Informática,  Gerencia de QA,  Gerencia de Tecnología,  Gerencia de Administración. |
| El impacto asociado es | Almacenar toda la información referente a los casos de soporte atendidos por cada uno de los sectores y que esta información esté al instante accesible y actualizada en lugares físicamente muy distantes es un proceso prácticamente imposible de realizar en el caso de que no esté informatizado. |
| Una solución adecuada sería | Informatizar el proceso, usando una red local con una base de datos accesible desde los distintos nodos de la red y generar interfaces amigables y sencillas con las que acceder a dicha base de datos. |

# Sentencia que define la posición del Producto

|  |  |
| --- | --- |
| para | Gerencia de Desarrollo,  Gerencia de Operaciones,  Gerencia de Seguridad Informática,  Gerencia de Tecnología, |
| quienes | Controlan los casos de soporte técnico. |
| El nombre del producto | Es una herramienta software. |
| que | Almacena la información necesaria para gestionar un área de soporte. |
| no como | El sistema actual. |
| Nuestro producto | Permite gestionar las distintas actividades de soporte de los sectores, mediante una interfaz gráfica sencilla y amigable. Además proporciona un acceso rápido y actualizado a la información desde cualquier punto que tenga acceso a la base de datos. |

# DEFINICIONES, ACRÓNIMOS Y ABREVIACIONES

## Definiciones

**CLIENTE:** Persona física que inicia el proceso de soporte.

**ANALISTA SOPORTE:** Persona física que se encarga de atender el pedido de soporte y derivarlo al sector técnico que corresponda.

**TÉCNICO:** Persona física que realiza el seguimiento y cierre de los tickets.

**TICKET ASIGNADO:** Caso desoporte que fue asignado a un técnico por el responsable del

sector.

**TICKET ABIERTO:** Caso de soporte que fue asignado a un técnico y que se encuentra en ejecución.

**TICKET CERRADO:** Caso de soporte que fue resuelto por el técnico y el sistema cierra

**TICKET SOLUCIONADO:** Caso de soporte que fue resuelto por el técnico y que queda a la espera de cierre por parte del sistema.

**TICKET PENDIENTE:** Caso de soporte que fue atendido por el técnico y se encuentra a la espera de algún requerimiento externo al sector, como por ejemplo a la espera del feedback del soporte de alguno de los vendors.

**TICKET RECHAZADO:** Caso de soporte que no cumple con ciertos requerimientos o que fue asignado a un grupo técnico erróneo.

**RESPONSABLE:** Persona física que se encarga de asignar los casos de soporte, que llegan al sector, al técnico que se encargará de darle seguimiento y solución al caso.

**BASE DE DATOS:** Repositorio lógico donde se resguardan los datos del sistema,

**SISTEMA:**

## Acrónimos

**SGISI:** Sistema de Gestión Integral de Soporte Interno.

**BBDD:** Base de Datos.

## Abreviaturas

**SIST**= Sistema.

**TKT:** Ticket.

**SOP:** Soporte.

**TEC:** Técnico

**RESP:** Responsable.

**TAS:** Ticket asignado.

**TA:** Ticket abierto.

**TC:** Ticket cerrado.

**TP:** Ticket pendiente.

**TR:** Ticket rechazado

**CLI:** Cliente

# DISEÑO

## ARQUITECTURA

El diseño más común de sistema de seguimiento de incidentes es relativamente simple. Una base de datos en el principal repositorio de almacenamiento para todos los datos. Estos datos son manejados por la capa de negocio de la aplicación. Esta capa brinda a los datos en bruto más estructura y significado. preparándola para ser comprensible por los usuarios. Los ahora datos comprensibles por humanos son presentados al soporte técnico por otra aplicación de software o página web. El usuario final del sistema de seguimiento de incidentes puede crear nuevos incidentes por completar, leer incidentes existentes, añadir detalles de los mismos o resolverlos. Cada vez que el usuario del sistema efectúa un cambio, el sistema de seguimiento de incidentes registra la acción y quién la hizo, llevando un histórico de las acciones tomadas. Cada usuario del sistema puede tener incidentes asignados, esto es, cada usuario es responsable por la apropiada resolución de ese incidente. Esto es presentado generalmente al usuario en un formato de lista. El usuario puede tener la opción de reasignar un incidente a otro usuario, de ser necesario. Por seguridad, un sistema de seguimiento de incidentes autenticará sus usuarios antes de permitirles el acceso al sistema.

## INCIDENTES

Los incidentes pueden tener muchos aspectos. Cada incidente en el sistema puede tener un nivel de urgencia asignado, basado en la importancia total de ese incidente. Los incidentes críticos son los más severos que deben ser resueltos en la forma más expedita posible, tomando precedencia sobre todos los demás incidentes. Los incidentes de urgencia baja o cero son menores, y deben ser resueltos como lo permita el tiempo. Otros detalles de los incidentes incluyen la experiencia del cliente con el incidente (sea interna o externa), fecha de registro, descripciones detalladas del problema experimentado, intentos de soluciones y otra información relevante. Como se notó previamente, cada incidente mantiene un historial de cada cambio.

## FLUJO DE TRABAJO

Este escenario sirve para demostrar cómo un sistema típico de seguimiento de incidencias puede trabajar:

1. Un técnico del sector soporte recibe una llamada telefónica, correo electrónico, u otra comunicación de un cliente acerca de un programa. Algunas aplicaciones proveen reporte automático de errores a partir de bloques de manejo de excepciones.
2. El técnico verifica que el problema es real y no sólo percibido. El técnico podría también asegurarse de que se ha obtenido suficiente información acerca del problema por parte del cliente. Esta información generalmente incluye el ambiente del cliente, cuándo y cómo ocurre el incidente, y otras circunstancias relevantes.
3. El técnico crea el incidente en el sistema, ingresa toda la información relevante tal como fue proporcionada por el cliente.
4. El técnico del sector soporte deriva el caso al sector técnico correspondiente para su análisis y posterior solución del incidente.
5. Conforme se trabaja en el incidente, el sistema es actualizado con nuevos datos por el técnico. Cada intento de reparar el problema debe ser anotado en el sistema de incidentes.
6. Después de que el incidente es totalmente solucionado, es marcado como resuelto en el sistema de seguimiento de incidentes.

El problema puede no ser totalmente corregido, aun cuando pueda estar marcado como resuelto. El problema puede deberse al diseño, un incidente conocido, o tener una solución parcial adecuada.

# DESCRIPCIÓN DE STAKEHOLDERS (PARTICIPANTES EN EL PROYECTO) Y USUARIOS

Para proveer de una forma efectiva servicios de soporte técnico que se ajusten a las necesidades de los usuarios, es necesario identificar e involucrar a todos los participantes en el proyecto como parte del proceso de modelado de requerimientos. También es necesario identificar a los usuarios del sistema y asegurarse de que el conjunto de participantes en el proyecto los representa adecuadamente. Esta sección muestra un perfil de los participantes y de los usuarios involucrados en el proyecto, así como los problemas más importantes que éstos perciben para enfocar la solución propuesta hacia ellos. No describe sus requisitos específicos ya que éstos se capturan mediante otro artefacto. En lugar de esto proporciona la justificación de por qué estos requisitos son necesarios.

## RESUMEN DE STAKEHOLDERS

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Descripción** | **Responsabilidades** |
| Cristian Azame | Responsable del sector de Soporte de 1er nivel. | El stakeholder realiza:  Representa a todos los usuarios posibles del sistema.  Seguimiento del desarrollo del proyecto.  Aprueba requisitos y funcionalidades |

## RESUMEN DE USUARIOS

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | | **Descripción** | **Stakeholder** |
| SupervisorACT1 AdministradSupervisorS – Sectores | Responsable de cada uno de los sectores afectados, encargado de la gestión del sector, de verificar el correcto desarrollo de las tareas internas. | | STK1 ServidoresssServidores, Almacenamiento, Desarrollo, QA, Seguridad Informática, Operaciones, Microinformática. |
| ACT2 Jefe de sector | Responsable de verificar el desempeño del sector, de generar los reportes de atención haciendo foco en el cumplimiento de los SLAs internos acordados. Es el responsable de analizar, en base a los reportes obtenidos la necesidad de incrementar la dotación de personal. | | Servidores, Almacenamiento, Desarrollo, QA, Seguridad Informática, Operaciones, Microinformática. |
| Administrador - Sectores | Responsable de la atención, seguimiento y solución de los casos de soporte recibidos por el sector | | Servidores, Almacenamiento, Desarrollo, QA, Seguridad Informática, Operaciones, Microinformática. |
| Responsable del sector Soporte | Responsable del seguimiento de las aperturas y derivación de los casos en los sectores responsables. | | STK3 Soporte |
| ACT5 Administrador Sector Soporte | Responsable de la apertura y derivación de los casos en los sectores responsables. | | STK10 Soporte |

## ENTORNO DE USUARIO

Los usuarios entrarán al sistema identificándose sobre un ordenador con un sistema operativo Windows 7/10 y tras este paso entrarán a la parte de aplicación diseñada para cada uno según su perfil en el sistema. Este sistema es similar a cualquier aplicación Windows y por tanto los usuarios estarán familiarizados con su entorno.

Los informes serán generados con Microsoft Word versión 2016, lo cual también resultará familiar.

# PERFIL DE LOS STAKEHOLDERS

## REPRESENTANTE DEL ÁREA TÉCNICA Y SISTEMAS DE INFORMACIÓN

|  |  |
| --- | --- |
| **Representante** | Cristian Azame |
| **Descripción** | Responsable del sector de Soporte de 1er nivel. |
| **Tipo** | Experto de Sistemas. |
| **Responsabilidades** | Encargado de mostrar las necesidades de cada usuario del sistema. Además, lleva a cabo un seguimiento del desarrollo del proyecto y aprobación de los requisitos y funcionalidades del sistema |
| **Criterio de Éxito** | A definir por el cliente |
| **Grado de participación** | Revisión de requerimientos, estructura del sistema |
| **Comentarios** | Ninguno |

## PERFILES DE USUARIO

### Supervisor - Sectores

|  |  |
| --- | --- |
| **Representante** | Servidores, Almacenamiento, Desarrollo, QA, Seguridad Informática, Operaciones, Microinformática. |
| **Descripción** | Supervisor - Sectores |
| **Tipo** | Usuario del sistema. |
| **Responsabilidades** | Responsable de cada uno de los sectores afectados, encargado de la gestión del sector, de verificar el correcto desarrollo de las tareas internas. |
| **Criterio de Éxito** | A definir por el cliente |
| **Grado de participación** | A definir por el cliente |
| **Comentarios** | Ninguno |

### Jefe de Sector

|  |  |
| --- | --- |
| **Representante** | Servidores, Almacenamiento, Desarrollo, QA, Seguridad Informática, Operaciones, Microinformática. |
| **Descripción** | Jefe de Sector. |
| **Tipo** | Usuario del sistema. |
| **Responsabilidades** | Responsable de verificar el desempeño del sector, de generar los reportes de atención haciendo foco en el cumplimiento de los SLAs internos acordados. Es el responsable de analizar, en base a los reportes obtenidos la necesidad de incrementar la dotación de personal. |
| **Criterio de Éxito** | A definir por el cliente |
| **Grado de participación** | A definir por el cliente |
| **Comentarios** | Ninguno. |

### Administrador - Sector

|  |  |
| --- | --- |
| **Representante** | Servidores, Almacenamiento, Desarrollo, QA, Seguridad Informática, Operaciones, Microinformática. |
| **Descripción** | Administrador - Sector |
| **Tipo** | Usuario experto. |
| **Responsabilidades** | Responsable de la atención, seguimiento y solución de los casos de soporte recibidos por el sector |
| **Criterio de Éxito** | A definir por el cliente |
| **Grado de participación** | A definir por el cliente |
| **Comentarios** | Ninguno. |

### Responsable del sector Soporte

|  |  |
| --- | --- |
| **Representante** | SoporteSTK3 Soporte |
| **Descripción** | Representante del Sector Soporte |
| **Tipo** | Usuario experto. |
| **Responsabilidades** | Responsable del seguimiento de las aperturas y derivación de los casos en los sectores responsables. |
| **Criterio de Éxito** | A definir por el cliente |
| **Grado de participación** | A definir por el cliente |
| **Comentarios** | Ninguno. |

### Administrador del Sector Soporte

|  |  |
| --- | --- |
| **Representante** | STK3 Soporte |
| **Descripción** | Administrador del sector Soporte |
| **Tipo** | Usuario experto. |
| **Responsabilidades** | Responsable de la apertura y derivación de los casos en los sectores responsables |
| **Criterio de Éxito** | A definir por el cliente |
| **Grado de participación** | A definir por el cliente |
| **Comentarios** | Ninguno. |

### 

### Cliente

|  |  |
| --- | --- |
| **Representante** | Servidores, Almacenamiento, Desarrollo, QA, Seguridad Informática, Operaciones, Microinformática. |
| **Descripción** | Usuario. |
| **Tipo** | Usuario de cada sector de la compañia. |
| **Responsabilidades** | Realiza la solicitud de soporte. |
| **Criterio de Éxito** | A definir por el cliente |
| **Grado de participación** | A definir por el cliente |
| **Comentarios** | Ninguno. |

# DIAGRAMA DE DESPLIEGUE

****

Para comprender más en detalle el funcionamiento del Despliegue, a continuación, se anexa un gráfico que explica como estarán distribuidos los equipos involucrados.



## DIAGRAMA DE COMPONENTES (ARQUITECTURA GENERAL)

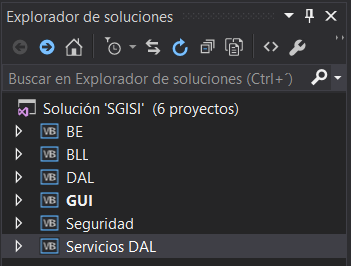
**Importante:**

Para ver en detalle las tareas y las clases que componen estas capas dirigirse a Diagrama de Paquetes.

Este diagrama es una base para entender la creación de la solución en Visual Studio.



Estas capas deberian quedar en Visual Estudio de la siguiente forma.



# DIAGRAMA DE ACTIVIDADES GENÉRICO

****

# MODELO DE DATOS



# CASOS DE USO

|  |  |
| --- | --- |
| **Gestión de Acceso**   * AC001: Ingresar al Sistema * AC002: Salir del Sistema * AC003: Gestión de Errores * AC004: Gestión de Encriptado   **Gestión (genérica) de Sector**   * GE001: Consultar Sector * GE002: Filtrar Sector * GE003: Leer Sector * GE004: Eliminar Sector * GE005: Guardar Sector   **Gestión de Clientes**   * CL001: Consultar Clientes * CL002: Filtrar Cliente * CL003: Eliminar Cliente * CL004: Crear Cliente   **Gestión de Usuarios**   * US001: Consultar Usuarios * US002: Filtrar Usuarios * US003: Eliminar Usuarios * US004: Crear Usuarios   **Gestión de Reportes**   * RE001: Crear Reporte * RE002: Copiar Reporte * RE003: Enviar Reporte (por Email) * RE004: Imprimir Reporte * RE005: Crear Ranking de Casos * RE006: Crear Ranking de Sectores * RE007: Crear Ranking de Técnicos | **Gestión de Casos (Catálogo)**   * CA001: Consultar Casos * CA002. Filtrar Casos * CA003: Leer Casos * CA004: Eliminar Casos * CA005: Guardar Casos   **Gestión de Familias**   * FA001: Crear Familias * FA002: Consultar Familias * FA002: Filtrar Familias * FA003: Leer Familia * FA004: Eliminar Familia * FA005: Guardar Familia   **Gestión de Sector**   * SE001: Consultar Sector * SE002: Filtrar Sector * SE003: Leer Sector * SE004: Eliminar Sector * SE005: Guardar Sector   **Gestión de Genéricos**   * CU001: Cargar Formulario Principal * CU002: Registrar Evento Bitácora * CU003: Consulta de Bitácora * CU004: Gestión Backup * CU005: Gestión Restore   **Gestión de Genéricos**   * ID001: Multi-Idioma * ID002: Traducción de Formularios |

## ESPECIFICACIONES, DIAGRAMAS DE CLASE Y DIAGRAMA DE SECUENCIA DE CASOS DE USO

### CASO DE USO – AC001: Ingresar al Sistema

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | AC001 |
| **NOMBRE** | Ingresar al Sistema |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Usuario necesita ingresar al sistema. | |
| **ACTORES** | Usuario |
| **PRECONDICION** |  |
| **FLUJO NORMAL** | |
| 1. El Usuario hace doble clic en el ejecutable del Software. 2. El Sistema devuelve un Formulario para ingresar Usuario y Contraseña. 3. El Usuario ingresa el Usuario. 4. El Usuario ingresa la Contraseña. 5. El Usuario hace clic en el botón “Ingresar” 6. El Sistema comprueba que exista el Usuario. 7. El Sistema comprueba que la contraseña coincida con la del Usuario. 8. El Sistema registra el ingreso del Usuario. 9. El Sistema abre la pantalla principal del Software. | |

#### Diagrama de Secuencia



#### Diagrama de Clases



### CASO DE USO – AC002: Salir del Sistema

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | AC002 |
| **NOMBRE** | Salir del Sistema |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Usuario necesita salir del sistema. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El Usuario debe estar dentro del sistema |
| **FLUJO NORMAL** | |
| 1. El Usuario Cierra los formularios abiertos. 2. El Usuario hace clic en el en Salir. 3. El Sistema vuelve al Formulario Login para ingresar Usuario y Contraseña. 4. El Usuario hace clic en salir. 5. El Sistema se cierra. | |

#### Diagrama de Secuencia



#### Diagrama de Clases



### CASO DE USO – AC003: Gestión de Errores

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | CU004 |
| **NOMBRE** | Gestión de Errores |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Usuario intenta ingresar al sistema, pero alguno de los datos (Usuario o password) son incorrectos y se muestra un mensaje de error. | |
| **ACTORES** | Usuario |
| **PRECONDICION** |  |
| **FLUJO NORMAL** | |
| 1. El Usuario ingresa el Usuario. 2. El Usuario ingresa la Contraseña. 3. El Usuario hace clic en el botón “Ingresar” 4. El Sistema comprueba que exista el Usuario. 5. El Sistema comprueba que la contraseña coincida con la del Usuario. 6. El Sistema registra el ingreso del Usuario. 7. El Usuario hace clic en el Botón de “Ingresar”. ([Referencia AC001](#_GE001:_Consultar_Entidad))    1. Los datos son incorrectos.       1. Se muestra un mensaje de error. | |

#### Diagrama de Secuencia



#### Diagrama de Clases



### CASO DE USO – FA001: Crear Familia

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | FA001 |
| **NOMBRE** | Crear Familia |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Empleado posiblemente un gerente, necesita una nueva jerarquía dentro de los empleados. Para esto puede crear una Familia en el sistema, el cual le proveerá la posibilidad de seleccionar los módulos a los cuales tiene acceso dicha familia. Para esto el sistema provee una interfaz gráfica simple para poder asignar permisos. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El Usuario debe estar Logeado en el Sistema. |
| **FLUJO NORMAL** | |
| 1. El Usuario hace clic en el Botón de “Familias”. ([Referencia GE001](#_GE001:_Consultar_Entidad)) 2. El Sistema devuelve el Listado de las Familias. 3. El Usuario hace clic en “Agregar”. 4. El Sistema devuelve un formulario vacío para completar los datos de la Familia. 5. El Usuario completa el Nombre de la Familia. 6. El Usuario selecciona los módulos a los cuales tendrá acceso dicha Familia. 7. El Usuario hace clic en “Guardar”. 8. El Sistema valida la Familia. ([Ver Anexo Validaciones](#_Validaciones))    1. Los datos son correctos.       1. Se cierra el formulario.    2. Los datos son incorrectos.       1. Se muestra un mensaje de error. 9. El Sistema guarda la Familia. 10. El Sistema actualiza la grilla de Familias. | |

### CASO DE USO – CU002: Registro en Bitácora

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | CU002 |
| **NOMBRE** | Registro de Bitácora |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Usuario, posiblemente un gerente, necesita relevar lo sucedido durante una jornada o durante otro periodo de tiempo. El sistema provee un listado de los eventos registrados, pero estos posiblemente sean abrumadores debido a su cantidad. Por este motivo el sistema debe proveer al Empelado la posibilidad de filtrar estos registros, tanto por fecha como por actor involucrado. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El Usuario debe estar Logeado en el Sistema. |
| **FLUJO NORMAL** | |
| 1. El Usuario hace clic en el Botón de “Bitacora”. 2. El Sistema devuelve el Listado de los Eventos Registrados. 3. El Usuario completa la fecha desde la cual desea buscar. 4. El Usuario completa la fecha hasta la cual desea buscar. 5. El Usuario completa el Empleado por el cual desea buscar. 6. El Usuario hace clic en “Filtrar” 7. El sistema devuelve un listado con los registros filtrados. | |

#### Diagrama de Secuencia



Un ejemplo en código fuente de cómo se filtra en la capa de Datos:



Nótese que utilizamos ADO Desconectado, por este motivo hacemos uso de la propiedad RowFilter de los DataView, más específicamente el DataView que viene por default en cada DataTable.

#### Diagrama de Clases



### CASO DE USO – CU003: Consulta de Bitácora

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | CU003 |
| **NOMBRE** | Consulta de Bitácora |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Empleado, posiblemente un gerente, necesita relevar lo sucedido durante una jornada o durante otro periodo de tiempo. El sistema provee un listado de los eventos registrados, pero estos posiblemente sean abrumadores debido a su cantidad. Por este motivo el sistema debe proveer al Empelado la posibilidad de filtrar estos registros, tanto por fecha como por actor involucrado. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El Usuario debe estar Logeado en el Sistema. |
| **FLUJO NORMAL** | |
| 1. El Usuario hace clic en el Botón de “Bitacora”. 2. El Sistema devuelve el Listado de los Eventos Registrados. 3. El Usuario completa la fecha desde la cual desea buscar. 4. El Usuario completa la fecha hasta la cual desea buscar. 5. El Usuario completa el Empleado por el cual desea buscar. 6. El Usuario hace clic en “Filtrar” 7. El sistema devuelve un listado con los registros filtrados. | |

#### Diagrama de Secuencia



#### Diagrama de Clases



### CASO DE USO - ID001: Crear Idioma

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | ID001 |
| **NOMBRE** | Crear Idioma |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Usuario posiblemente un gerente, necesita incluir el software en una región donde el idioma no es el mismo que el original del sistema. Para esto tiene la posibilidad de crear un nuevo idioma y completando las distintas palabras y frases en el idioma de destino, poder dar de alta un idioma y de ese modo permitir a los Empleados, en este caso usuarios, hacer uso del software. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El Usuario debe estar Logueado en el Sistema. |
| **FLUJO NORMAL** | |
| 1. El Usuario hace clic en el Botón de “Idiomas”. 2. El Sistema devuelve el Listado de los Idiomas. 3. El Usuario hace clic en “Agregar”. 4. El Sistema devuelve un formulario vacío para completar los datos del Idioma. 5. El Usuario completa el Nombre del Idioma. 6. El Usuario completa una por una las palabras / frases a traducir. 7. El Usuario hace clic en “Guardar”. 8. El Sistema valida el Idioma. ([Ver Anexo Validaciones](#_Validaciones))    1. Los datos son correctos.       1. Se cierra el formulario.    2. Los datos son incorrectos.       1. Se muestra un mensaje de error. 9. El Sistema guarda el Idioma. 10. El Sistema actualiza la grilla de Idiomas. | |

#### Diagrama de Secuencia



#### Diagrama de Clases



### CASO DE USO - ID002: Cambiar Idioma

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | ID001 |
| **NOMBRE** | Crear Idioma |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Usuario se encuentra con un Software poco entendible, debido a que no está en el idioma que el comprende, por ese motivo este caso de uso apunta a la posibilidad de modificar el idioma del software. Lógicamente debe estar creado y configurado previamente. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El Usuario debe estar Logeado en el Sistema. |
| **FLUJO NORMAL** | |
| 1. El Usuario hace clic en el Botón de “Configurar”. 2. El Sistema despliega una lista de idiomas soportados. 3. El Usuario hace clic en el Idioma de su preferencia. 4. El Sistema actualiza las cadenas de texto del Sistema por las del idioma solicitado. | |

#### Diagrama de Secuencia



Para más información sobre cómo se traduce todo lo relacionado con el sistema, ir al [Anexo Servicio de Traducción](#_Servicio_de_Traducción)

#### Diagrama de Clases



### CASO DE USO – CU004: Resguardo de datos

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | CU004 |
| **NOMBRE** | Resguardo de datos |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Usuario será el encargado de realizar el backup de los datos. El sistema permite realizar un backup de la base de datos guardando la información con fecha del backup en una tabla de la base de datos. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El Usuario debe estar Logeado en el Sistema. |
| **FLUJO NORMAL** | |
| 1. El Usuario hace clic en el Menú de “Datos”. 2. El Sistema despliega el Form Backup y Restore de datos. 3. El Usuario hace clic en el Botón Realizar Backup. 4. El Sistema se conecta a la BBDD y realiza el backup de los datos. | |

#### Diagrama de Secuencia



#### Diagrama de Clases



### CASO DE USO – CU005: Recuperación de datos

#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | CU004 |
| **NOMBRE** | Resguardo de datos |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| El Usuario será el encargado de realizar el restore de los datos. En caso de ser necesario realizar el restore se podrán visualizar las fechas con los backups realizados anteriormente y al seleccionar una comenzar con el restore de la base. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El Usuario debe estar Logeado en el Sistema. |
| **FLUJO NORMAL** | |
| 1. El Usuario hace clic en el Menú de “Datos”. 2. El Sistema despliega el Form Backup y Restore de datos. 3. El Usuario hace clic en el Botón Realizar Restore. 4. El Sistema se conecta a la BBDD y realiza el Restote de los datos. | |

#### Diagrama de Secuencia



#### Diagrama de Clases



### CASO DE USO – AC004 Gestión de Encriptado

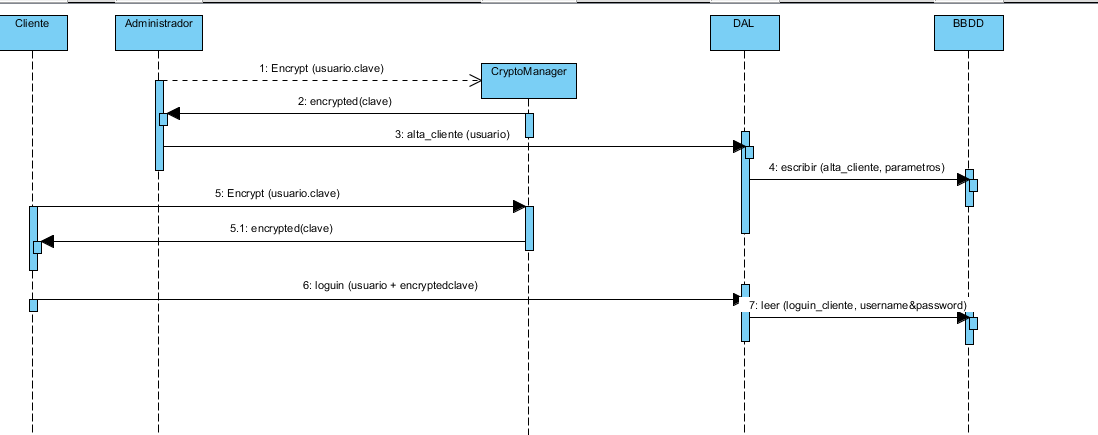
#### Diagrama



#### Especificación

|  |  |
| --- | --- |
| **CODIGO** | AC004 |
| **NOMBRE** | Gestión Encriptado |
| **AUTOR** | Esteban Mas |
| **DESCRIPCION** | |
| Encriptar la información importante en el sistema.  El sistema cuenta con la encriptación de la información sensible tal como el usuario y la contraseña para ingresar a la aplicación. El algoritmo que se utiliza es de criptografía simétrica para contraseñas y hash (md5) para almacenamiento de los dígitos verificadores. | |
| **ACTORES** | Usuario |
| **PRECONDICION** | El usuario debe estar en condiciones de ser dado de alta en el sistema |
| **FLUJO NORMAL** | |
| 1. Alta de usuario en el sistema. 2. Persistencia del usuario con clave encriptado | |
| **FLUJO NORMAL** | |
| 1. Login Usuario. 2. El sistema toma la clave ingresada, la encripta y la compara con la almacenada. | |

#### Diagrama de Secuencia

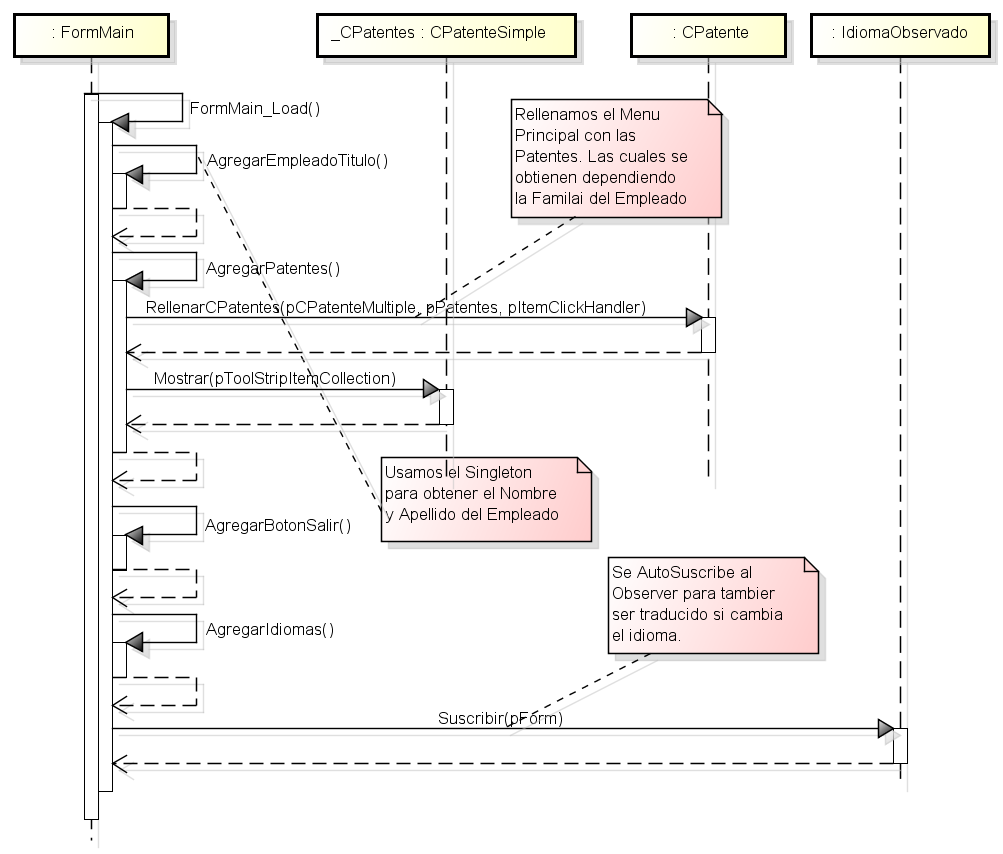


#### Diagrama de Clases

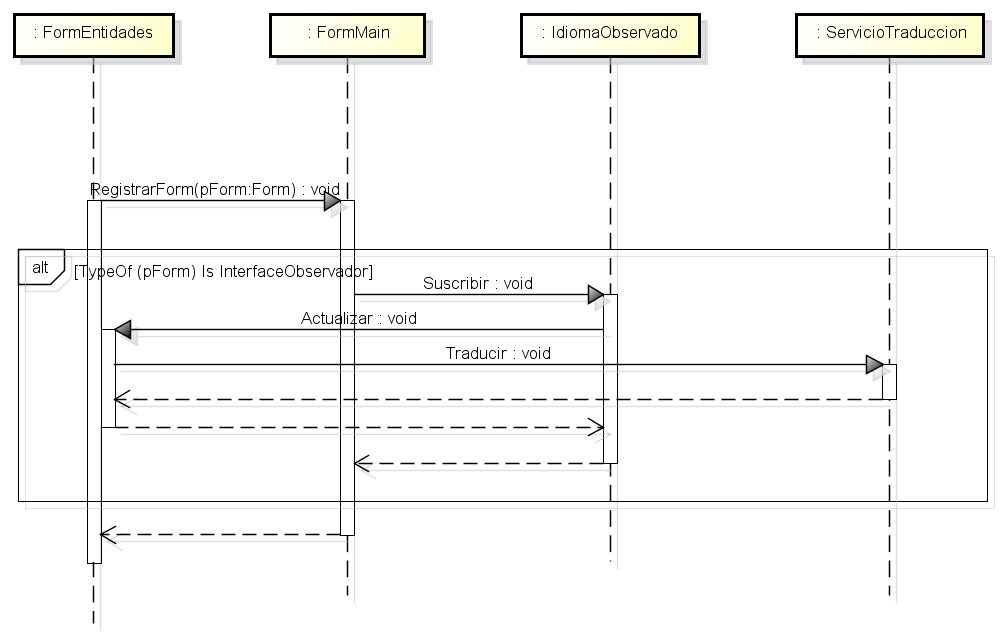
****

## CASOS DE USO RELACIONADOS

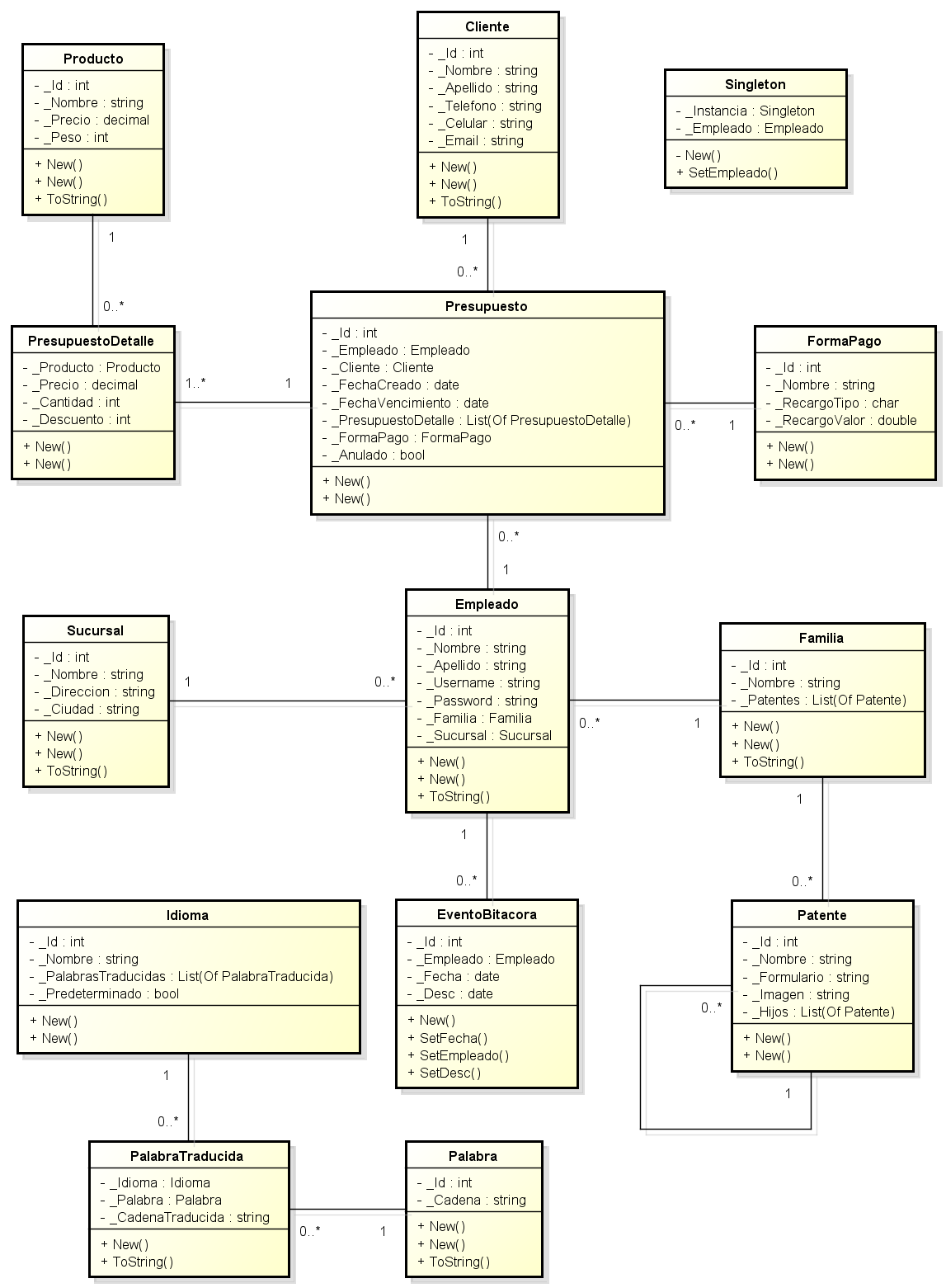
### CU001: CARGAR FORMULARIO PRINCIPAL



### CU005: SUSCRIBIR FORMULARIO



# DIAGRAMA DE CLASES DE NEGOCIO



# DIAGRAMA DE BASE DE DATOS



# DIAGRAMA DE PAQUETES (ARQUITECTURA)



Como se mencionó anteriormente, la solución cuenta con 6 capas.

Estos paquetes a su vez están compuestos por otros paquetes los cuales garantizan el correcto funcionamiento de la aplicación.

A continuación del se detalla cada una, junto con su funcionamiento, dependencias y otros datos importantes para llevar acabo la implementación.

### CAPA DE ACCESO A DATOS (DAL)

Esta capa es la capa que se encarga de acceder a la Base de Datos. Y realiza dos tareas fundamentales:

1. Recuperar el Estado de los Objetos
2. Almacenar el Estado de los Objetos

Funciona como un “Objeto Mapeador”, por lo tanto todas las clases de esta capa deben contar por lo tanto con dos métodos muy importantes. Uno que Recupere el Estado, es decir complete un objeto del tipo del negocio partiendo de un DataRow y viceversa, tomar los datos de un objeto y llevarlos a un registro Datarow, del Datatable correspondiente a la tabla de nuestro negocio.

Un ejemplo de estos métodos en código serían los siguientes.



Es importante destacar que estos métodos deberían ser shared, así nos evitamos repetir código.

Por ejemplo: Nuestra clase DAL para Presupuestos, tiene asociado un Cliente. Entonces cuando el Presupuesto es recuperado, puedo invocar este método CrearObjeto, y pasarle por parámetro lo obtenido en un DataRelation.



Como se puede ver utilizando la funciona shared de cada DAL, nos evitamos repetir código.

Por último, a continuación, se muestran las clases que están involucradas en el negocio.



IMPORTANTE

Evaluar la posibilidad de llevar todo a una clase más general y posteriormente especializar cada una por separado. Y que muchos métodos se repiten, en firma e implementación.

### CAPA DE SERVICIOS DAL

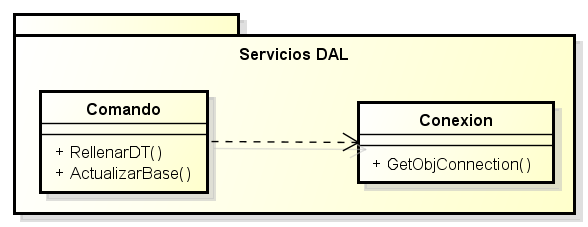
Esta capa estará a cargo de proveer la capacidad de obtener DataTables con la información de una entidad de nuestro negocio. Y de actualizar la información en el motor de persistencia, utilizando como entrada dichos DataTables.

A su vez, tendrá la capacidad de generar el conexionado necesario para que la clase correspondiente pueda obtener esta información.

Como se ve en el Diagrama de Paquete correspondiente, son solo dos clases:

Una que rellena los DataTables, con un parámetro opcional, el cual solo recupera la estructura (Importante a la hora de crear nuevos DataRows) Y que a su vez tiene la capacidad de persistirlos.

Y una clase que provee el conexionado. En nuestro caso será SQLConnection, junto con DataAdapter y CommandBuilder para almacenar y recuperar la información.



IMPORTANTE

Debe evaluarse la posibilidad de intentar llevar esta capa a un nivel inferior y permitir a la capa DAL abstraerse del motor de persistencia. Lo cual implica que esta capa sea la única experta en el motor de persistencia, y que la capa siguiente (DAL), únicamente sea experta en DataSets, DataTable, etc.

Esto nos permitirá en un futuro migrar el motor de base de datos, sin la necesidad de realizar cambios en la capa DAL.

Un ejemplo del conexionado sería:



Y un ejemplo de la Clase Comando sería la siguiente:



Es importante destacar que estamos hablando de un servicio, por este motivo intentamos que todos los métodos sean del tipo shared.

### CAPA DE ESTRUCTURA (BE)

Contendrá las clases que representarán los objetos relacionados con nuestro dominio de problema.

Cada una de estas clases, debe estar preparada para recibir el estado del objeto, por lo tanto tendrá propiedades que representen los atributos y relaciones de las entidades.

Normalmente las clases solo contendrán las propiedades, sin embargo, algunas de estas propiedades, pueden contener un poco de “negocio”, con esto me refiero a que algunas clases como por ejemplo Presupuesto, realiza el cálculo de totales y subtotales, utilizando la información provista en su estado.

De ese modo algunas propiedades se calculan dinámicamente cada vez que se accede a esa propiedad. Por ejemplo:



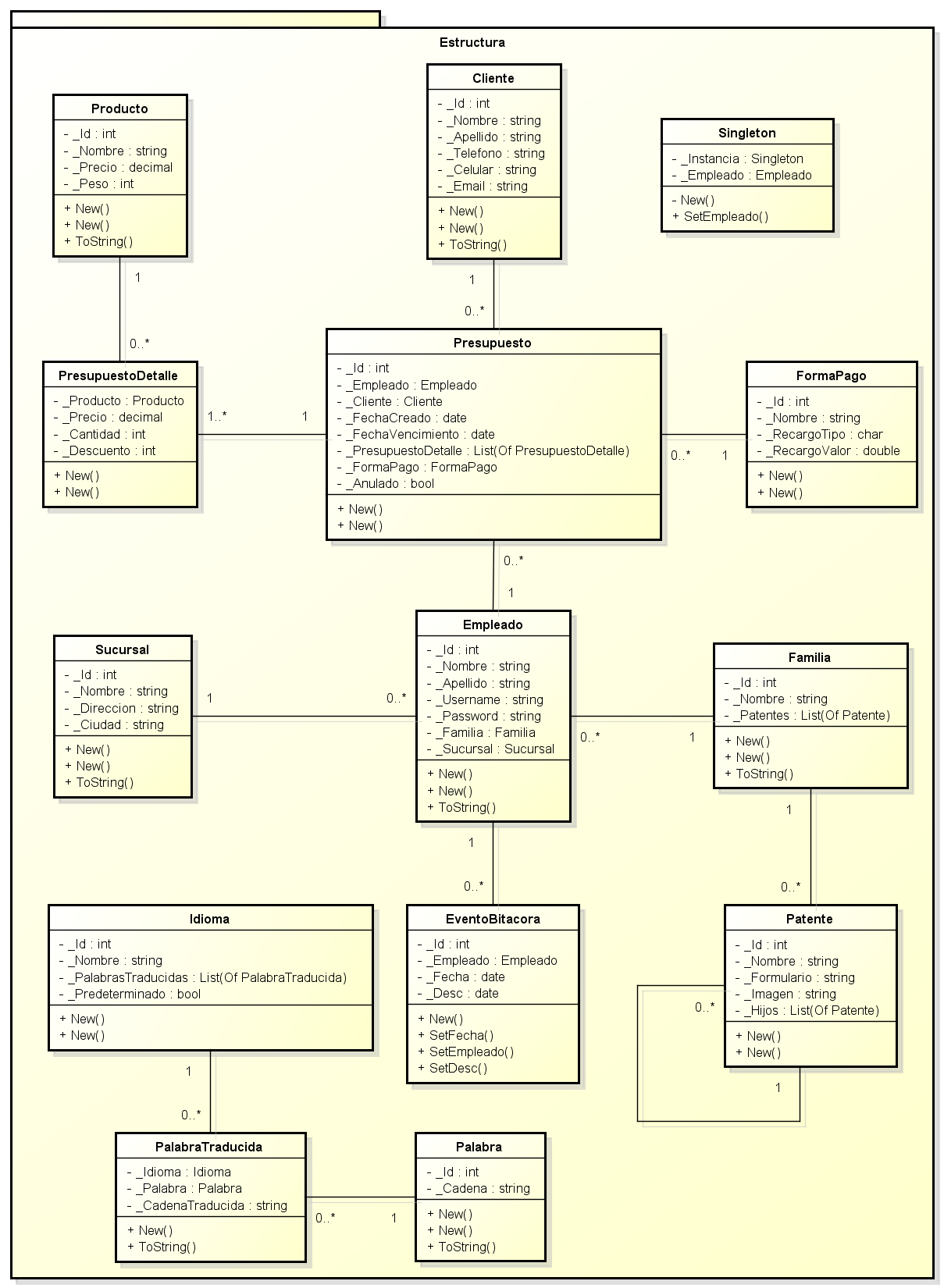
Esta propiedad de los Presupuestos propone sumar los totales de sus detalles, cada vez que se accede a SubTotal.

Esto nos brinda una enorme ventaja, por ejemplo, cuando lo estamos cargando, ya que lo podemos hacer directamente a nivel de vista, y no llamar a la capa de negocio para realizar ningún cálculo.

Como complemento extra en nuestra capa de estructura, nos encontramos con una clase llamada Singleton, esta clase nos provee la capacidad de acceder a un mismo objeto en memoria desde cualquier capa que la referencie.

En nuestro programa, es un elemento clave para almacenar una instancia del usuario logueado.

A continuación, se detallan cada una de las clases involucradas:



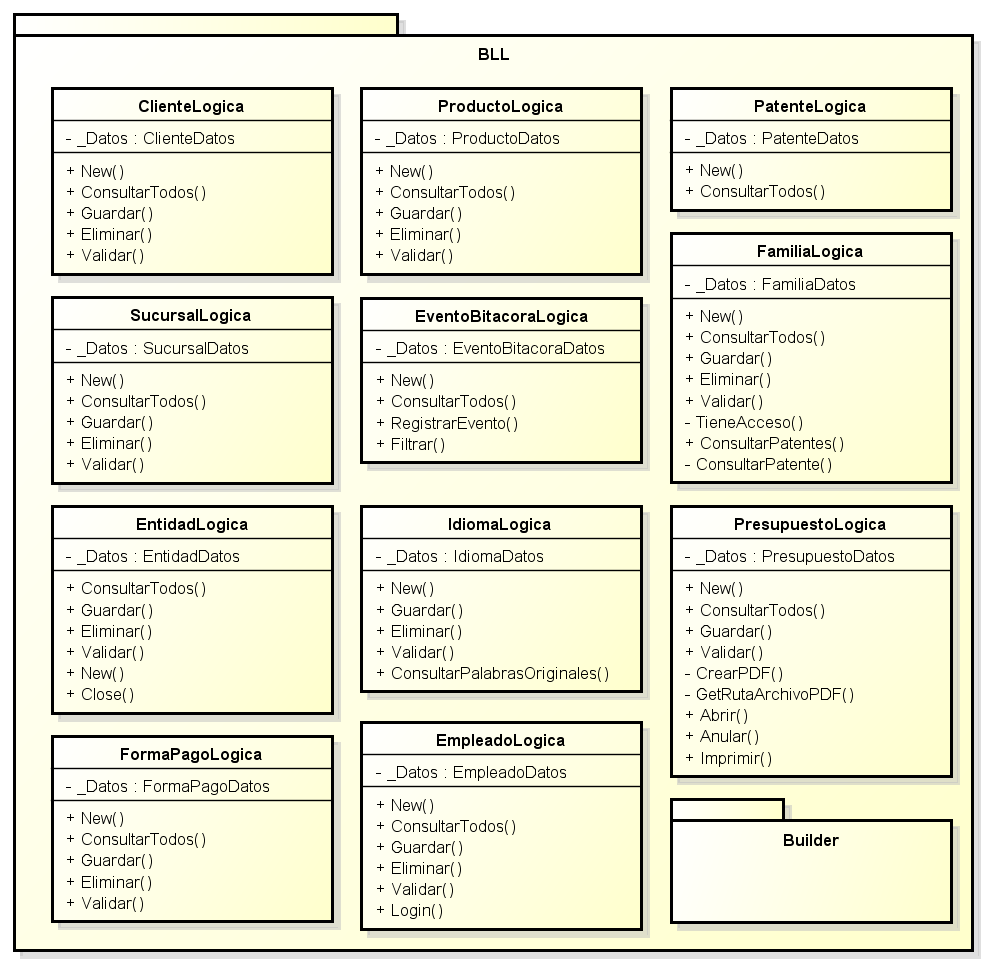
### CAPA DE NEGOCIO (BLL)

Esta capa contendrá las reglas de negocio de nuestro Dominio. Es decir, todas las operaciones que impliquen almacenar, consultar o generar información deben pasar o ser ejecutadas por la capa de negocio, donde estará la lógica necesaria para realizar estas tareas.

Sumado a esta tarea se encuentra la de validar los objetos, por ejemplo, verificar que un Presupuesto no repita dos veces el mismo Producto. Que no se apliquen descuentos excesivos, o inclusive que una cadena no supere el máximo establecido.

Como función extra, es la encargada muchas veces de invocar métodos de Bitácora, para registrar lo sucedido en el programa.

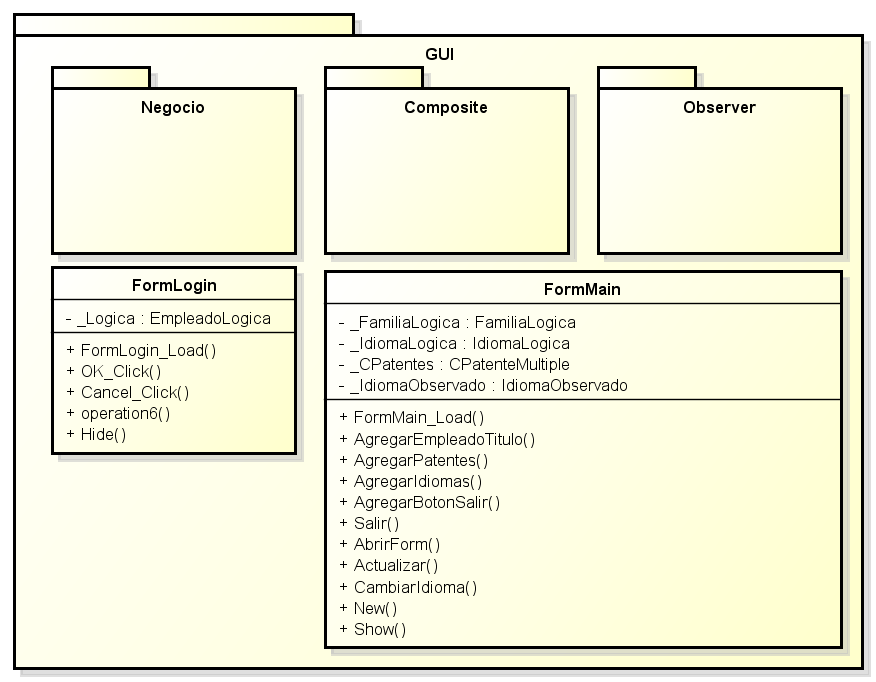
A continuación, se detallan las clases involucradas en esta capa:



### CAPA DE PRESENTACIÓN (GUI)

Será la capa encargada de presentar la información que envía la capa de Negocio en un formato adecuado para interactuar con el usuario. También será la encargada de detectar cambios de idioma y llamar a los servicios pertinentes para traducir los formularios involucrados.

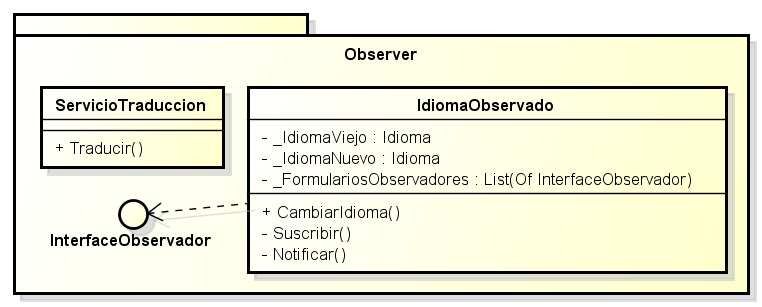
En cuanto a los formularios: para permitir una mayor flexibilidad se optó por utilizar un contenedor MDI, y varios formularios hijos. Para más información sobre el diseño, dirigirse al anexo [Prototipo de Pantalla](#_Prototipo_de_Pantalla).

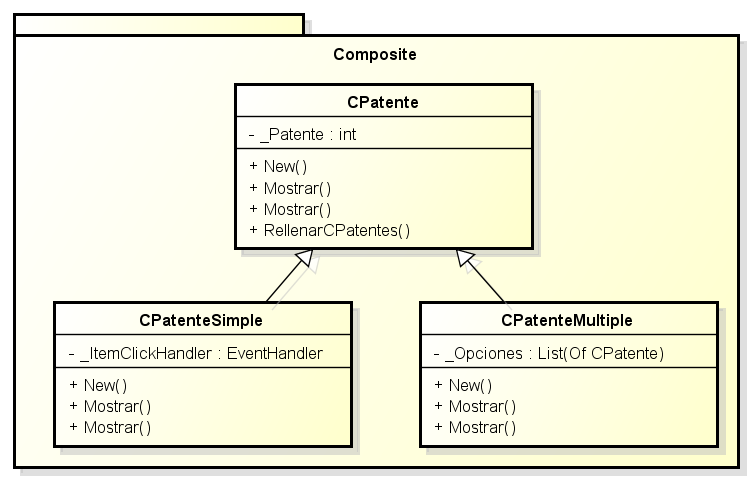


IMPORTANTE

Recordar que será la Capa inicial de la Solución, por lo cual la configuración inicial de algunos parámetros se realizará en el archivo **“App.config”** de esta Capa.

Más específicamente, en nuestro caso se almacenarÁ la cadena de conexión necesaria para conectarse con la base de datos y como segundo parámetro la carpeta de salida de los presupuestos generados.







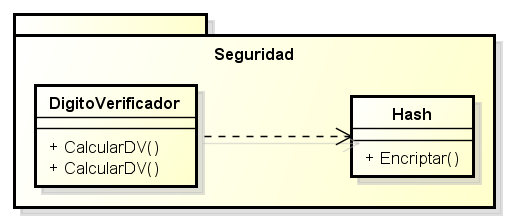
### CAPA DE SEGURIDAD

Esta capa será utilizada por dos capas:

La capa de negocios para realizar la autenticación de los usuarios. Específicamente en la encriptación de las contraseñas almacenadas en la base de datos. Se utiliza el método MD5, el cual, en caso de ser robada la base de datos, impide se conozca la contraseña original del empleado.

La capa DAL para calcular los dígitos verificadores.

Es importante destacar que esta capa ofrece el servicio de Encriptar para cualquier capa que la referencie, esto quiere decir que en un futuro puede ampliarse el funcionamiento o inclusive cambiar el método de encriptación con muy poco impacto.

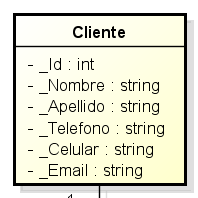
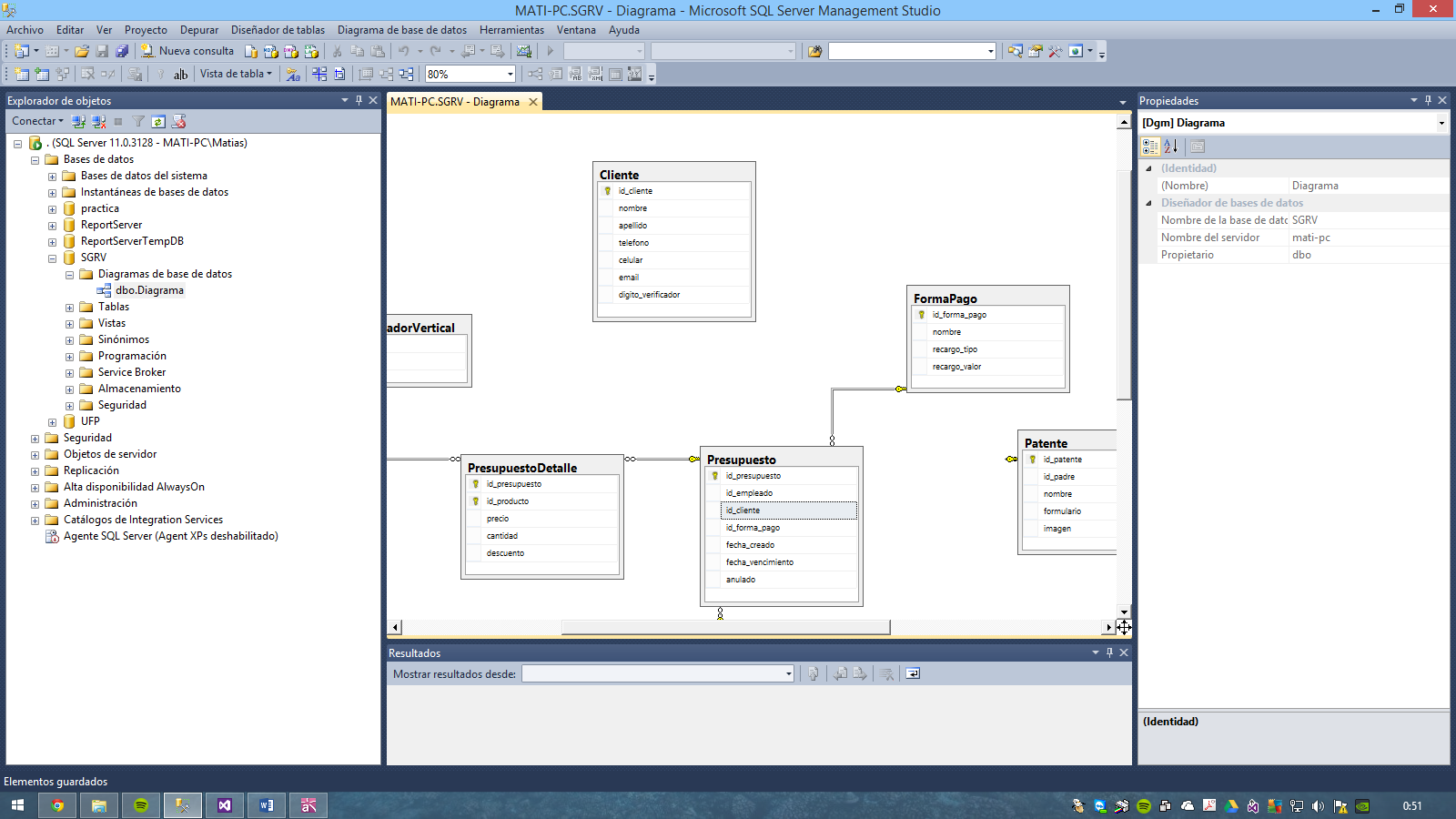


# Tutorial - ¿Cómo crear un ABM?

## CREACIÓN DE ESTRUCTURA

Lo primero que deber realizarse es la Estructura con la cual vamos a trabajar, normalmente es una Clase con los mismos atributos que las tablas en la Base de Datos.

Por ejemplo, convertirnos la siguiente tabla “Cliente” a la clase “Cliente”.



Dejándonos una implementación como al siguiente



Una vez realizada la estructura (propiedades), debemos agregar los constructores.

Primero uno vacío, el cual normalmente se utiliza cuando se crea un nuevo cliente. Y el segundo será uno que permita inicializar el estado, junto con la instanciación.

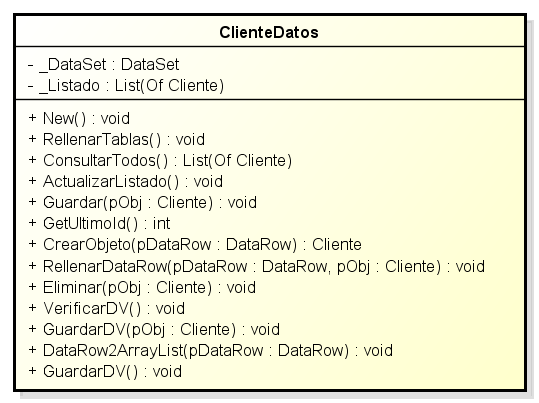


## Creación del ORM (DAL)

Para que funcione correctamente, debemos implementar distintos métodos que nos permitan Mapear desde Objetos a Base de Datos y viceversa. Para esto utilizaremos dos métodos comunes a todos los ORM.

* RellenarDataRow: Encargado de hacer el pase de información desde un Objeto a un Data Row.
* Crear Objeto: Encargado de crear un Objeto desde un DataRow.

Para simplificar la codificación de otras Entidades que manejen Cliente como una agregación. El método RellenarDataRow y CrearObjeto son del tipo Friend. Es decir, no se necesita instanciar la clase Datos para utilizarlos.



A continuación, un ejemplo de la codificación de la clase ClienteDatos:

Public Class ClienteDatos

Private \_DataSet As New DataSet

Private \_Listado As New List(Of Estructura.Cliente)

Public Sub New()

End Sub

Private Sub RellenarTablas()

Me.\_DataSet.Reset()

ServiciosDAL.Comando.RellenarDT("SELECT \* FROM Cliente", Me.\_DataSet.Tables.Add("Cliente"))

Me.\_DataSet.Tables("Cliente").PrimaryKey = {Me.\_DataSet.Tables("Cliente").Columns("id\_cliente")}

ServiciosDAL.Comando.RellenarDT("SELECT \* FROM DigitoVerificadorVertical WHERE tabla = 'Cliente'", Me.\_DataSet.Tables.Add("DigitoVerificadorVertical"))

Me.\_DataSet.Tables("DigitoVerificadorVertical").PrimaryKey = {Me.\_DataSet.Tables("DigitoVerificadorVertical").Columns("tabla")}

End Sub

Public Function ConsultarTodos() As List(Of Estructura.Cliente)

Me.RellenarTablas()

Me.\_DataSet.Tables("Cliente").DefaultView.Sort = "id\_cliente DESC"

Me.\_DataSet.Tables("Cliente").DefaultView.RowFilter = Nothing

Me.ActualizarListado()

Return Me.\_Listado

End Function

Private Sub ActualizarListado()

Me.\_Listado.Clear()

For Each DataRowView As DataRowView In Me.\_DataSet.Tables("Cliente").DefaultView

Me.\_Listado.Add(CrearObjeto(DataRowView.Row))

Next

Me.VerificarDV()

End Sub

Public Sub Guardar(pObj As Estructura.Cliente)

Dim DataRow = Me.\_DataSet.Tables("Cliente").Rows.Find(pObj.Id)

If IsNothing(DataRow) Then

DataRow = Me.\_DataSet.Tables("Cliente").NewRow

RellenarDataRow(DataRow, pObj)

Me.\_DataSet.Tables("Cliente").Rows.Add(DataRow)

Else

RellenarDataRow(DataRow, pObj)

End If

ServiciosDAL.Comando.ActualizarBase("SELECT \* FROM Cliente", Me.\_DataSet.Tables("Cliente"))

If pObj.Id = 0 Then pObj.Id = Me.GetUltimoId()

Me.GuardarDVHorizontal(pObj)

Me.GuardarDVVertical()

Me.RellenarTablas()

End Sub

Private Function GetUltimoId() As Integer

Dim DataTableAux As New DataTable

ServiciosDAL.Comando.RellenarDT("SELECT TOP 1 \* FROM Cliente ORDER BY id\_cliente DESC", DataTableAux)

Dim UltimoCliente As Estructura.Cliente = CrearObjeto(DataTableAux.Rows(0))

Return UltimoCliente.Id

End Function

Public Shared Function CrearObjeto(pDataRow As DataRow) As Estructura.Cliente

Return New Estructura.Cliente(

pDataRow.Item("id\_cliente"),

pDataRow.Item("nombre"),

pDataRow.Item("apellido"),

pDataRow.Item("telefono"),

pDataRow.Item("celular"),

pDataRow.Item("email")

)

End Function

Public Shared Sub RellenarDataRow(pDataRow As DataRow, pObj As Estructura.Cliente)

With pDataRow

.Item("id\_cliente") = pObj.Id

.Item("nombre") = pObj.Nombre

.Item("apellido") = pObj.Apellido

.Item("telefono") = pObj.Telefono

.Item("celular") = pObj.Celular

.Item("email") = pObj.Email

End With

End Sub

Public Sub Eliminar(pObj As Estructura.Cliente)

Dim DataRow = Me.\_DataSet.Tables("Cliente").Rows.Find(pObj.Id)

DataRow.Delete()

ServiciosDAL.Comando.ActualizarBase("SELECT \* FROM Cliente", Me.\_DataSet.Tables("Cliente"))

Me.GuardarDVVertical()

End Sub

Private Sub VerificarDV()

For Each DataRow As DataRow In Me.\_DataSet.Tables("Cliente").Rows

If Not DataRow.Item("digito\_verificador") = Seguridad.DigitoVerificador.CalcularDV(DataRow2ArrayList(DataRow)) Then Throw New Exception("Error en el Digito Verificador")

Next

If Not Me.\_DataSet.Tables("DigitoVerificadorVertical").Rows(0).Item("digito") = Seguridad.DigitoVerificador.CalcularDV(Me.\_DataSet.Tables("Cliente").Rows) Then Throw New Exception("Error en el Digito Verificador")

End Sub

Private Sub GuardarDVVertical()

Dim DataTableAux As New DataTable

ServiciosDAL.Comando.RellenarDT("SELECT digito\_verificador FROM Cliente", DataTableAux)

Me.\_DataSet.Tables("DigitoVerificadorVertical").Rows(0).Item("digito") = Seguridad.DigitoVerificador.CalcularDV(DataTableAux.Rows)

ServiciosDAL.Comando.ActualizarBase("SELECT \* FROM DigitoVerificadorVertical", Me.\_DataSet.Tables("DigitoVerificadorVertical"))

End Sub

Private Sub GuardarDVHorizontal(pObj As Estructura.Cliente)

Dim DataTableAux As New DataTable

ServiciosDAL.Comando.RellenarDT(String.Format("SELECT \* FROM Cliente WHERE id\_cliente = {0}", pObj.Id), DataTableAux)

Dim DataRowAux As DataRow = DataTableAux.Rows(0)

DataRowAux.Item("digito\_verificador") = Seguridad.DigitoVerificador.CalcularDV(Me.DataRow2ArrayList(DataRowAux))

ServiciosDAL.Comando.ActualizarBase("SELECT \* FROM Cliente", DataTableAux)

End Sub

Private Function DataRow2ArrayList(pDataRow As DataRow) As ArrayList

Dim ItemsAux As New ArrayList

ItemsAux.AddRange(pDataRow.ItemArray)

ItemsAux.RemoveAt(ItemsAux.Count - 1)

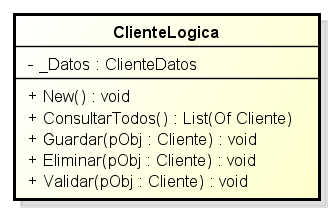
Return ItemsAux

End Function

End Class

## CREACIÓN DE LAS REGLAS DE NEGOCIO (BLL)

En caso de ser necesario, puede implementarse todas las restricciones y lógica de nuestro negocio en esta Capa. Dado que estamos haciendo un ejemplo simple, únicamente funcionará como paso de mensajes. Y una función que valide los datos.



Public Class ClienteLogica

Private \_Datos As New DAL.ClienteDatos

Public Function ConsultarTodos() As List(Of Estructura.Cliente)

Return Me.\_Datos.ConsultarTodos

End Function

Public Sub Guardar(pObj As Estructura.Cliente)

Me.\_Datos.Guardar(pObj)

End Sub

Public Sub Eliminar(pObj As Estructura.Cliente)

Me.\_Datos.Eliminar(pObj)

End Sub

Public Shared Sub Validar(pObj As Estructura.Cliente)

If pObj.Nombre = "" Then Throw New Exception("El Cliente debe tener nombre.")

If pObj.Nombre.Length > 50 Then Throw New Exception("El nombre del Cliente es demasiado largo.")

If pObj.Apellido = "" Then Throw New Exception("El Cliente debe tener apellido.")

If pObj.Apellido.Length > 50 Then Throw New Exception("El nombre del Cliente es demasiado largo.")

If pObj.Telefono.Length > 50 Then Throw New Exception("El telefono del Cliente es demasiado largo.")

If pObj.Celular.Length > 50 Then Throw New Exception("El celular del Cliente es demasiado largo.")

If pObj.Email.Length > 50 Then Throw New Exception("El e-mail del Cliente es demasiado largo.")

End Sub

End Class

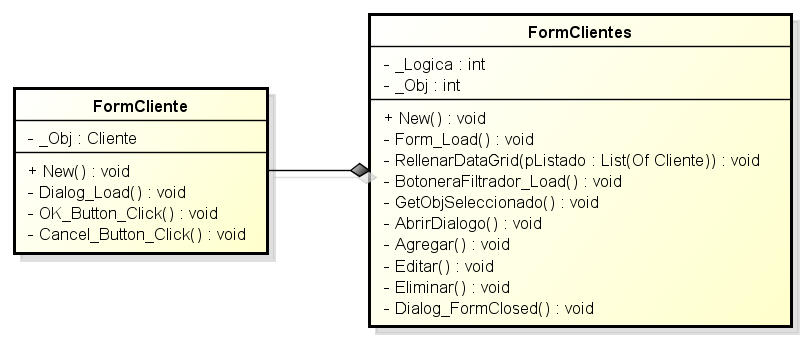
## CREACIÓN DE LA VISTA

La creación de la vista consta de dos Clases fundamentales.

La primera es el formulario principal, donde se encontrará el listado de los Clientes, junto con una barra que ayuda a la manipulación de los mismos.

La segunda clase es un formulario de propiedades, básicamente no hace nada más que pasar la información al objeto, el cual es una referencia del original mantenido siempre desde el formulario principal.

Una vez cerrado este segundo formulario se dispara un evento en el principal, el cual es el encargado de “conversar” con la lógica de la entidad con la cual trabajamos.



Ejemplo de la Implementación:

Public Class FormClientes

Private \_Logica As New BLL.ClienteLogica

Private \_Obj As Estructura.Cliente

Private Sub Form\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

End Sub

Private Sub RellenarDataGrid(pListado As List(Of Estructura.Cliente))

With Me.DataGridPrincipal

.DataSource = Nothing

.Columns.Clear()

.Rows.Clear()

.DataSource = pListado

End With

End Sub

Private Sub BotoneraFiltrador\_Load(sender As Object, e As EventArgs) Handles BotoneraFiltrador.Load

With Me.BotoneraFiltrador

.AgregarBotonAgregar(AddressOf Agregar)

.AgregarBotonEditar(AddressOf Editar)

.AgregarBotonEliminar(AddressOf Eliminar, False)

End With

End Sub

Private Sub GetObjSeleccionado()

Me.\_Obj = Me.DataGridPrincipal.Rows(Me.DataGridPrincipal.SelectedCells(0).RowIndex).DataBoundItem

If IsNothing(Me.\_Obj) Then Throw New Exception("No se ha seleccionado ningún Cliente.")

End Sub

Private Sub AbrirDialogo()

Dim Dialogo As New FormCliente(Me.\_Obj)

With Dialogo

.TopMost = True

.Show()

End With

AddHandler Dialogo.FormClosed, AddressOf Me.Dialog\_FormClosed

End Sub

Private Sub Agregar()

Me.\_Obj = New Estructura.Cliente

Me.AbrirDialogo()

End Sub

Private Sub Editar()

Try

Me.GetObjSeleccionado()

Me.AbrirDialogo()

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub Eliminar()

If Not Alertador.Alertar("¿Está seguro que desea eliminar el Cliente?", "Eliminar", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = Windows.Forms.DialogResult.Yes Then Exit Sub

Try

Me.GetObjSeleccionado()

Me.\_Logica.Eliminar(Me.\_Obj)

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

End Sub

Private Sub Dialog\_FormClosed(sender As Object, e As FormClosedEventArgs)

If DirectCast(sender, Form).DialogResult = Windows.Forms.DialogResult.OK Then

Try

Me.\_Logica.Guardar(Me.\_Obj)

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

End If

End Sub

End Class

Public Class FormCliente

Private \_Obj As Estructura.Cliente

Sub New(ByRef pObj As Estructura.Cliente)

InitializeComponent()

Me.\_Obj = pObj

End Sub

Private Sub Dialog\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

With Me

.TextBoxNombre.Text = .\_Obj.Nombre

.TextBoxApellido.Text = .\_Obj.Apellido

.TextBoxTelefono.Text = .\_Obj.Telefono

.TextBoxCelular.Text = .\_Obj.Celular

.TextBoxEmail.Text = .\_Obj.Email

End With

End Sub

Private Sub OK\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK\_Button.Click

Try

With Me

.\_Obj.Nombre = .TextBoxNombre.Text

.\_Obj.Apellido = .TextBoxApellido.Text

.\_Obj.Telefono = .TextBoxTelefono.Text

.\_Obj.Celular = .TextBoxCelular.Text

.\_Obj.Email = .TextBoxEmail.Text

End With

BLL.ClienteLogica.Validar(Me.\_Obj)

Me.DialogResult = System.Windows.Forms.DialogResult.OK

Me.Close()

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub Cancel\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel\_Button.Click

If MessageBox.Show("¿Está seguro que desea cancelar?", "Cancelar", MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.No Then Exit Sub

Me.DialogResult = System.Windows.Forms.DialogResult.Cancel

Me.Close()

End Sub

End Class

Para comprender la visualización de estos Formularios ver [Anexo de Prototipo de Pantalla](#_Prototipo_de_Pantalla_1).

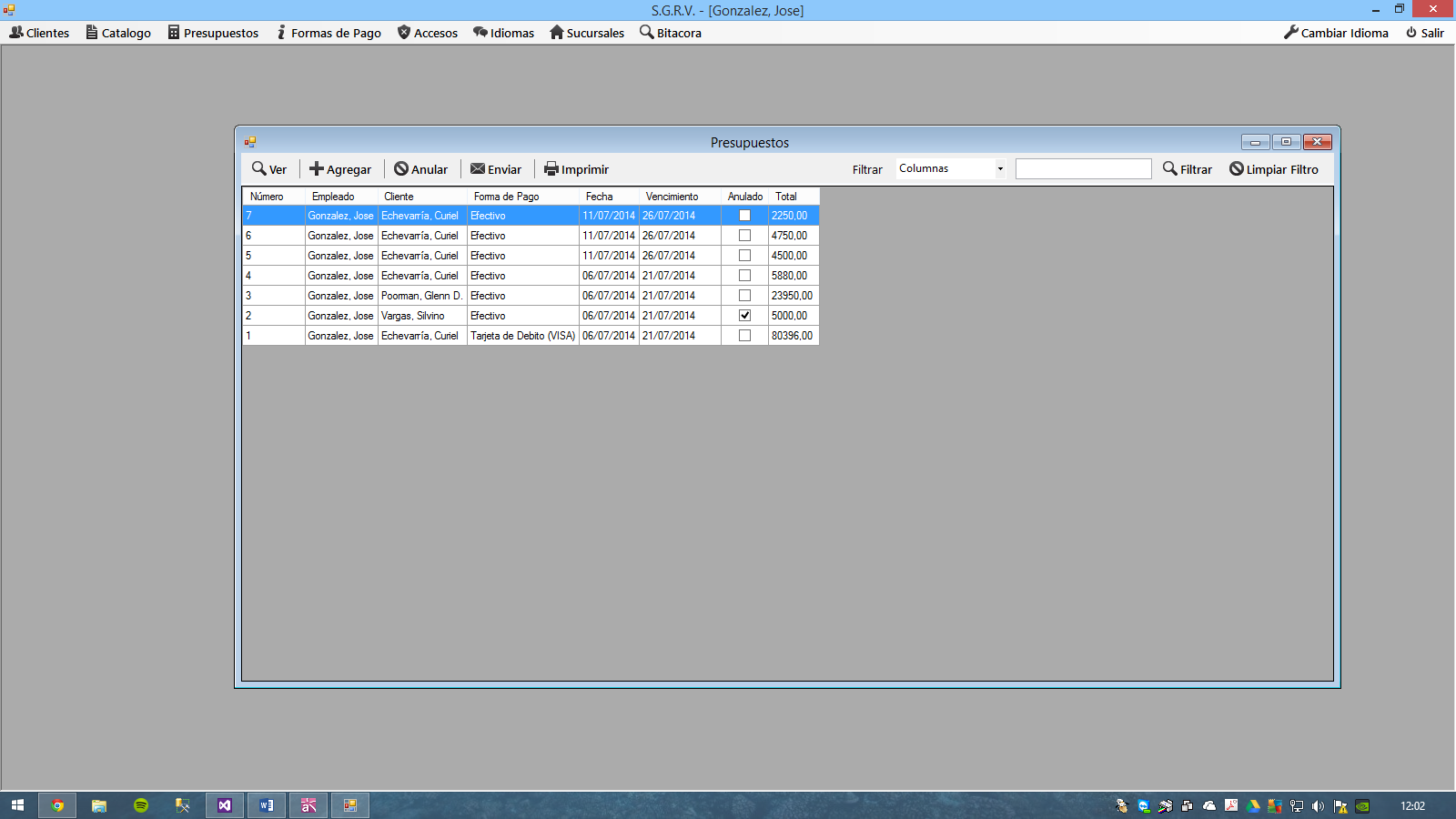
# ANEXOS

## PROTOTIPO DE PANTALLA

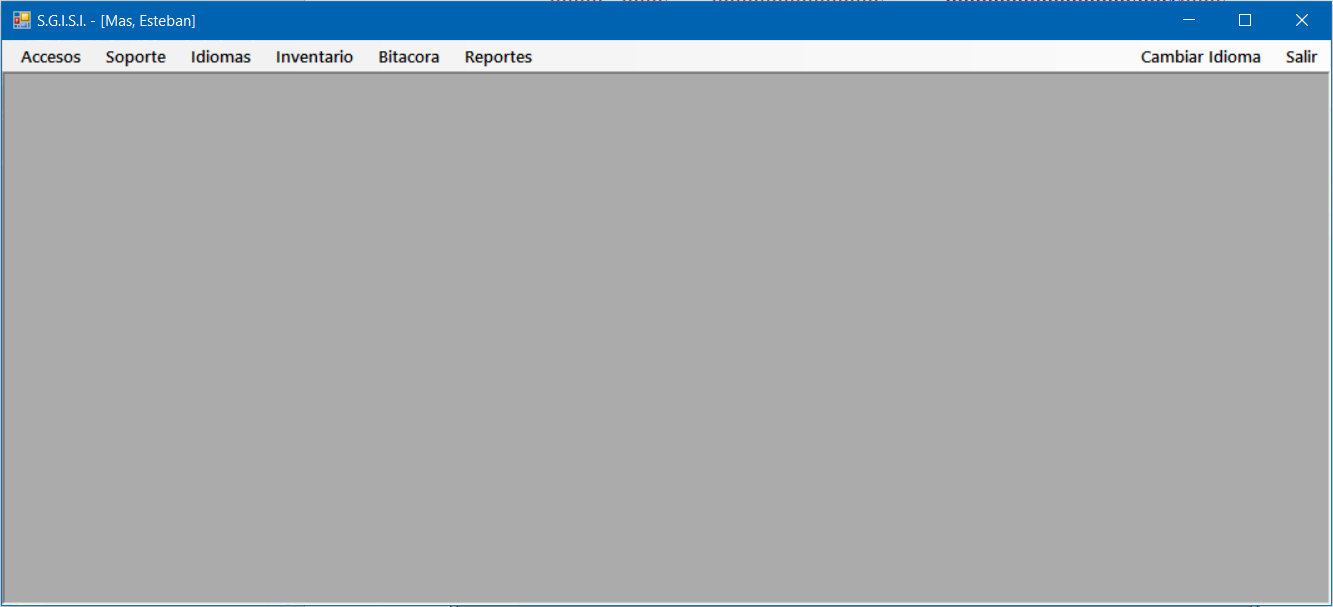
La idea es utilizar formularios MDI, para poder mostrar varias pantallas al mismo tiempo y que el Usuario trabaje cómodamente.

El proyecto cuenta con muchos ABMs, por lo cual se considera tener un componente para repetir en todos los listados, el cual facilite una botonera clásica (Alta, Baja, Modificación, Eliminación, etc) y un Filtrador o Buscador para buscar dentro del listado de la Entidad.

Ejemplo de la Botonera:



Ejemplo de la Pantalla con la Botonera Principal más el Componente.



Public Class FormMain

Implements InterfaceObservador

Private \_Familia As New BLL.FAMILIA

Private \_Idioma As New BLL.IDIOMA

Private \_CPatentes As New CPatenteMultiple(Nothing)

Private Shared \_IdiomaObservado As New IdiomaObservado

Private Sub FormMain\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Me.AgregarEmpleadoTitulo()

Me.AgregarPatentes()

Me.AgregarBotonSalir()

Me.AgregarIdiomas()

\_IdiomaObservado.Suscribir(Me)

End Sub

Private Sub FormMain\_FormClosed(sender As Object, e As FormClosedEventArgs) Handles Me.FormClosed

BLL.EVENTOBITACORA.RegistrarEvento("Salió")

End Sub

Private Sub AgregarEmpleadoTitulo()

Me.Text = String.Format("{0} - [{1}]", Me.Text, BE.Singleton.Instancia.USUARIO)

End Sub

Private Sub AgregarPatentes()

CPatente.RellenarCPatentes(Me.\_CPatentes, Me.\_Familia.ConsultarPatentes(BE.Singleton.Instancia.USUARIO.FAMILIA), AddressOf AbrirForm)

Me.\_CPatentes.Mostrar(Me.MenuStrip1.Items)

End Sub

Private Sub AgregarIdiomas()

Dim IdiomaMenuItem As New ToolStripMenuItem

IdiomaMenuItem.Name = "CambiarIdioma"

IdiomaMenuItem.Text = "Cambiar Idioma"

IdiomaMenuItem.Image = My.Resources.Configuracion

IdiomaMenuItem.Alignment = ToolStripItemAlignment.Right

IdiomaMenuItem.DropDownDirection = ToolStripDropDownDirection.BelowLeft

For Each Idioma As BE.IDIOMA In Me.\_Idioma.ConsultarTodos

Dim NuevoIdioma As New ToolStripMenuItem(Idioma.ToString)

NuevoIdioma.Tag = Idioma

IdiomaMenuItem.DropDownItems.Add(NuevoIdioma)

Next

AddHandler IdiomaMenuItem.DropDownItemClicked, AddressOf Me.CambiarIdioma

Me.MenuStrip1.Items.Add(IdiomaMenuItem)

End Sub

Private Sub AgregarBotonSalir()

Dim SalirMenuItem As New ToolStripMenuItem

SalirMenuItem.Text = "Salir"

SalirMenuItem.Image = My.Resources.Apagar

SalirMenuItem.Alignment = ToolStripItemAlignment.Right

AddHandler SalirMenuItem.Click, AddressOf Me.Salir

Me.MenuStrip1.Items.Add(SalirMenuItem)

End Sub

Private Sub Salir()

Me.Close()

End Sub

Private Sub AbrirForm(sender As Object, e As EventArgs)

Dim FormularioNombre = DirectCast(sender.Tag, BE.PATENTE).Formulario

If FormularioNombre = "" Then Exit Sub

Try

Dim Form As Form = Activator.CreateInstance(Type.GetType(String.Format("{0}.{1}", "GUI", FormularioNombre)))

With Form

.MdiParent = Me

.Show()

End With

FormMain.RegistrarForm(Form)

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Public Shared Sub RegistrarForm(pForm As Form)

If TypeOf (pForm) Is InterfaceObservador Then \_IdiomaObservado.Suscribir(pForm)

End Sub

Public Sub Actualizar(pIdiomaNuevo As BE.IDIOMA, pIdiomaViejo As BE.IDIOMA) Implements InterfaceObservador.Actualizar

ServicioTraduccion.Traducir(Me, pIdiomaNuevo, pIdiomaViejo)

End Sub

Private Sub CambiarIdioma(sender As Object, e As ToolStripItemClickedEventArgs)

\_IdiomaObservado.CambiarIdioma(DirectCast(DirectCast(e.ClickedItem, ToolStripMenuItem).Tag, BE.IDIOMA))

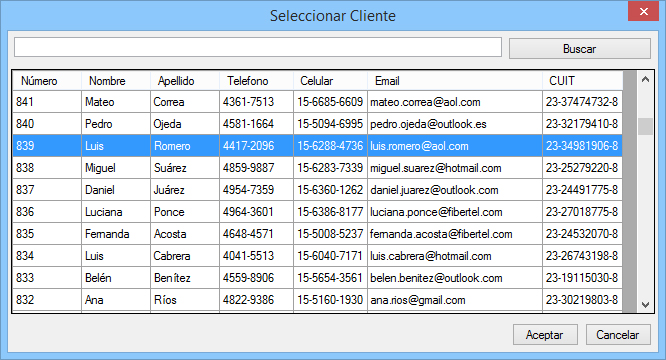
End Sub

End Class

## PROTOTIPO DE BUSCADOR DE CLIENTES

Debido a la gran cantidad de clientes, a veces, es necesario buscar clientes por distintos valores, no solo nombre, apellido, etc.

Para solucionar este problema se propone un buscador interno en cada módulo que necesite utilizar clientes.



Al hacer clic en aceptar un puntero que se pasa por referencia es modificado con el cliente establecido.

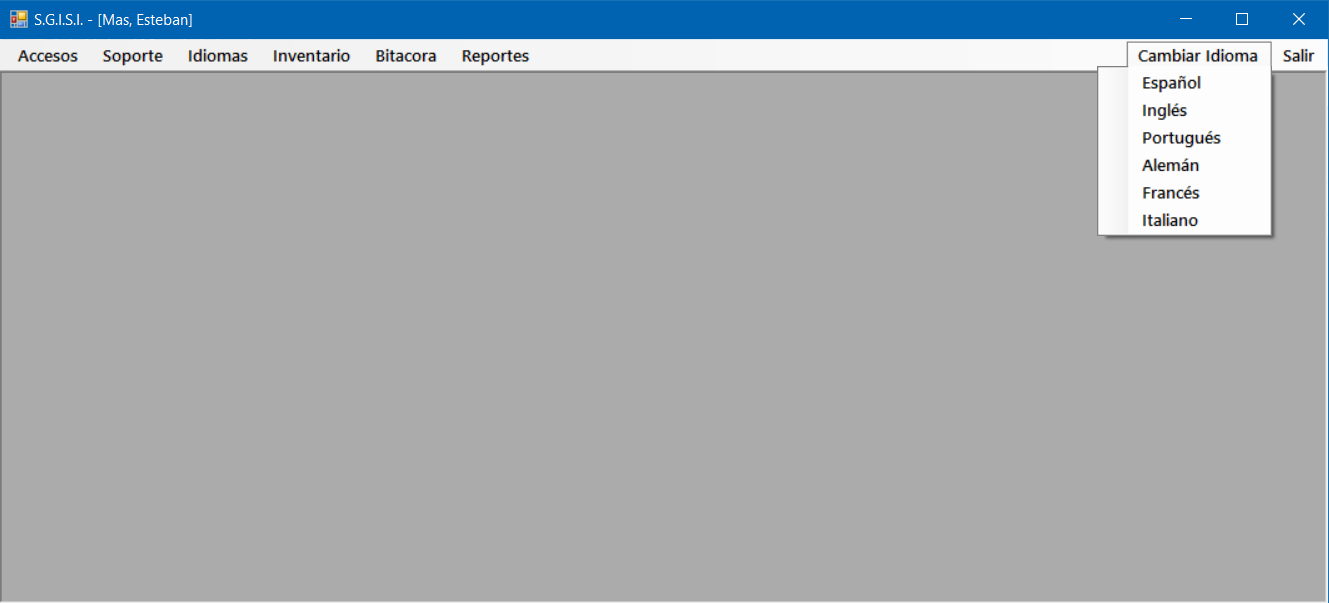


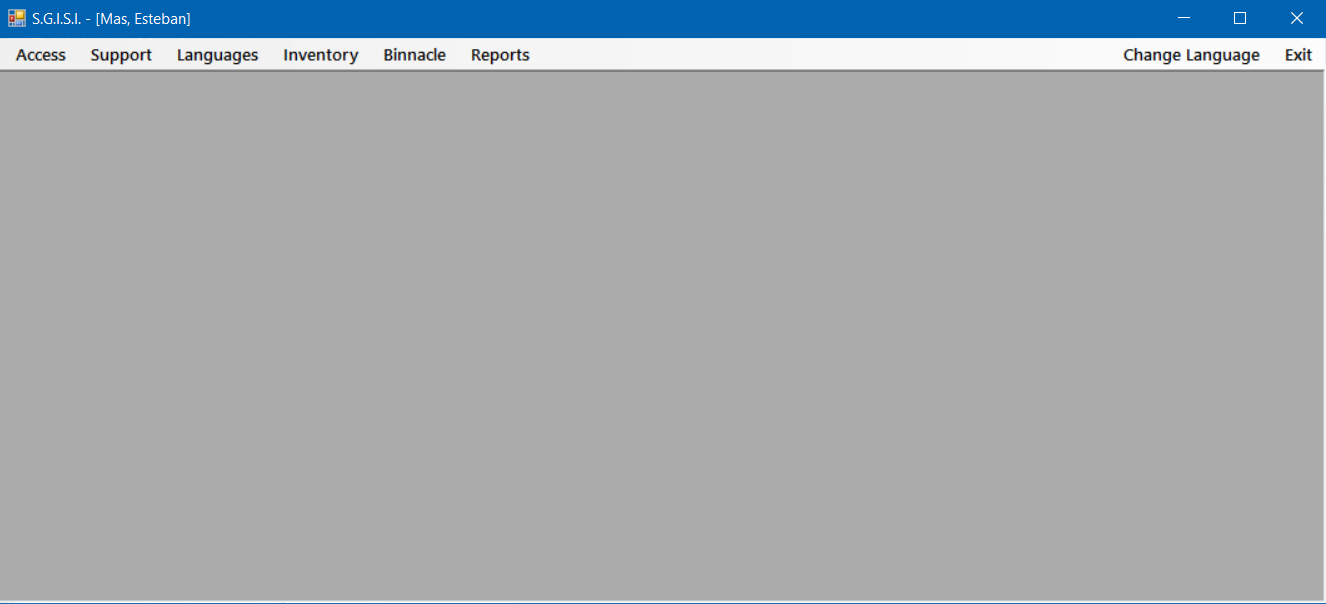
## VALIDACIONES

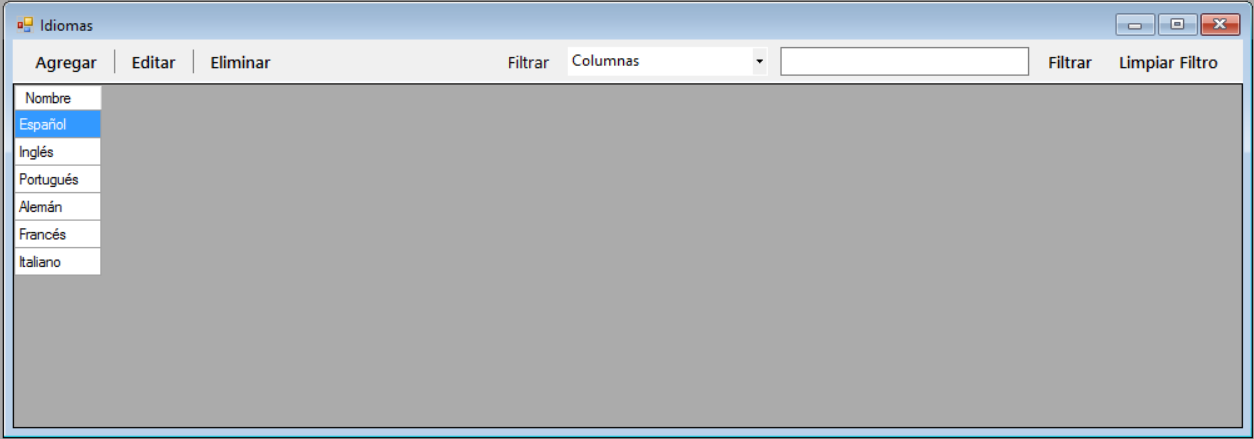
|  |  |  |
| --- | --- | --- |
| Nombre de la Entidad | Nombre del Campo | Validación |
| **Clientes** | nombre | Texto con un largo menor a 50 caracteres. |
|  | apellido | Texto con un largo menor a 50 caracteres. |
|  | teléfono | Texto con un largo menor a 50 caracteres. |
|  | celular | Texto con un largo menor a 50 caracteres. |
|  | email | Texto con un largo menor a 50 caracteres. |
| **Usuarios** | nombre | Texto con un largo menor a 50 caracteres. |
|  | apellido | Texto con un largo menor a 50 caracteres. |
|  | username | Texto con un largo menor a 50 caracteres. |
|  | password | Cualquier largo. Despues se encripta en MD5 |
| **Familias** | nombre | Texto con un largo menor a 50 caracteres. |
| **Idiomas** | nombre | Texto con un largo menor a 50 caracteres. |
| **Palabras Traducidas** | cadena\_traducida | Texto con un largo menor a 50 caracteres. |
| **Gerencias** | nombre | Texto con un largo menor a 50 caracteres. |
|  | ubicación | Texto con un largo menor a 50 caracteres. |

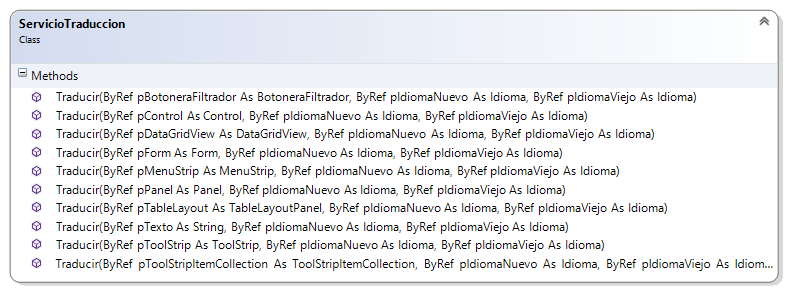
## SERVICIO DE TRADUCCIÓN

Dado que el multilenguaje debe aplicarse a muchos objetos, hemos optado por abstraer ese problema, y simplemente poner Traducir y que el Servicio de Traducción se encargue.  
Para realizar esta tarea utilizamos un código que va deduciendo que tipo de objeto es.









Por Ejemplo:



Aquí llega por primera vez el Form, y comienza (por referencia) a pasar los controles que contiene, este luego es deducido:



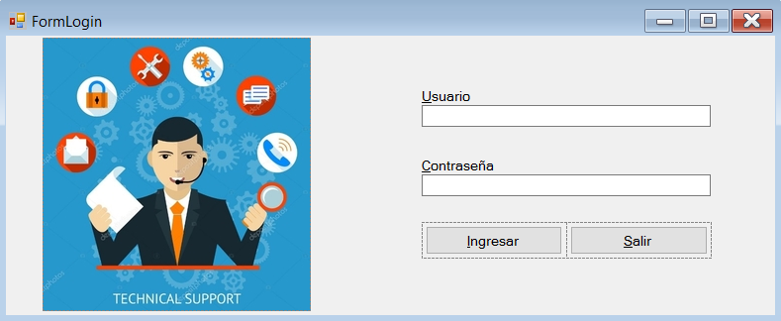
Y ejecutado en su correspondiente bloque de código, gracias a la utilización de múltiples firmas.

Llegando finalmente a traducir el String:



Es importante que no se traduzca si los idiomas coinciden, ya que sería una ejecución sin sentido, y tampoco remplazar cadenas no traducidas por cadenas nulas, lo cual permite al momento de probar el software mostrar las cadenas que no está traducirlas y poder hacerlo.

## LOGIN



Public Class FormLogin

Private \_Logica As New BLL.USUARIO

Private Sub FormLogin\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'[BORRAR]

UsernameTextBox.Text = "admin"

PasswordTextBox.Text = "admin"

'[/BORRAR]

End Sub

Private Sub OK\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK.Click

Try

Me.\_Logica.Login(Me.UsernameTextBox.Text, PasswordTextBox.Text)

Dim FormMain As New FormMain

FormMain.Show()

AddHandler FormMain.FormClosed, AddressOf Me.Show

Me.Hide()

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

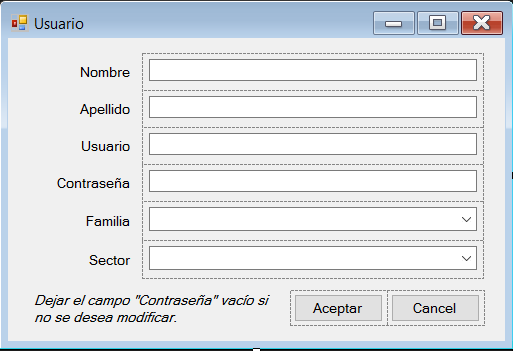
Private Sub Cancel\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel.Click

Me.Close()

End Sub

End Class

## USUARIOS



Public Class FormUsuario

Implements InterfaceObservador

Private \_Obj As BE.USUARIO

Private \_FamiliaLogica As New BLL.FAMILIA

Private \_SucursalLogica As New BLL.SECTOR

Sub New(ByRef pObj As BE.USUARIO)

InitializeComponent()

Me.\_Obj = pObj

End Sub

Private Sub Dialog\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Me.ComboBoxFamilia.DataSource = Me.\_FamiliaLogica.ConsultarTodos()

Me.ComboBoxSector.DataSource = Me.\_SucursalLogica.ConsultarTodos()

With Me

.TextBoxNombre.Text = .\_Obj.Nombre

.TextBoxApellido.Text = .\_Obj.Apellido

.TextBoxUsuario.Text = .\_Obj.Username

End With

If Not Me.\_Obj.Familia Is Nothing Then Me.ComboBoxFamilia.Text = Me.\_Obj.Familia.ToString

If Not Me.\_Obj.SECTOR Is Nothing Then Me.ComboBoxSector.Text = Me.\_Obj.SECTOR.ToString

Me.LabelLeyendaPassword.Visible = IIf(Me.\_Obj.ID\_usuario > 0, True, False)

End Sub

Private Sub OK\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK\_Button.Click

Try

With Me

.\_Obj.Nombre = .TextBoxNombre.Text

.\_Obj.Apellido = .TextBoxApellido.Text

.\_Obj.Username = .TextBoxUsuario.Text

.\_Obj.Password = .TextBoxPassword.Text

.\_Obj.Familia = .ComboBoxFamilia.SelectedValue

.\_Obj.SECTOR = .ComboBoxSector.SelectedValue

End With

BLL.USUARIO.Validar(Me.\_Obj)

Me.DialogResult = System.Windows.Forms.DialogResult.OK

Me.Close()

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub Cancel\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel\_Button.Click

If Alertador.Alertar("¿Está seguro que desea cancelar?", "Cancelar", MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.No Then Exit Sub

Me.DialogResult = System.Windows.Forms.DialogResult.Cancel

Me.Close()

End Sub

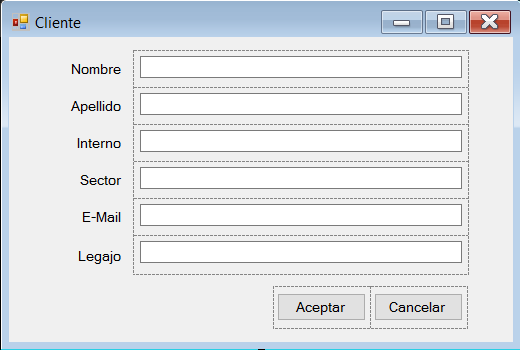
Public Sub Actualizar(pIdiomaNuevo As BE.IDIOMA, pIdiomaViejo As BE.IDIOMA) Implements InterfaceObservador.Actualizar

ServicioTraduccion.Traducir(Me, pIdiomaNuevo, pIdiomaViejo)

End Sub

End Class

## FORM CLIENTE



Public Class FormCliente

Implements InterfaceObservador

Private \_Obj As BE.Cliente

Sub New(ByRef pObj As BE.Cliente)

InitializeComponent()

Me.\_Obj = pObj

End Sub

Private Sub Dialog\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

With Me

.TextBoxNombre.Text = .\_Obj.Nombre

.TextBoxApellido.Text = .\_Obj.Apellido

.TextBoxInterno.Text = .\_Obj.Interno

.TextBoxSector.Text = .\_Obj.Sector

.TextBoxEmail.Text = .\_Obj.Email

.TextBoxLegajo.Text = .\_Obj.Legajo

End With

End Sub

Private Sub OK\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK\_Button.Click

Try

With Me

.\_Obj.Nombre = .TextBoxNombre.Text

.\_Obj.Apellido = .TextBoxApellido.Text

.\_Obj.Interno = .TextBoxInterno.Text

.\_Obj.Sector = .TextBoxSector.Text

.\_Obj.Email = .TextBoxEmail.Text

.\_Obj.Legajo = .TextBoxLegajo.Text

End With

BLL.Cliente.Validar(Me.\_Obj)

Me.DialogResult = System.Windows.Forms.DialogResult.OK

Me.Close()

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub Cancel\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel\_Button.Click

If Alertador.Alertar("¿Está seguro que desea cancelar?", "Cancelar", MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.No Then Exit Sub

Me.DialogResult = System.Windows.Forms.DialogResult.Cancel

Me.Close()

End Sub

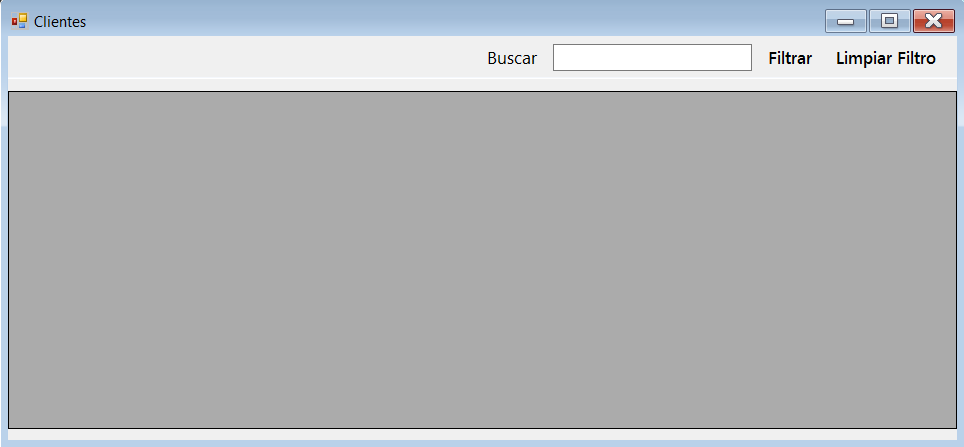
Public Sub Actualizar(pIdiomaNuevo As BE.IDIOMA, pIdiomaViejo As BE.IDIOMA) Implements InterfaceObservador.Actualizar

ServicioTraduccion.Traducir(Me, pIdiomaNuevo, pIdiomaViejo)

End Sub

End Class

## FORM CLIENTES



Public Class FormClientes

Implements InterfaceObservador

Private \_Logica As New BLL.Cliente

Private \_Obj As BE.Cliente

Private Sub Form\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Try

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub RellenarDataGrid(pListado As List(Of BE.Cliente))

With Me.DataGridPrincipal

.DataSource = Nothing

.Columns.Clear()

.Rows.Clear()

.DataSource = pListado

.AutoResizeColumns()

End With

End Sub

Private Sub BotoneraFiltrador\_Load(sender As Object, e As EventArgs) Handles BotoneraFiltrador.Load

With Me.BotoneraFiltrador

.AgregarBotonAgregar(AddressOf Agregar)

.AgregarBotonEditar(AddressOf Editar)

.AgregarBotonEliminar(AddressOf Eliminar, False)

.AgregarManejadorFiltro(AddressOf Me.Filtrar)

End With

End Sub

Private Sub Filtrar()

Try

Me.RellenarDataGrid(Me.\_Logica.Filtrar(Me.BotoneraFiltrador.GetTextoFiltro))

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub GetObjSeleccionado()

Me.\_Obj = Me.DataGridPrincipal.Rows(Me.DataGridPrincipal.SelectedCells(0).RowIndex).DataBoundItem

If IsNothing(Me.\_Obj) Then Throw New Exception("No se ha seleccionado ningún Cliente.")

End Sub

Private Sub AbrirDialogo()

Dim Dialogo As New FormCliente(Me.\_Obj)

With Dialogo

.TopMost = True

.Show()

End With

FormMain.RegistrarForm(Dialogo)

AddHandler Dialogo.FormClosed, AddressOf Me.Dialog\_FormClosed

End Sub

Private Sub Agregar()

Me.\_Obj = New BE.Cliente

Me.AbrirDialogo()

End Sub

Private Sub Editar()

Try

Me.GetObjSeleccionado()

Me.AbrirDialogo()

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub Eliminar()

If Not Alertador.Alertar("¿Está seguro que desea eliminar el Cliente?", "Eliminar", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = Windows.Forms.DialogResult.Yes Then Exit Sub

Try

Me.GetObjSeleccionado()

Me.\_Logica.Eliminar(Me.\_Obj)

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

End Sub

Private Sub Dialog\_FormClosed(sender As Object, e As FormClosedEventArgs)

If DirectCast(sender, Form).DialogResult = Windows.Forms.DialogResult.OK Then

Try

Me.\_Logica.Guardar(Me.\_Obj)

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

End If

End Sub

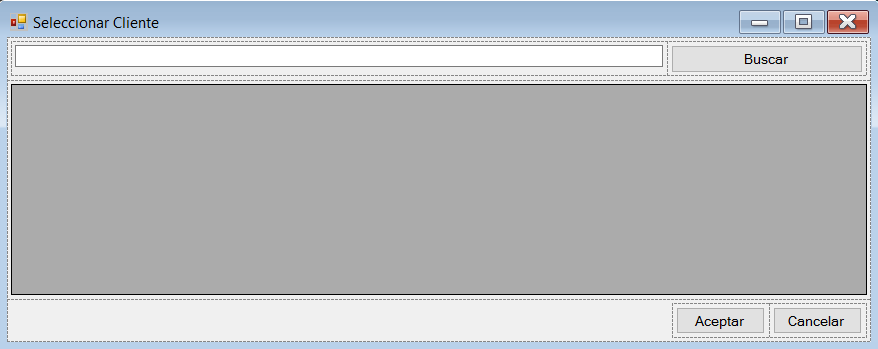
Public Sub Actualizar(pIdiomaNuevo As BE.IDIOMA, pIdiomaViejo As BE.IDIOMA) Implements InterfaceObservador.Actualizar

ServicioTraduccion.Traducir(Me, pIdiomaNuevo, pIdiomaViejo)

End Sub

End Class

## FORM SELECCIONA CLIENTE



Public Class FormSeleccionaCliente

Implements InterfaceObservador

Private \_Logica As New BLL.Cliente

Private \_Obj As BE.Cliente

Sub New(ByRef pCliente As BE.Cliente)

InitializeComponent()

Me.\_Obj = pCliente

End Sub

Private Sub FormSeleccionarClient\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Me.DataGridClientes.DataSource = Me.\_Logica.ConsultarTodos

End Sub

Private Sub OK\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK\_Button.Click

Me.\_Obj.Id = DirectCast(Me.DataGridClientes.Rows(Me.DataGridClientes.SelectedCells(0).RowIndex).DataBoundItem, Estructura.Cliente).Id

Me.DialogResult = System.Windows.Forms.DialogResult.OK

Me.Close()

End Sub

Private Sub Cancel\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel\_Button.Click

If Alertador.Alertar("¿Está seguro que desea cancelar?", "Cancelar", MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.No Then Exit Sub

Me.DialogResult = System.Windows.Forms.DialogResult.Cancel

Me.Close()

End Sub

Public Sub Actualizar(pIdiomaNuevo As BE.IDIOMA, pIdiomaViejo As BE.IDIOMA) Implements InterfaceObservador.Actualizar

ServicioTraduccion.Traducir(Me, pIdiomaNuevo, pIdiomaViejo)

End Sub

Private Sub ButtonBuscar\_Click(sender As Object, e As EventArgs) Handles ButtonBuscar.Click

Dim visible = False

For Each row As DataGridViewRow In Me.DataGridClientes.Rows

visible = False

For Each cell As DataGridViewCell In row.Cells

If cell.Value.ToString.Contains(Me.TextBoxFiltro.Text) Then visible = True

Next

Try

row.Visible = visible

Catch ex As Exception

Continue For

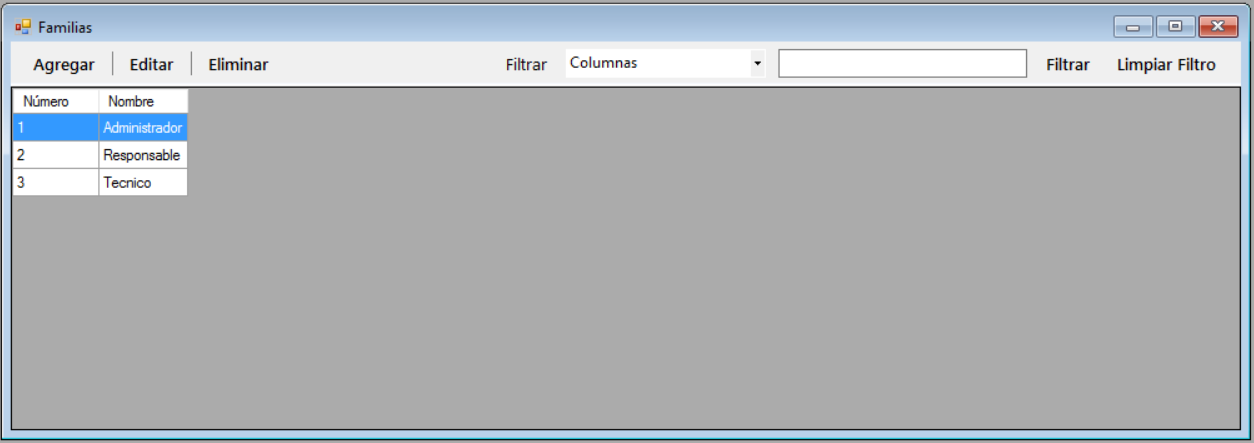
End Try

Next

End Sub

End Class

## FORM FAMILIAS



Public Class FormFamilias

Implements InterfaceObservador

Private \_Logica As New BLL.FAMILIA

Private \_Obj As BE.FAMILIA

Private Sub Form\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

End Sub

Private Sub RellenarDataGrid(pListado As List(Of BE.FAMILIA))

With Me.DataGridPrincipal

.DataSource = Nothing

.Columns.Clear()

.Rows.Clear()

.DataSource = pListado

.AutoResizeColumns()

End With

End Sub

Private Sub BotoneraFiltrador\_Load(sender As Object, e As EventArgs) Handles BotoneraFiltrador.Load

With Me.BotoneraFiltrador

.AgregarBotonAgregar(AddressOf Agregar)

.AgregarBotonEditar(AddressOf Editar)

.AgregarBotonEliminar(AddressOf Eliminar, False)

End With

End Sub

Private Sub GetObjSeleccionado()

Me.\_Obj = Me.DataGridPrincipal.Rows(Me.DataGridPrincipal.SelectedCells(0).RowIndex).DataBoundItem

If IsNothing(Me.\_Obj) Then Throw New Exception("No se ha seleccionado ninguna Familia.")

End Sub

Private Sub AbrirDialogo()

Dim Dialogo As New FormFamilia(Me.\_Obj)

With Dialogo

.TopMost = True

.Show()

End With

FormMain.RegistrarForm(Dialogo)

AddHandler Dialogo.FormClosed, AddressOf Me.Dialog\_FormClosed

End Sub

Private Sub Agregar()

Me.\_Obj = New BE.FAMILIA

Me.AbrirDialogo()

End Sub

Private Sub Editar()

Try

Me.GetObjSeleccionado()

Me.AbrirDialogo()

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub Eliminar()

If Not Alertador.Alertar("¿Está seguro que desea eliminar la Familia?", "Eliminar", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = Windows.Forms.DialogResult.Yes Then Exit Sub

Try

Me.GetObjSeleccionado()

Me.\_Logica.Eliminar(Me.\_Obj)

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

End Sub

Private Sub Dialog\_FormClosed(sender As Object, e As FormClosedEventArgs)

If DirectCast(sender, Form).DialogResult = Windows.Forms.DialogResult.OK Then

Try

Me.\_Logica.Guardar(Me.\_Obj)

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

End If

End Sub

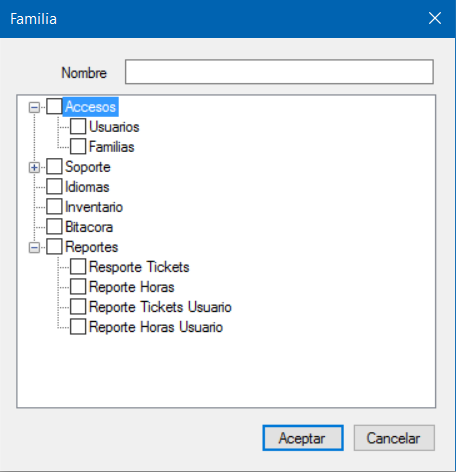
Public Sub Actualizar(pIdiomaNuevo As BE.IDIOMA, pIdiomaViejo As BE.IDIOMA) Implements InterfaceObservador.Actualizar

ServicioTraduccion.Traducir(Me, pIdiomaNuevo, pIdiomaViejo)

End Sub

End Class

### FORM FAMILIA



Imports System.Windows.Forms

Public Class FormFamilia

Implements InterfaceObservador

Private \_Obj As BE.FAMILIA

Private \_Logica As New BLL.PATENTE

Sub New(ByRef pObj As BE.FAMILIA)

InitializeComponent()

Me.\_Obj = pObj

End Sub

Private Sub Dialog\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Dim Patentes As New CPatenteMultiple(Nothing)

CPatente.RellenarCPatentes(Patentes, Me.\_Logica.ConsultarTodos)

Patentes.Mostrar(Me.TreeViewPatentes.Nodes)

Me.TextBoxNombre.Text = Me.\_Obj.Nombre

Me.TildarNodos(Me.TreeViewPatentes.Nodes)

End Sub

Private Sub OK\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK\_Button.Click

Try

Me.\_Obj.Nombre = Me.TextBoxNombre.Text

Me.\_Obj.Patentes.Clear()

Me.AgregarPatentesSeleccionadas(Me.TreeViewPatentes.Nodes)

BLL.FAMILIA.Validar(Me.\_Obj)

Me.DialogResult = System.Windows.Forms.DialogResult.OK

Me.Close()

Catch ex As Exception

Alertador.Alertar(ex.Message)

End Try

End Sub

Private Sub Cancel\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel\_Button.Click

If Alertador.Alertar("¿Está seguro que desea cancelar?", "Cancelar", MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.No Then Exit Sub

Me.DialogResult = System.Windows.Forms.DialogResult.Cancel

Me.Close()

End Sub

Private Sub TildarNodos(ByRef Nodos As TreeNodeCollection)

For Each Nodo As TreeNode In Nodos

For Each Patente As BE.PATENTE In Me.\_Obj.Patentes

If Patente.Id = DirectCast(Nodo.Tag, BE.PATENTE).Id Then

Nodo.Checked = True

End If

Next

If Nodo.Nodes.Count > 0 Then Me.TildarNodos(Nodo.Nodes)

Next

End Sub

Private Sub AgregarPatentesSeleccionadas(Nodes As TreeNodeCollection)

For Each Node As TreeNode In Nodes

If Node.Checked Then

Me.\_Obj.Patentes.Add(Node.Tag)

End If

If Node.Nodes.Count > 0 Then

AgregarPatentesSeleccionadas(Node.Nodes)

End If

Next

End Sub

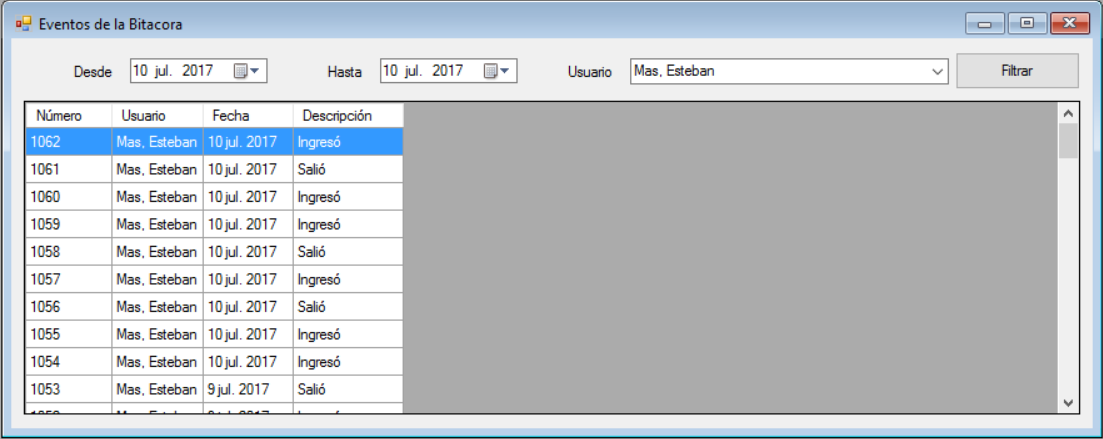
Public Sub Actualizar(pIdiomaNuevo As BE.IDIOMA, pIdiomaViejo As BE.IDIOMA) Implements InterfaceObservador.Actualizar

ServicioTraduccion.Traducir(Me, pIdiomaNuevo, pIdiomaViejo)

End Sub

End Class

## FORM BITÁCORA



Public Class FormEventosBitacora

Implements InterfaceObservador

Private \_Logica As New BLL.EVENTOBITACORA

Private \_Obj As BE.EVENTOBITACORA

Private Sub Form\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Me.RellenarDataGrid(Me.\_Logica.ConsultarTodos)

Dim Usuario As New BLL.USUARIO

Me.ComboBoxUsuario.DataSource = Usuario.ConsultarTodos

End Sub

Private Sub RellenarDataGrid(pListado As List(Of BE.EVENTOBITACORA))

With Me.DataGridPrincipal

.DataSource = Nothing

.Columns.Clear()

.Rows.Clear()

.DataSource = pListado

.AutoResizeColumns()

End With

End Sub

Private Sub ButtonFiltrar\_Click(sender As Object, e As EventArgs) Handles ButtonFiltrar.Click

Me.RellenarDataGrid(Me.\_Logica.Filtrar(Me.DateTimeDesde.Text, Me.DateTimeHasta.Text, Me.ComboBoxUsuario.SelectedValue))

Me.DataGridPrincipal.AutoResizeColumns()

End Sub

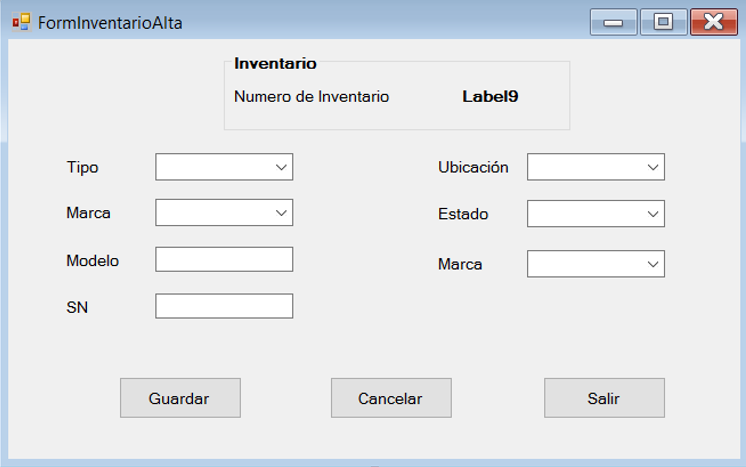
Public Sub Actualizar(pIdiomaNuevo As BE.IDIOMA, pIdiomaViejo As BE.IDIOMA) Implements InterfaceObservador.Actualizar

ServicioTraduccion.Traducir(Me, pIdiomaNuevo, pIdiomaViejo)

End Sub

End Class

## FORM INVENTARIO ALTA



# REPOSITORIO GITHUB

<https://cuervo1974.github.io/2017_UAI_TCD_58223_MAS/>

# BIBLIOGRAFÍA

Centralización de Datos

<http://manuelgross.bligoo.com/content/view/721661/Organizacion-101-Ventajas-y-desventajas-de-la-centralizacion.html>

Arquitectura en Capas

Orientación a objetos, teoría y práctica – Grady Booch, Darío Cardacci – Capitulo 6

<http://msdn.microsoft.com/en-us/library/aa581778.aspx>

<http://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas>

http://msdn.microsoft.com/en-us/library/aa581780.aspx

<http://es.wikipedia.org/wiki/Mapeo_objeto-relacional>

<http://msdn.microsoft.com/es-es/library/aa288587(v=vs.71).aspx>

<http://es.wikipedia.org/wiki/Modelo_Vista_Controlador>

<http://msdn.microsoft.com/en-us/library/xyhh2e7e(v=vs.110).aspx>

Documentos

<http://es.wikipedia.org/wiki/Presupuesto>

<http://es.wikipedia.org/wiki/Factura>

Diagrama de Despliegue

<http://es.wikipedia.org/wiki/Diagrama_de_despliegue>

<http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html>

Diagrama de Componentes

<http://es.wikipedia.org/wiki/Diagrama_de_componentes>

<http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html>

Objetos inmersos en los DataGrid

<http://msdn.microsoft.com/es-es/library/system.windows.forms.datagridviewrow.databounditem(v=vs.110).aspx>

Escenarios

<http://wiki.fluxit.com.ar/display/PUBLIC/Escenarios+de+Casos+de+Uso>

Manejo del Software (Enterprise Architect)

<http://www.sparxsystems.com.au/resources/uml2_tutorial/>

<http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_usecasediagram.html>

<http://www.sparxsystems.com/enterprise_architect_user_guide/8.0/modeling_languages/sequencediagram.html>

<http://www.youtube.com/watch?v=FzPET2TCpDQ>

<http://www.youtube.com/watch?v=sYUJq_JLkR0>

Manejo de Software (ASTAH)

<http://astah.net/>.

<http://astah.net/videos>

<http://astah.net/tutorials>

Equivalencias de tipos SQL a .NET (Validación)

<http://technet.microsoft.com/en-us/library/ms131092(v=sql.110).aspx>

Seguridad

<https://crackstation.net/hashing-security.htm>

<http://msdn.microsoft.com/en-us/library/system.security.cryptography.rngcryptoserviceprovider.asp>

<http://www.miraclesalad.com/webtools/md5.php>

Typeahead

<http://stackoverflow.com/questions/9934634/type-ahead-combo>

Layer Supertipe

<http://prashantbrall.wordpress.com/2011/10/25/layer-supertype-pattern/>

Conexión en APP.config

<http://www.codeproject.com/Tips/416198/How-to-get-Connection-String-from-App-Config-in-Cs>

Creación de PDFs desde .NET (Librería)

<http://www.pdfsharp.net/MigraDocOverview.ashx> (Librería)

<http://www.pdfsharp.net/wiki/Default.aspx> (Documentación)