**Universidad Abierta Interamericana**

Facultad de Tecnología informática

Ingeniería en Sistemas Informáticos

Trabajo de Campo

*EvOrg*

*Sistema de Organización de Eventos*

Profesor: Esteban Calabria

Alumno: Facundo Tripelhorn

Curso: 3º A

Turno: Mañana

Sede: Centro

**2017**

Índice

[Historial de revisión 5](#_Toc487911486)

[Descripción global del producto 6](#_Toc487911487)

[Propósito 6](#_Toc487911488)

[Descripción funcional del producto y alcance 6](#_Toc487911489)

[Definiciones, acrónimos y abreviaciones 6](#_Toc487911490)

[Descripción de las personas participantes en el desarrollo del sistema de información y los usuarios 7](#_Toc487911491)

[Diagrama de actividad 8](#_Toc487911492)

[Listado de Casos de Uso 10](#_Toc487911493)

[Negocio 10](#_Toc487911494)

[Sistema 10](#_Toc487911495)

[Especificaciones de Casos de Uso 11](#_Toc487911496)

[Diagrama de Caso de Uso ABM Entidad 11](#_Toc487911497)

[Caso de Uso 001 – Alta de Entidad 12](#_Toc487911498)

[Historial de Revisión 12](#_Toc487911499)

[Descripción Detallada 12](#_Toc487911500)

[Diagrama de Secuencia 13](#_Toc487911501)

[Caso de uso 002 – Baja de Entidad 14](#_Toc487911502)

[Historial de Revisión 14](#_Toc487911503)

[Descripción Detallada 14](#_Toc487911504)

[Diagrama de secuencia 15](#_Toc487911505)

[Caso de Uso 003 – Modificación de Entidad 16](#_Toc487911506)

[Historial de Revisión 16](#_Toc487911507)

[Descripción Detallada 16](#_Toc487911508)

[Diagrama de Secuencia 17](#_Toc487911509)

[Especificaciones de Casos de Uso – ABM de Entidad 18](#_Toc487911510)

[Pantallas Casos de Uso ABM 20](#_Toc487911511)

[Diagrama de Clases de Entidades ABM 28](#_Toc487911512)

[Diagrama Entidad-Relación de Entidades ABM 29](#_Toc487911513)

[Caso de Uso 004 – Ingresar al sistema 30](#_Toc487911514)

[Historial de Revisión 30](#_Toc487911515)

[Descripción Detallada 30](#_Toc487911516)

[Diagrama de Caso de Uso 31](#_Toc487911517)

[Diagrama de Secuencia 31](#_Toc487911518)

[Diagrama de Clases 32](#_Toc487911519)

[Diagrama Entidad-Relación 33](#_Toc487911520)

[Pantalla 33](#_Toc487911521)

[Caso de Uso 005 – Asignar Familias y Patentes 34](#_Toc487911522)

[Historial de Revisión 34](#_Toc487911523)

[Descripción Detallada 34](#_Toc487911524)

[Diagrama de Caso de Uso 34](#_Toc487911525)

[Diagrama de Secuencia 35](#_Toc487911526)

[Diagrama de Clases 36](#_Toc487911527)

[Diagrama Entidad-Relación 37](#_Toc487911528)

[Pantalla 38](#_Toc487911529)

[Patrón Composite 39](#_Toc487911530)

[Código de Implementación del Patrón Composite 40](#_Toc487911531)

[Diagrama de Clases del Patrón Composite 45](#_Toc487911532)

[Diagrama Entidad-Relación del patrón Composite 46](#_Toc487911533)

[Caso de Uso 006 – Encriptar 47](#_Toc487911534)

[Historial de Revisión 47](#_Toc487911535)

[Descripción Detallada 47](#_Toc487911536)

[Diagrama de Caso de Uso 47](#_Toc487911537)

[Diagrama de Secuencia 48](#_Toc487911538)

[Diagrama de Clases 48](#_Toc487911539)

[Diagrama Entidad-Relación 48](#_Toc487911540)

[Caso de Uso 007 – Cambiar Idioma 49](#_Toc487911541)

[Historial de Revisión 49](#_Toc487911542)

[Descripción Detallada 49](#_Toc487911543)

[Diagrama de Caso de Uso 49](#_Toc487911544)

[Diagrama de Secuencia 50](#_Toc487911545)

[Diagrama de Clases 51](#_Toc487911546)

[Diagrama Entidad-Relación 52](#_Toc487911547)

[Pantalla 53](#_Toc487911548)

[Patrón Observer 54](#_Toc487911549)

[Interfaz IObservador 54](#_Toc487911550)

[Interfaz IObservado 54](#_Toc487911551)

[Clase Traductor 55](#_Toc487911552)

[Diagrama de Clases del Patrón Observer 57](#_Toc487911553)

[Especificación de Aspectos Técnicos del Sistema 57](#_Toc487911554)

[Historial de Revisión 57](#_Toc487911555)

[Gestión de perfiles de Usuario 57](#_Toc487911556)

[Gestión de Inicio de Sesión del Sistema 58](#_Toc487911557)

[Gestión de Múltiples Idiomas 58](#_Toc487911558)

[Gestión de Encriptado 59](#_Toc487911559)

[Diagrama de Clases de Aspectos Técnicos del Sistema 60](#_Toc487911560)

[Diagrama Entidad-Relación de Aspectos Técnicos del Sistema 61](#_Toc487911561)

[Pantallas del Sistema 62](#_Toc487911562)

[Inicio de sesión 62](#_Toc487911563)

[Pantalla principal 62](#_Toc487911564)

[Organizar Evento 63](#_Toc487911565)

[Reservar material 64](#_Toc487911566)

[Ver materiales 64](#_Toc487911567)

[Disponibilidad de materiales 65](#_Toc487911568)

[Reservar servicio 65](#_Toc487911569)

[Seleccionar salón 66](#_Toc487911570)

[Ver eventos 66](#_Toc487911571)

[Generar presupuesto 67](#_Toc487911572)

[Ver tareas del día 67](#_Toc487911573)

[ABM Clientes 68](#_Toc487911574)

[ABM Materiales 68](#_Toc487911575)

[ABM Salones 69](#_Toc487911576)

[ABM Servicios 70](#_Toc487911577)

[ABM Tipos de evento 70](#_Toc487911578)

[ABM Familias 71](#_Toc487911579)

[ABM Patentes 71](#_Toc487911580)

[Nueva patente 72](#_Toc487911581)

[ABM Usuarios 72](#_Toc487911582)

[Nuevo usuario 73](#_Toc487911583)

[Gestionar idiomas 73](#_Toc487911584)

[Anexo 74](#_Toc487911585)

[Arquitectura en 6 capas 74](#_Toc487911586)

[Capa DAL 75](#_Toc487911587)

[Capa DAL\_ABMC 76](#_Toc487911588)

[Capa BLL\_Estatica 79](#_Toc487911589)

[Capa BLL\_Dinamica 81](#_Toc487911590)

[Capa Framework 82](#_Toc487911591)

[Capa GUI 83](#_Toc487911592)

[Cómo bajar el proyecto de GitHub 91](#_Toc487911593)

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Entregable | Descripción |
| 13/4/2017 | 1.0 | Documento de visión | Creación del documento |
| 24/4/2017 | 1.1 | Entidades | Se agregan las entidades Usuario, Familia, Patente y GrupoPatente |
| 24/4/2017 | 1.1 | PantallasABM | Se agregan las pantallas de Usuario, Familia y Patente |
| 5/5/2017 | 1.2 | Caso de Uso | Creación de los casos de uso LogIn, Asignar Familias y Patentes, Encriptar |
| 5/5/2017 | 1.2 | PantallasABM | Se actualizan las pantallas |
| 4/6/2017 | 1.3 | Diagrama de actividad | Se actualiza el diagrama de actividad del sistema |
| 4/6/2017 | 1.3 | Caso de Uso | Se actualiza el caso de uso de ABM Genérico |
| 7/7/2017 | 1.4 | Caso de Uso | Se agrega el caso de uso referido al MultiIdioma |
| 14/7/2017 | 1.5 | Especificación de Aspectos Técnicos del Sistema | Se crea la sección |
| 14/7/2017 | 1.5 | Anexo | Se crea la sección |
| 14/7/2017 | 1.5 | Patrón Composite | Se crea la sección |
| 14/7/2017 | 1.5 | Patrón Observer | Se crea la sección |
| 14/7/2017 | 1.5 | Caso de Uso | Se agregan diagrama de clases y DER de las entidades involucradas en ABM genérico |
| 14/7/2017 | 1.5 | Anexo | Se agrega tutorial de como bajar el proyecto de GitHub |
| 15/7/2017 | 1.5 | Pantallas del sistema | Se agregan las pantallas del sistema |

Historial de revisión

Descripción global del producto

Propósito

Organizar un evento es encargarse de la reserva del salón, la contratación de todos los servicios necesarios, la decoración del salón y la limpieza del mismo luego de que el evento haya terminado. Por lo general, la iluminación, el sonido, el catering, el fotógrafo y el bar no están incluidos en el servicio principal, ya que son servicios extra por el hecho de que se contrata a terceros de confianza para realizarlos. El propósito de este EvOrg es que quien se encargue de esto, tenga las herramientas necesarias para que la organización del evento resulte de la mejor manera posible.

Descripción funcional del producto y alcance

Este sistema se propone administrar su empresa de organización de eventos de forma tal que usted pueda administrar sus tiempos y posea los materiales de trabajo siempre disponibles en caso de necesitar usarlos. El sistema también le ofrecerá consejos de optimización tales como la compra de material extra para que pueda trabajar mejor sin depender completamente de los tiempos de otros para aceptar o rechazar trabajos. También le recordará cuando deba retirar el material luego de ser lavado y le informará qué actividades debe realizar cada día.

Al momento de seleccionar un salón, el sistema filtrará los salones disponibles dependiendo de los datos del evento. Al verificar la lista de materiales, se podrá saber que materiales estarán disponibles en determinada fecha. En caso de no estar disponibles, el sistema indicará la fecha en la cual se podrá disponer de los materiales.

Además, cada evento poseerá una seguidilla de pasos a seguir, la cual le ayudará a avanzar según lo planeado.

La primera vez que se ingrese al sistema el usuario deberá crearse una cuenta y a la vez cargar los datos de la empresa.

Luego de haber ingresado, el sistema le proveerá algunos datos precargados. Estos serán, una lista de tipos de evento, pasos a seguir dependiendo del tipo de evento seleccionado, lista de servicios tercerizados, lista de salones y lista de pasos a realizar comunes en todo evento.

Por último, al finalizar cada mes el sistema le proveerá un análisis de su trabajo indicándole que cosas debe mejorar para brindar un mejor servicio

Definiciones, acrónimos y abreviaciones

Definiciones:

* Evento: suceso de importancia que se encuentra programado. Dicho acontecimiento puede ser social, artístico o deportivo.
* Tercerización o Subcontratación: práctica llevada a cabo por una empresa cuando contrata a otra firma para que preste un servicio que, en un principio, debería ser brindado por ella misma.
* Catering: consiste en un servicio de alimentación destinado a un conjunto más o menos grande de personas. Quien ofrece el catering sirve un cierto tipo y una cantidad de comida ya estipulada.
* Salón: aposento de grandes dimensiones en una casa que se destina a la recepción de las visitas o para albergar fiestas y reuniones.

Descripción de las personas participantes en el desarrollo del sistema de información y los usuarios

* Usuario (Rol Primario): Es el consumidor final. El usuario puede acceder a todas las funcionalidades que provee el sistema. También se encarga de las actividades externas al sistema.
* Administrador del sistema (Rol Primario): Persona que se encarga de administrar y mantener el sistema. Es el encargado de realizar altas, bajas y modificaciones de las entidades del sistema, acceder a la bitácora y a las copias de seguridad. También es quien se encargará de ingresar los datos que se encuentren precargados al iniciar el sistema por primera vez. Pero no podrá utilizar las funcionalidades que provee el sistema

Diagrama de actividad

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 13/4/2017 | 1.0 | Creación del diagrama de actividad |
| 4/6/2017 | 1.3 | Actualización del diagrama, la verificación del pago pasa a ser un proceso externo |



Listado de Casos de Uso

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 13/4/2017 | 1.0 | Creación del listado |

Negocio

1. **Organizar evento**
2. **Seleccionar salón para evento**
3. **Reservar material para evento**
4. **Actualización de stock**
5. **Modificar lista de materiales**
6. **Generar presupuesto**
7. **Reservar servicios**
8. **Ver lista de materiales**
9. **Ver evento**
10. **Ver lista de eventos**
11. **Ver tareas del día**

Sistema

1. **Gestión de Perfiles de Usuario**
2. **Gestión de Log In / Log Out**
3. **Gestión de Múltiples Idiomas**
4. **Gestión de Bitácora**
5. **Gestión de Back Up**
6. **Gestión de Dígitos Verificadores**
7. **Gestión de Encriptado**

Especificaciones de Casos de Uso

Diagrama de Caso de Uso ABM Entidad



Caso de Uso 001 – Alta de Entidad

Historial de Revisión

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 13/4/2017 | 1.0 | Creación del caso de uso |

Descripción Detallada

|  |  |
| --- | --- |
| **1 - Nombre:** | **CUSO 001 – Alta de entidad** |
| Objetivo: | Se realiza una operación Alta sobre una entidad. |
| Actores: | Administrador del sistema. |
| **2 - Flujo de Eventos** | |
| Disparador/Condición: | Necesidad de crear una nueva entidad. |
| Flujo Básico: | |
|  | 1. El administrador ingresa al formulario “ABM” de la entidad deseada. 2. El administrador presiona el botón para crear una nueva entidad. 3. El sistema solicita el ingreso de datos necesarios. 4. El administrador ingresa los datos y acepta. 5. El sistema valida los datos ingresados. |
| Flujos Alternativos: | |
|  | 1. La entidad ingresada ya existe.    1. El sistema deberá notificar que la entidad ingresada ya existe.    2. Se requerirá el ingreso de datos nuevamente.    3. Se vuelve al punto 3 del Escenario Principal 2. Los datos ingresados no son válidos.    1. El sistema deberá informar que datos no son aceptados.    2. Se vuelve al punto 3 del Escenario Principal. |
| **3 - Requerimientos Especiales** | |
| **4 - Precondiciones:** | El administrador debe ingresar al sistema con su cuenta. |
| **5 - Postcondiciones:** | Una nueva entidad se dio de alta. |
| **6 - Puntos de Extensión** |  |
| **7 - Puntos de Inclusión.** |  |

Diagrama de Secuencia



Caso de uso 002 – Baja de Entidad

Historial de Revisión

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 13/4/2017 | 1.0 | Creación del caso de uso |

Descripción Detallada

|  |  |
| --- | --- |
| **1 - Nombre:** | **CUSO 002 – Baja de entidad** |
| Objetivo: | Se realiza una operación Baja sobre una entidad. |
| Actores: | Administrador del sistema. |
| **2 - Flujo de Eventos** | |
| Disparador/Condición: | Necesidad de crear una nueva entidad. |
| Flujo Básico: | |
|  | 1. El administrador ingresa al formulario “ABM” de la entidad deseada. 2. El administrador selecciona una entidad a eliminar de la lista. 3. El administrador presiona el botón para eliminar la entidad seleccionada. 4. El sistema solicita la confirmación 5. El administrador acepta. |
| Flujos Alternativos: | |
|  | 1. El administrador no confirma la eliminación    1. Se vuelve al punto 2 del escenario principal |
| **3 - Requerimientos Especiales** | |
| **4 - Precondiciones:** | El administrador debe ingresar al sistema con su cuenta. |
| **5 - Postcondiciones:** | Una entidad se dio de baja. |
| **6 - Puntos de Extensión** |  |
| **7 - Puntos de Inclusión.** |  |

Diagrama de secuencia



Caso de Uso 003 – Modificación de Entidad

Historial de Revisión

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 13/4/2017 | 1.0 | Creación del caso de uso |

Descripción Detallada

|  |  |
| --- | --- |
| **1 - Nombre:** | **CUSO 003 – Modificación de entidad** |
| Objetivo: | Se realiza una operación de Modificación sobre una entidad. |
| Actores: | Administrador del sistema. |
| **2 - Flujo de Eventos** | |
| Disparador/Condición: | Necesidad de editar una entidad. |
| Flujo Básico: | |
|  | 1. El administrador ingresa al formulario “ABM” de la entidad deseada. 2. El administrador selecciona una entidad a modificar y presiona modificar. 3. El sistema solicita el ingreso de datos nuevos. 4. El administrador ingresa los datos y acepta. 5. El sistema valida los datos ingresados. 6. El sistema encripta los datos a almacenar. 7. El sistema calcula el dígito verificador de las respectivas tablas y lo guarda. 8. El sistema registra en la bitácora la modificación correspondiente. |
| Flujos Alternativos: | |
|  | 1. Los nuevos datos corresponden a otra entidad ya existente.    1. El sistema deberá notificar que los datos ingresados ya existen.    2. Se requerirá el ingreso de datos nuevamente.    3. Se vuelve al punto 3 del Escenario Principal 2. Los datos ingresados no son válidos.    1. El sistema deberá informar que datos no son aceptados.    2. Se vuelve al punto 3 del Escenario Principal. |
| **3 - Requerimientos Especiales** | |
| **4 - Precondiciones:** | El administrador debe ingresar al sistema con su cuenta. |
| **5 - Postcondiciones:** | Una entidad ha sido de modificada. |
| **6 - Puntos de Extensión** |  |
| **7 - Puntos de Inclusión.** |  |

Diagrama de Secuencia



Especificaciones de Casos de Uso – ABM de Entidad

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 13/4/2017 | 1.0 | Creación del listado |
| 24/4/2017 | 1.1 | Se agregan las entidades Usuario, Familia, Patente y GrupoPatente |
| 4/6/2017 | 1.3 | Se actualizan las propiedades de las entidades |

Las entidades a las que se les puede realizar un ABM son: Cliente, Dirección, Familia, Grupo Patente, Material, Paso, Patente, Salón, Servicio, Tipo de Evento, Usuario.

Atributos de cada entidad:

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Cliente | DNI | Numérico | No puede ser duplicado, no debe estar vacío |
| Nombre | Texto | No debe estar vacío, tiene que empezar con mayúscula |
| Apellido | Texto | No debe estar vacío, tiene que empezar con mayúscula |
| Teléfono | Numérico | Debe contener 10 caracteres |
| Email | Texto | Debe tener formato aaa@bbb.ccc |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Dirección | Id | Numérico | No puede ser duplicado, no debe estar vacío |
| Calle | Texto | No puede ser duplicado, no debe estar vacío |
| Número | Numérico |  |
| Localidad | Texto |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Familia | Id | Numérico | No puede ser duplicado, no debe estar vacío |
| Nombre | Texto | No puede ser duplicado, no debe estar vacío |
| Patente | Lista |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Grupo Patente | Id | Numérico | No puede ser duplicado, no debe estar vacío |
| Nombre | Texto | No puede ser duplicado, no debe estar vacío |
| Padre | Numérico |  |
| Patentes | Lista |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Material | Id | Numérico | No puede ser duplicado, no debe estar vacío |
| Nombre | Texto | No debe estar vacío |
| Cantidad | Numérico |  |
| Precio | Numérico |  |
| FechaDesde | Fecha |  |
| FechaHasta | Fecha |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Paso | Id | Numérico | No puede ser duplicado, no debe estar vacío |
| Descripción | Texto | No debe estar vacío |
| Fecha | Fecha |  |
| Prioridad | Texto | Debe elegir entre tres opciones: Baja, Media o Alta |
| Tipo | Texto | Debe elegir entre dos opciones: Concreto o Genérico |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Patente | Id | Numérico | No puede ser duplicado, no debe estar vacío |
| Nombre | Texto | No puede ser duplicado, no debe estar vacío |
| Formulario | Texto |  |
| Padre | Numérico |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Salón | Id | Numérico | No puede ser duplicado, no debe estar vacío |
| Nombre | Texto | No debe estar vacío |
| Capacidad | Numérico | No debe estar vacío |
| Email | Texto | Debe tener formato aaa@bbb.ccc |
| Teléfono | Numérico | Debe contener 10 caracteres |
| Dirección | Dirección | Debe estar compuesta de calle, número y código postal |
| Descripción | Texto |  |
| Precio | Numérico |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Servicio | Id | Numérico | No puede ser duplicado, no debe estar vacío |
| Nombre | Texto | No debe estar vacío |
| Teléfono | Numérico | Debe contener 10 caracteres |
| Dirección | Texto | Debe estar compuesta de calle, número y código postal |
| Email | Texto | Debe tener formato [aaa@bbb.ccc](mailto:aaa@bbb.ccc) |
| Observación | Texto |  |
| Precio | Numérico |  |

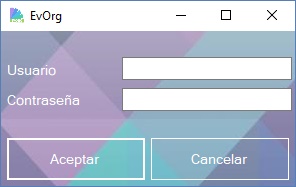
|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Tipo de Evento | Id | Numérico | No debe estar vacío. Debe ser único |
| Nombre | Texto | No debe estar vacío |
| Descripción | Texto |  |
| Pasos | Lista |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Tipo de dato | Observación |
| Usuario | Id | Texto | No puede ser duplicado, no debe estar vacío |
| Email | Texto | Debe tener formato [aaa@bbb.ccc](mailto:aaa@bbb.ccc) |
| Contraseña | Texto | Debe contener entre 8 y 16 caracteres |
| Familia | Numérico | Debe ser una Familia existente |

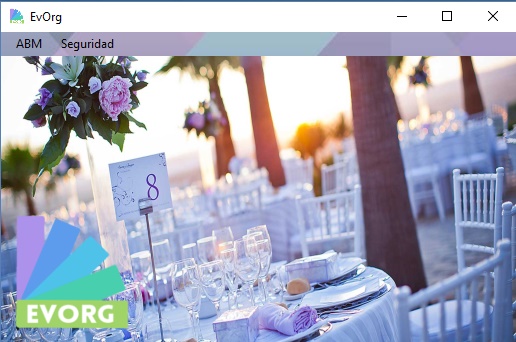
Pantallas Casos de Uso ABM

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 13/4/2017 | 1.0 | Creación de las pantallas |
| 24/4/2017 | 1.1 | Se agregan las pantallas de Usuario, Familia y Patente |
| 5/5/2017 | 1.2 | Se modifican las pantallas |
| 6/5/2017 | 1.2 | Se les agrega un icono a las pantallas |
| 4/6/2017 | 1.3 | Se actualizan las pantallas de paso, tipo de evento, salón, materiales y servicios |

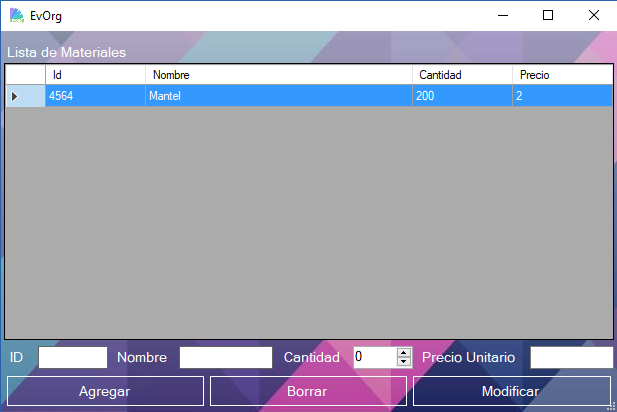
Log In



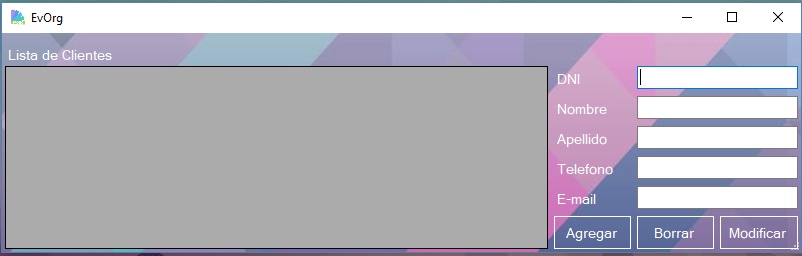
Pantalla de Inicio



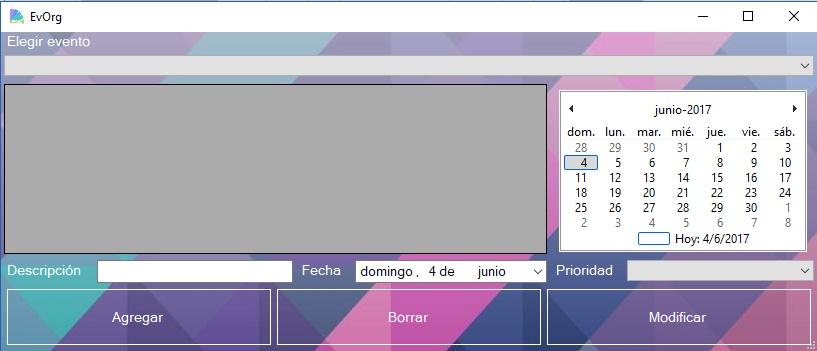
ABM de Materiales



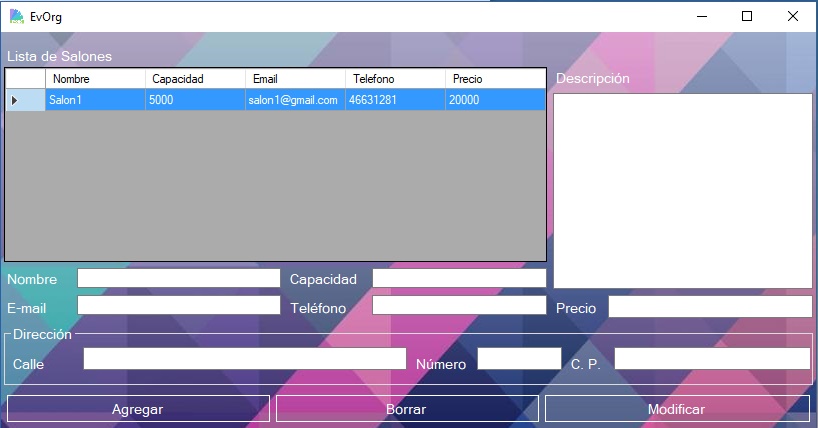
ABM de Cliente



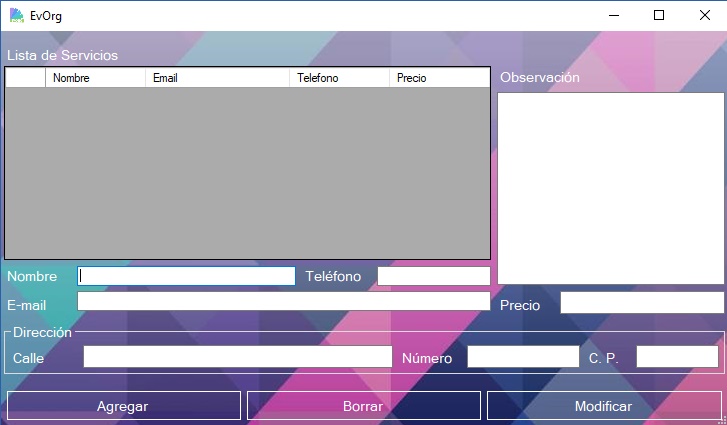
ABM de Paso



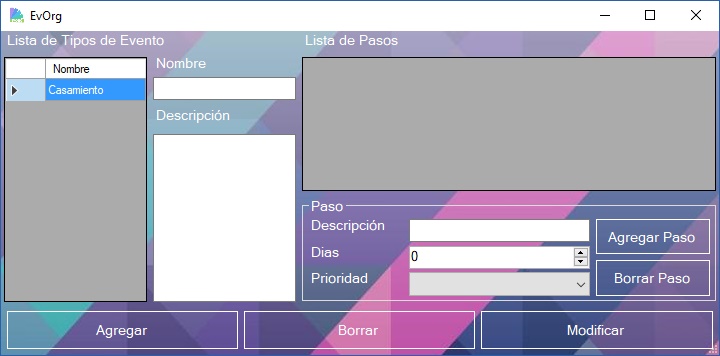
ABM de Salón



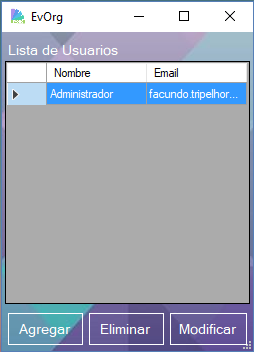
ABM de Servicio



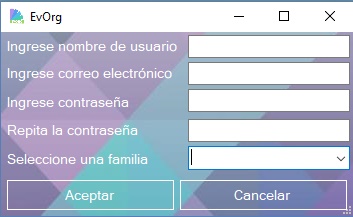
ABM de Tipo de Evento



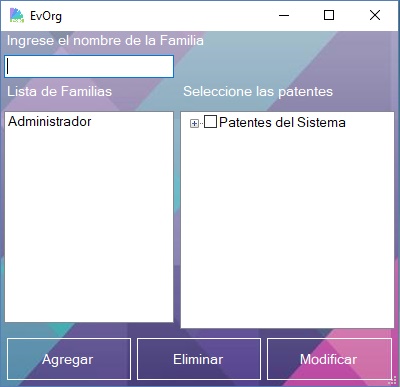
ABM de Usuario



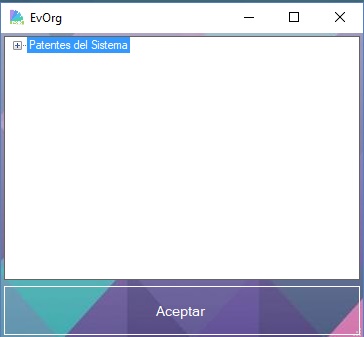
Nuevo Usuario



ABM de Familia



ABM de Patente



Nueva Patente

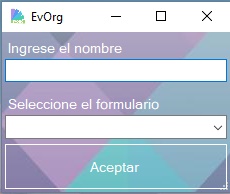


Diagrama de Clases de Entidades ABM

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 14/7/2017 | 1.5 | Creación de la sección Diagrama de Clases de Entidades ABM |



Diagrama Entidad-Relación de Entidades ABM

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 14/7/2017 | 1.5 | Creación de la sección Diagrama Entidad-Relación de Entidades ABM |



Caso de Uso 004 – Ingresar al sistema

Historial de Revisión

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 5/5/2017 | 1.2 | Creación del caso de uso |

Descripción Detallada

|  |  |
| --- | --- |
| **1 - Nombre:** | **CUSO 004 – Ingresar al sistema** |
| Objetivo: | El Usuario ingresa al sistema |
| Actores: | Usuario |
| **2 - Flujo de Eventos** | |
| Disparador/Condición: | El usuario inicia el sistema |
| Flujo Básico: | |
|  | 1. El sistema muestra la pantalla de log in y solicita al usuario que ingrese sus datos: 2. Datos a ingresar: Id de usuario y contraseña. 3. El usuario ingresa los datos requeridos. 4. El sistema realiza una validación de los campos verificando que cumplan el formato establecido. 5. El sistema encripta la contraseña 6. El sistema verifica la existencia del usuario, y que la contraseña sea correcta, en la base de datos 7. El sistema permite el ingreso mostrando la pantalla principal del sistema 8. El sistema asigna la familia correspondiente al usuario 9. El sistema carga el menú de forma dinámica según los permisos que posea el usuario. |
| Flujos Alternativos: | |
|  | Para el paso 3: Los datos ingresados no tienen un formato válido   * 1. El sistema deberá notificar que los datos no son válidos.   2. Se requerirá el ingreso de datos nuevamente.   3. Se vuelve al punto 2 del Escenario Principal.   Para el paso 5: Los datos ingresados no son válidos   1. El usuario ingresado no existe    1. El sistema deberá informar que el usuario ingresado no existe.    2. Se requerirá el ingreso de datos nuevamente    3. Se vuelve al punto 2 del Escenario Principal. 2. La contraseña ingresada no coincide con la almacenada en la base de datos 3. El sistema deberá informar que la contraseña no es correcta 4. Se requerirá el ingreso de datos nuevamente 5. Se vuelve al punto 2 del Escenario Principal |
| **3 - Requerimientos Especiales** | |
| **4 - Precondiciones:** | El usuario debe estar previamente registrado en el sistema |
| **5 - Postcondiciones:** | El usuario ingresa en el sistema |
| **6 - Puntos de Extensión** |  |
| **7 - Puntos de Inclusión.** | [CU005 – Asignar Familias y Patentes](#CU005)  [CU006 – Encriptar](#CU006) |

Diagrama de Caso de Uso



Diagrama de Secuencia



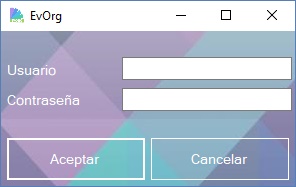
Diagrama de Clases



Diagrama Entidad-Relación



Pantalla



Caso de Uso 005 – Asignar Familias y Patentes

Historial de Revisión

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 5/5/2017 | 1.2 | Creación del caso de uso |

Descripción Detallada

|  |  |
| --- | --- |
| **1 - Nombre:** | **CUSO 005 – Asignar Familias y Patentes** |
| Objetivo: | El sistema busca la familia del usuario que ingresó al sistema y carga de forma dinámica las opciones a las que este tenga acceso. |
| Actores: | Usuario, Sistema |
| **2 - Flujo de Eventos** | |
| Disparador/Condición: | [CU004 – Ingresar al sistema](#CU004) |
| Flujo Básico: | |
|  | 1. El usuario se loguea en el sistema. 2. El sistema busca la familia del usuario. 3. El sistema lista las patentes de la familia correspondiente. 4. El sistema agrega al menú en tiempo de ejecución las patentes correspondientes. 5. El sistema muestra el menú en la pantalla principal. |
| Flujos Alternativos: | |
|  |  |
| **3 - Requerimientos Especiales** | |
| **4 - Precondiciones:** | El login debe haber sido existoso |
| **5 - Postcondiciones:** | El usuario visualiza las opciones del menú a las que tenga acceso |
| **6 - Puntos de Extensión** |  |
| **7 - Puntos de Inclusión.** |  |

Diagrama de Caso de Uso



Diagrama de Secuencia



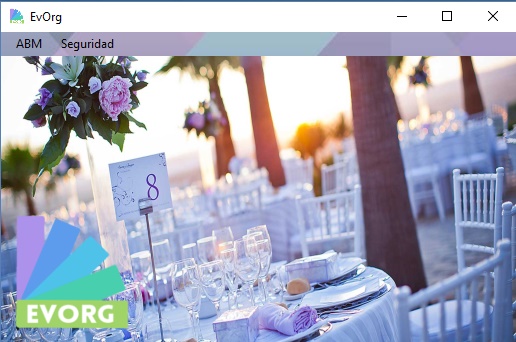
Diagrama de Clases



Diagrama Entidad-Relación



Pantalla



Patrón Composite

El patrón Composite sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol.

Mediante el composite se maneja el patrón de Usuario – Familia – Patente. A cada usuario al ser dado de alta, se le asigna una familia, de modo que al loguearse en el sistema, éste dibuje dinámicamente los controles en el menú a los que el usuario tiene el acceso permitido.

Está formado por las siguientes clases

* PatenteAbstracta: Clase abstracta que posee las propiedades:
  + - Id
    - Nombre
    - Padre
* PatenteAbstractaDinámica: Clase abstracta que posee los métodos:
  + - MostrarEnTreeView
    - MostrarEnMenuStrip
* Patente: Clase que hereda de patente abstracta y tiene la propiedad Formulario.
* PatenteDinámica: Clase que hereda de patente abstracta dinámica, pero decide no implementar de ninguna forma los métodos heredados.
* GrupoPatente: Clase que hereda de patente abstracta y tiene una lista de patentes abstractas, la cual va a recorrer con los métodos MostrarEnMenuStrip y MostrarEnTreeView.
* GrupoPatenteDinámico: Clase que hereda de patente abstracta dinámica e implementa los métodos MostrarEnMenuStrip y MostrarEnTreeView.

Código de Implementación del Patrón Composite

Capa BLL\_Estatica

Clase PatenteAbstracta

Public MustInherit Class PatenteAbstracta

Property Id As Integer

Property Nombre As String

Property Padre As Integer

End Class

Clase Patente

Public Class Patente

Inherits PatenteAbstracta

Property Formulario As String

Sub New()

End Sub

Sub New(pId As Integer, pNombre As String, pFormulario As String, Optional pPadre As Integer = 1)

Id = pId

Nombre = pNombre

Formulario = pFormulario

Padre = pPadre

End Sub

End Class

Clase GrupoPatente

Public Class GrupoPatente

Inherits PatenteAbstracta

Property ListaPatentes As New List(Of PatenteAbstracta)

Sub New()

End Sub

Sub New(pId As Integer, pNombre As String, Optional pPadre As Integer = 1)

Id = pId

Nombre = pNombre

Padre = pPadre

End Sub

End Class

Capa BLL\_Dinamica

Clase PatenteAbstractaDinamica

Public MustInherit Class PatenteAbstractaDinamica

Public MustOverride Function MostrarEnTreeView(pTreeView As Windows.Forms.TreeView) As Windows.Forms.TreeView

Public MustOverride Sub MostrarEnMenuStrip(pMenuStrip As Windows.Forms.MenuStrip, pUsuario As BLL\_Estatica.Usuario, pFormulario As Windows.Forms.Form)

End Class

Clase PatenteDinamica

Public Class PatenteDinamica

Inherits PatenteAbstractaDinamica

Implements IABMC

Dim PatenteDatos As New PatenteDatos

Public Sub Alta(Optional pObjeto As Object = Nothing) Implements IABMC.Alta

PatenteDatos.Alta(pObjeto)

End Sub

Public Sub Baja(Optional pObjeto As Object = Nothing) Implements IABMC.Baja

PatenteDatos.Baja(pObjeto)

End Sub

Public Sub Modificacion(Optional pObjeto As Object = Nothing) Implements IABMC.Modificacion

PatenteDatos.Modificacion(pObjeto)

End Sub

Public Overrides Sub MostrarEnMenuStrip(pMenuStrip As MenuStrip, pUsuario As Usuario, pFormulario As Form)

Throw New NotImplementedException()

End Sub

Public Function ConsultaTodo() As List(Of Object) Implements IABMC.ConsultaTodo

Return PatenteDatos.ConsultaTodo()

End Function

Public Overrides Function MostrarEnTreeView(pTreeView As TreeView) As TreeView

Throw New NotImplementedException()

End Function

Public Function CheckPatente(pNombre As String) As Boolean

Return PatenteDatos.CheckPatente(pNombre)

End Function

End Class

Clase GrupoPatenteDinamico

Public Class GrupoPatenteDinamico

Inherits PatenteAbstractaDinamica

Implements IABMC

Dim GrupoPatenteDatos As New GrupoPatenteDatos

Public Sub Alta(Optional pObjeto As Object = Nothing) Implements IABMC.Alta

GrupoPatenteDatos.Alta(pObjeto)

End Sub

Public Sub Baja(Optional pObjeto As Object = Nothing) Implements IABMC.Baja

GrupoPatenteDatos.Baja(pObjeto)

End Sub

Public Sub Modificacion(Optional pObjeto As Object = Nothing) Implements IABMC.Modificacion

GrupoPatenteDatos.Modificacion(pObjeto)

End Sub

Public Function ConsultaTodo() As List(Of Object) Implements IABMC.ConsultaTodo

Return GrupoPatenteDatos.ConsultaTodo()

End Function

Public Overrides Function MostrarEnTreeView(pTreeView As TreeView) As TreeView

Try

Dim vListaGrupoPatente As New List(Of PatenteAbstracta)

vListaGrupoPatente = GrupoPatenteDatos.Listar()

Dim vNode As TreeNode = pTreeView.Nodes.Add(vListaGrupoPatente.Item(0).Nombre)

vNode.Tag = vListaGrupoPatente.Item(0)

AgregarHijos(vNode.Tag, vNode)

Catch ex As Exception

End Try

Return pTreeView

End Function

Private Sub AgregarHijos(pPadre As GrupoPatente, pTreeNode As TreeNode)

For Each vPAbstracta As PatenteAbstracta In pPadre.ListaPatentes

Dim vNode As New TreeNode

If TypeOf vPAbstracta Is GrupoPatente Then

Dim vGPatente As GrupoPatente

vGPatente = DirectCast(vPAbstracta, GrupoPatente)

vNode.Text = vGPatente.Nombre

vNode.Tag = vGPatente

pTreeNode.Nodes.Add(vNode)

If vGPatente.ListaPatentes.Count > 0 Then

AgregarHijos(vGPatente, pTreeNode.LastNode)

End If

Else

Dim vPatente As Patente

vPatente = DirectCast(vPAbstracta, Patente)

vNode.Text = vPatente.Nombre

vNode.Tag = vPatente

pTreeNode.Nodes.Add(vNode)

End If

Next

End Sub

Public Overrides Sub MostrarEnMenuStrip(pMenuStrip As MenuStrip, pUsuario As Usuario, pFormulario As Form)

Dim vLista As New List(Of PatenteAbstracta)

Dim vUsuarioDinamico As New UsuarioDinamico

Dim vUsuario As Usuario = vUsuarioDinamico.DevuelveUsuario(pUsuario.IdUsuario)

vLista = vUsuario.Familia.GetPatentes

If vLista.Item(0).Nombre = "Patentes del Sistema" Then

Dim vPSistema As GrupoPatente = DirectCast(vLista.Item(0), GrupoPatente)

For Each vPAbstracta As PatenteAbstracta In vPSistema.ListaPatentes

Dim vTool As New ToolStripMenuItem

vTool.Name = vPAbstracta.Nombre

vTool.Text = vPAbstracta.Nombre

vTool.Tag = vPAbstracta

pMenuStrip.Items.Add(vTool)

pMenuStrip.Items.Item(vTool.Name).Text = vPAbstracta.Nombre

If TypeOf vPAbstracta Is GrupoPatente Then

AgregarToolStrip(vPAbstracta, vTool, pFormulario)

Else

AddHandler vTool.Click, AddressOf Menu\_Click

End If

Next

Else

For Each vPAbstracta As PatenteAbstracta In vLista

Dim vTool As New ToolStripMenuItem

vTool.Name = vPAbstracta.Nombre

vTool.Tag = vPAbstracta

pMenuStrip.Items.Add(vTool)

pMenuStrip.Items.Item(vTool.Name).Text = vPAbstracta.Nombre

If TypeOf vPAbstracta Is GrupoPatente Then

AgregarToolStrip(vPAbstracta, vTool, pFormulario)

End If

Next

End If

End Sub

Public Sub AgregarToolStrip(pPAbstracta As PatenteAbstracta, pTool As ToolStripMenuItem, pFormulario As Form)

Try

Dim vPadre As GrupoPatente = DirectCast(pPAbstracta, GrupoPatente)

If Not vPadre.ListaPatentes Is Nothing Then

For Each vPAbstracta As PatenteAbstracta In vPadre.ListaPatentes

Dim vTool As New ToolStripMenuItem

vTool.Name = vPAbstracta.Nombre

vTool.Text = vPAbstracta.Nombre

vTool.Tag = vPAbstracta

vTool.BackColor = System.Drawing.Color.Lavender

vTool.BackgroundImage = System.Drawing.Image.FromFile(Application.StartupPath + "\Fondo.jpg")

pTool.DropDownItems.Add(vTool)

pTool.DropDownItems.Item(vTool.Name).Text = vPAbstracta.Nombre

If TypeOf vTool.Tag Is Patente Then AddHandler vTool.Click, AddressOf Menu\_Click

If TypeOf vPAbstracta Is GrupoPatente Then

AgregarToolStrip(vPAbstracta, vTool, pFormulario)

End If

Next

End If

Catch ex As Exception

End Try

End Sub

Public Sub Menu\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Dim vSTool As ToolStripMenuItem = CType(sender, ToolStripMenuItem)

Click(vSTool)

End Sub

Private Sub Click(pTool As ToolStripItem)

Dim vFName As String = "OrganizacionDeEventos." & DirectCast(pTool.Tag, Patente).Formulario.ToString

Dim vAsm As Assembly = Assembly.GetEntryAssembly

Dim vTipo As Type = vAsm.GetType(vFName)

Dim vForm = Activator.CreateInstance(vTipo)

vForm.Show()

End Sub

Public Function CheckGrupoPatente(pNombre As String) As Boolean

Return GrupoPatenteDatos.CheckGrupoPatente(pNombre)

End Function

End Class

Diagrama de Clases del Patrón Composite



Diagrama Entidad-Relación del patrón Composite



Caso de Uso 006 – Encriptar

Historial de Revisión

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 5/5/2017 | 1.2 | Creación del caso de uso |

Descripción Detallada

|  |  |
| --- | --- |
| **1 - Nombre:** | **CUSO 006 – Encriptar** |
| Objetivo: | El sistema guarda información encriptada en la base de datos para que usuarios externos no puedan acceder a ella |
| Actores: | Administrador, Sistema |
| **2 - Flujo de Eventos** | |
| Disparador/Condición: | Alta de usuario, modificación de usuario |
| Flujo Básico: | |
|  | 1. El administrador ingresa los datos necesarios para la operación. 2. El sistema llama a encriptar la contraseña por medio del método MD5. 3. El sistema encripta la contraseña y la devuelve. 4. El sistema continúa con la operación. |
| Flujos Alternativos: | |
|  |  |
| **3 - Requerimientos Especiales** | |
| **4 - Precondiciones:** |  |
| **5 - Postcondiciones:** | El sistema guarda los datos encriptados en la base de datos |
| **6 - Puntos de Extensión** |  |
| **7 - Puntos de Inclusión.** |  |

Diagrama de Caso de Uso



Diagrama de Secuencia



Diagrama de Clases



Diagrama Entidad-Relación



Caso de Uso 007 – Cambiar Idioma

Historial de Revisión

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 7/7/2017 | 1.4 | Creación del caso de uso |

Descripción Detallada

|  |  |
| --- | --- |
| **1 - Nombre:** | **CUSO 007 – Cambiar Idioma** |
| Objetivo: | El usuario cambia el idioma seleccionado por otro disponible |
| Actores: | Usuario |
| **2 - Flujo de Eventos** | |
| Disparador/Condición: | El usuario desea cambiar de idioma |
| Flujo Básico: | |
|  | 1. El usuario hace click en el combobox de la pantalla de inicio 2. El sistema abre el combobox y muestra los idiomas disponibles 3. El usuario selecciona el idioma 4. El sistema notifica a las pantallas abiertas el cambio de idioma 5. Cada pantalla abierta envía lo que desea traducir al componente encargado de las traducciones 6. El sistema carga en memoria todas las traducciones precargadas para el idioma seleccionado por el usuario 7. El sistema traduce todas las leyendas a partir de lo cargado 8. El sistema muestra las leyendas traducidas |
| Flujos Alternativos: | |
|  |  |
| **3 - Requerimientos Especiales** | |
| **4 - Precondiciones:** | Debe haber más de un idioma cargado en la base de datos |
| **5 - Postcondiciones:** | El usuario visualiza el sistema en el idioma que haya seleccionado |
| **6 - Puntos de Extensión** |  |
| **7 - Puntos de Inclusión.** |  |

Diagrama de Caso de Uso



Diagrama de Secuencia



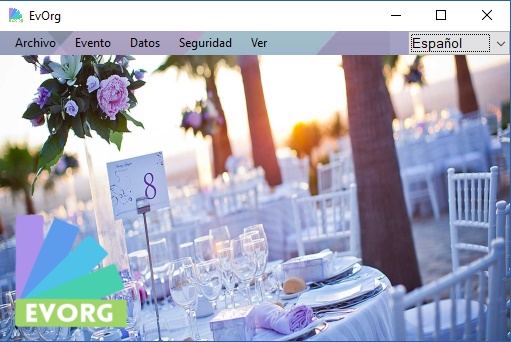
Diagrama de Clases

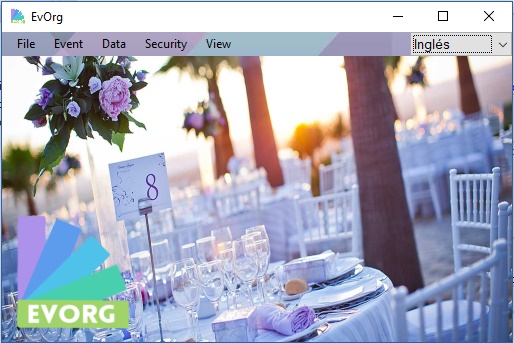


Diagrama Entidad-Relación



Pantalla





Patrón Observer

El patrón Observer es un patrón de comportamiento que permite relacionar diferentes objetos entre sí, en torno a uno principal, así cada vez que este último cambie su estado, todos sus objetos dependientes sean notificados y actualizados automáticamente.

Podemos decir entonces que dicho patrón se compone de un objeto observable u Observado y objetos observadores. Un objeto Observable puede tener uno o más Observadores.

En nuestro sistema se utiliza este patrón para realizar la actualización de cambio de idioma. Se implementan dos interfaces con los siguientes métodos y propiedades:

* IObservado

Propiedad:

* + Registrados: Listo (of IObservador)

Métodos:

* + Registrar(pObservador as IObservador)
  + Notificar()
* IObservador

Método:

* + ActualizarObservador(pControl as Control)

La clase “Traductor” es la que posee la propiedad del idioma que se está utilizando en el momento, por lo que es la que implementa la interfaz IObservado. Todos los formularios implementan la interfaz IObserver, registrándose, al ser abierto, en la lista Registrados de la clase “Traductor”, de manera que cuando el idioma se cambie, se notifique a todos los formularios en ejecución y se realice la traducción correspondiente.

Interfaz IObservador

Public Interface IObservador

Sub ActualizarObservador(Optional pControl As Control = Nothing)

End Interface

Interfaz IObservado

Public Interface IObservado

ReadOnly Property Registrados As List(Of IObservador)

Sub Registrar(pObservador As IObservador)

Sub Notificar()

End Interface

Clase Traductor

Public Class Traductor

Implements IObservado

#Region "Singleton"

Private Shared vTraductor As Traductor

Private Sub New()

End Sub

Public Shared Function GetInstance() As Traductor

If vTraductor Is Nothing Then

vTraductor = New Traductor

vTraductor.IdiomaSeleccionado = vTraductor.GetIdiomas().Item(0)

End If

Return vTraductor

End Function

Public Function Traducir(pTraducible As String) As String

If Me.IdiomaSeleccionado.Diccionario.ContainsKey(pTraducible) Then

Return Me.IdiomaSeleccionado.Diccionario(pTraducible)

Else

Return pTraducible

End If

End Function

#End Region

Private vIdiomaSeleccionado As Idioma

Public Property IdiomaSeleccionado() As Idioma

Get

Return vIdiomaSeleccionado

End Get

Set(ByVal value As Idioma)

If vIdiomaSeleccionado Is Nothing OrElse vIdiomaSeleccionado.Nombre <> value.Nombre Then

vIdiomaSeleccionado = value

Notificar()

End If

End Set

End Property

Dim vListaRegistrados As New List(Of IObservador)

Dim vIdiomaDinamico As IdiomaDinamico = IdiomaDinamico.GetInstance

Public ReadOnly Property Registrados As List(Of IObservador) Implements IObservado.Registrados

Get

Return vListaRegistrados

End Get

End Property

Public Sub Notificar() Implements IObservado.Notificar

For Each vRegistrado As IObservador In vListaRegistrados

vRegistrado.ActualizarObservador(vRegistrado)

Next

End Sub

Public Sub Registrar(pObservador As IObservador) Implements IObservado.Registrar

If vListaRegistrados.Count > 0 Then

For Each vRegistrado As IObservador In vListaRegistrados

If vRegistrado.ToString = pObservador.ToString Then

vListaRegistrados.Remove(vRegistrado)

Exit For

End If

Next

End If

vListaRegistrados.Add(pObservador)

End Sub

Public Function GetIdiomas() As List(Of Idioma)

Try

Dim vLista As New List(Of Idioma)

For Each vIdioma As Idioma In vIdiomaDinamico.ConsultaTodo

vLista.Add(vIdioma)

Next

Return vLista

Catch ex As Exception

Return Nothing

End Try

End Function

Public Sub AltaIdioma(pNombre As String, pDiccionario As Dictionary(Of String, String))

Dim vIdioma As New Idioma(pNombre, pDiccionario)

vIdiomaDinamico.Alta(vIdioma)

End Sub

Public Sub BajaIdioma(pIdioma As Idioma)

vIdiomaDinamico.Baja(pIdioma)

End Sub

Public Sub ModificacionIdioma(pIdioma As Idioma)

vIdiomaDinamico.Modificacion(pIdioma)

End Sub

End Class

Diagrama de Clases del Patrón Observer



Especificación de Aspectos Técnicos del Sistema

Historial de Revisión

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 14/7/2017 | 1.5 | Creación de la sección Especificación de Aspectos Técnicos del Sistema |

Gestión de perfiles de Usuario

Si en un sistema todos los usuarios tuvieran permiso de acceso a todos los componentes que hacen al mismo, sería un desastre. La curiosidad llamaría a usuarios poco especializados a adentrarse en secciones que no saben cómo funcionan y de esta manera generar problemas en el funcionamiento del sistema, como así también, inconsistencia en datos almacenados, pérdida de registros, robo de información, entre otros.

También, es necesario, que en una empresa se deleguen las tareas, de forma de que cada usuario del sistema esté encargado de llevar a cabo distintas funciones sin tener que entrometerse en las de otro.

Es por esto que es sumamente importante gestionar los permisos de los usuarios, permitiendo de esta forma que los usuarios puedan ver o no las pantallas correspondientes.

La gestión de perfiles de usuario en el sistema se realizará mediante el patrón “Usuario – Familia – Patente”.

Las patentes representan a los permisos para acceder a determinados formularios. Por ejemplo: permiso para ABM de usuarios, permiso para alta de idioma, permiso para alta de clientes, etc.

Una familia es un conjunto de patentes. Por ejemplo, la familia ABM puede tener asignadas todas las patentes para realizar ABM, como ABM usuarios, ABM clientes, etc.

De esta manera, al ser dado de alta, se le asigna a cada usuario una familia para que tenga permisos de acceso habilitados o denegados a los formularios. Además, el único que pueda dar de alta a usuarios va a ser aquel que posea permisos de Administrador.

En el sistema, se utilizó el patrón Composite para su implementación. Para ver más detalles de este patrón y su implementación en el sistema ver el anexo “[Patrón Composite](#Composite)”.

Gestión de Inicio de Sesión del Sistema

Permite verificar la identidad del usuario a través del ingreso de su nombre de usuario y su contraseña, asignándole el perfil que tenga asignado en el sistema. Previo a la validación de credenciales, se realiza una validación de campos. En esta se verifica que los campos requeridos para el ingreso no se encuentren vacíos y que además posean el formato adecuado.

Luego se procede a consultar la familia que el usuario tiene asignada y cargar las patentes correspondientes para mostrar de forma dinámica los formularios a los que el usuario que ingresó al sistema tiene permiso de acceso.

Gestión de Múltiples Idiomas

El sistema ofrecerá la posibilidad de mostrar su interfaz en dos idiomas, pudiendo visualizar todas las leyendas, etiquetas de los componentes y mensajes de error o alerta en el idioma elegido.

Se utilizarán los idiomas Inglés y Español por defecto, considerando que el idioma predeterminado será el Español.

El sistema contará con la opción de agregar nuevos idiomas permitiendo al usuario a cargo traducir manualmente todas las leyendas cargadas. Además, las leyendas traducidas a los idiomas predeterminados pueden ser editadas también.

En el sistema se utilizó el patrón Observer para su implementación. Para ver más detalles acerca del patrón y su implementación ver el anexo “[Patrón Observer](#Observer)”.

Gestión de Encriptado

La encriptación se puede definir como el proceso en que uno o varios archivos o datos, son protegidos mediante el uso de un algoritmo que desordena sus componentes, haciendo imposible que sean abiertos o decodificados si es que no se tiene la “llave” para desencriptarlos. Si una persona intercepta un archivo encriptado que no está dirigido a ella, por más que consiga abrirlo sólo verá un grupo de caracteres ilegibles, ya que se requiere de una “llave” especial para llevarlos al formato normal. Es una medida de seguridad utilizada para que al momento de almacenar o transmitir información sensible, ésta no pueda ser obtenida con facilidad por terceros.

La información que se almacena en la base de datos no es del todo segura, varios usuarios, con poco de conocimiento, pueden llegar a infiltrarse en la misma para robar información, como puede ser la contraseña de los usuarios.

Es por esto, que en el sistema se utilizará un método de encriptación denominado MD5, el cual es un algoritmo de reducción criptográfico de 128 bits. Este método se caracteriza por no permitir obtener la palabra original partiendo de la palabra encriptada.

La codificación del MD5 de 128 bits es representada típicamente como número de 32 símbolos hexadecimales.

Diagrama de Clases de Aspectos Técnicos del Sistema

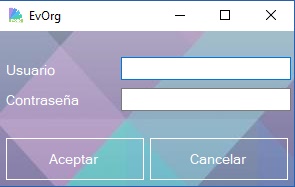


Diagrama Entidad-Relación de Aspectos Técnicos del Sistema

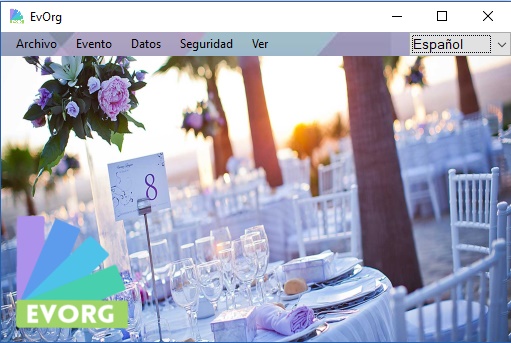


Pantallas del Sistema

Inicio de sesión



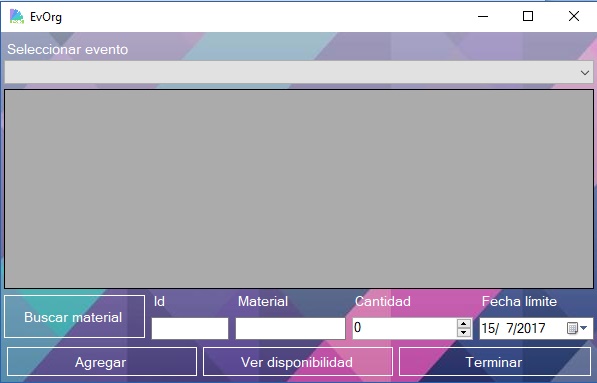
Pantalla principal



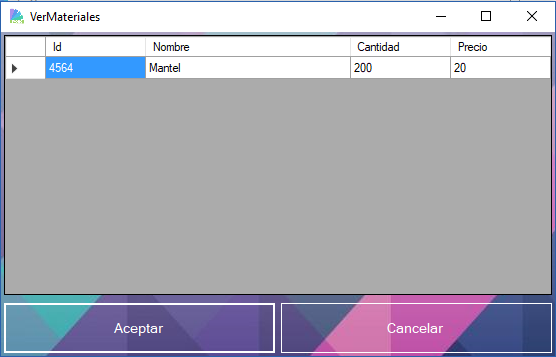
Organizar Evento



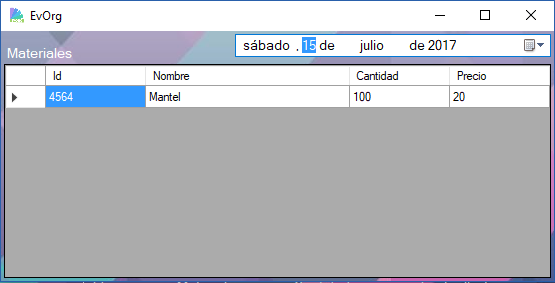
Reservar material



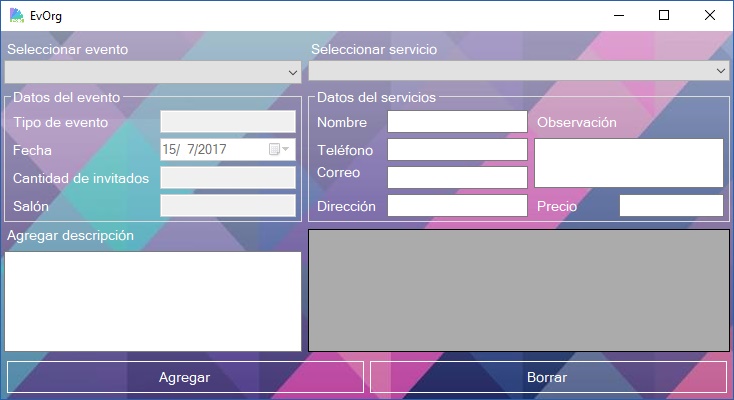
Ver materiales



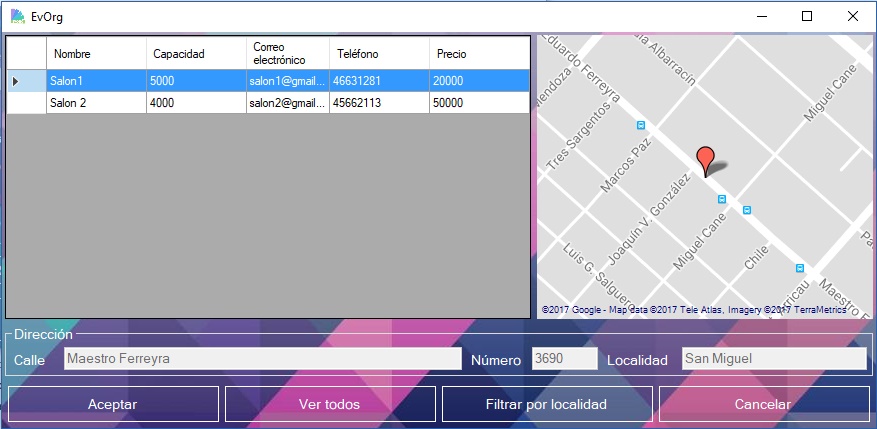
Disponibilidad de materiales



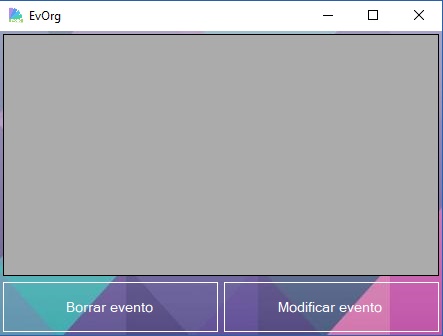
Reservar servicio



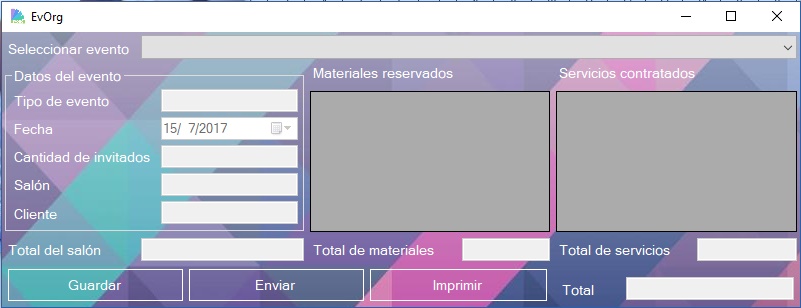
Seleccionar salón



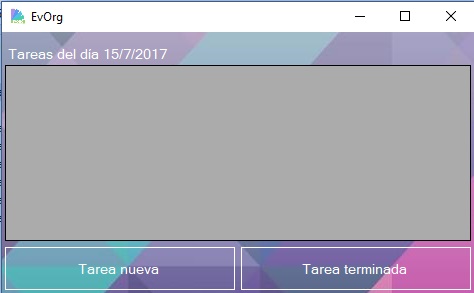
Ver eventos



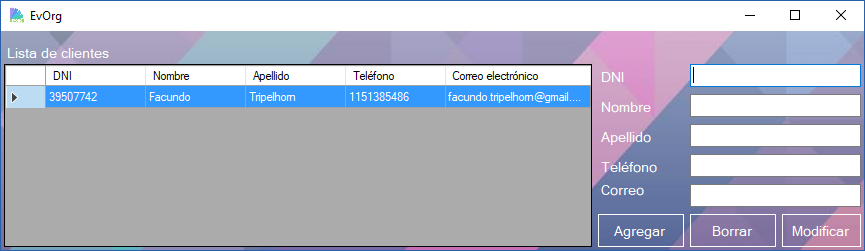
Generar presupuesto



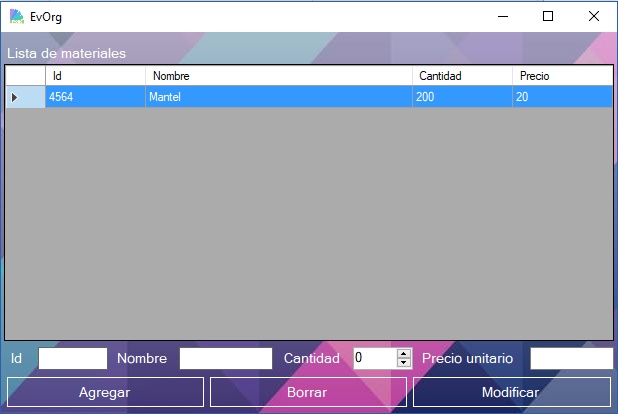
Ver tareas del día

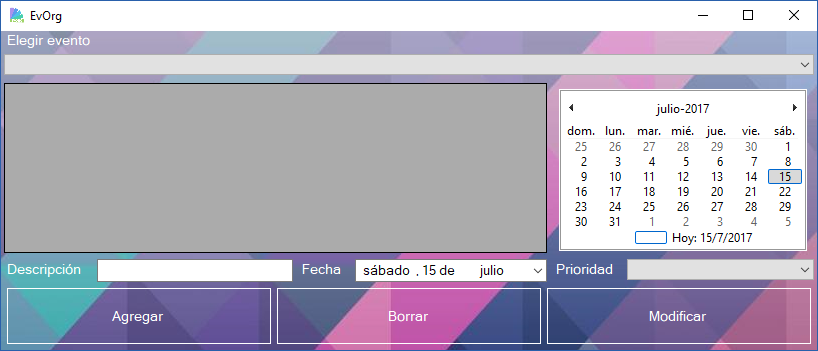


ABM Clientes

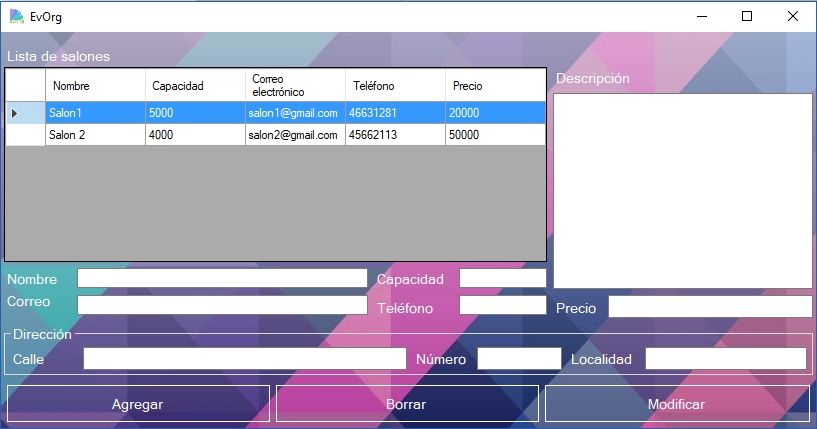


ABM Materiales

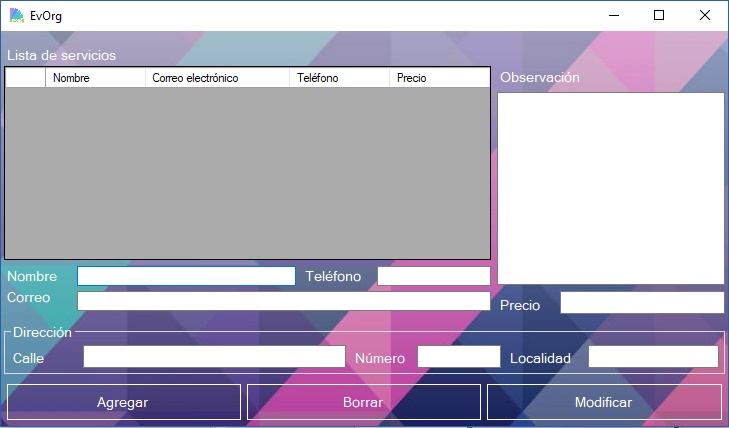


ABM Pasos

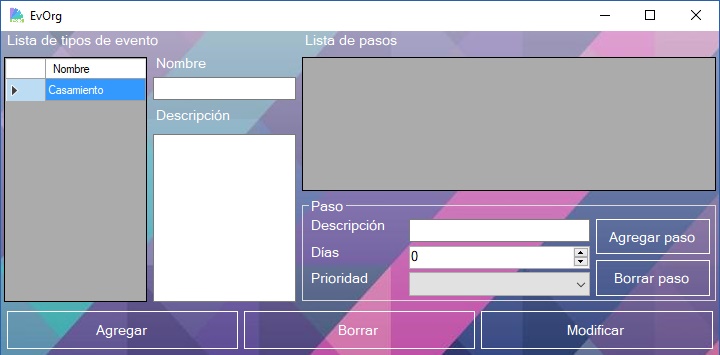
ABM Salones



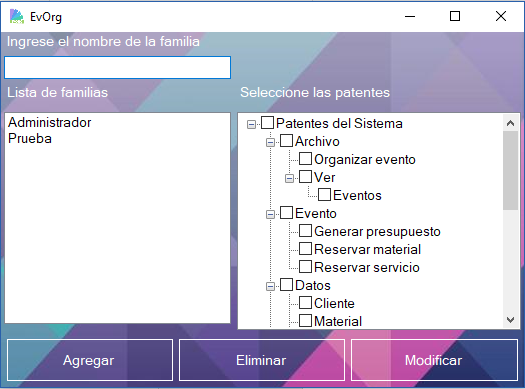
ABM Servicios



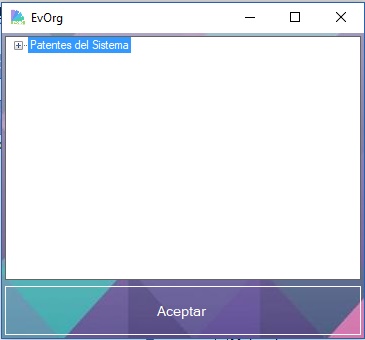
ABM Tipos de evento



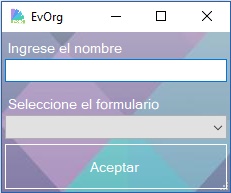
ABM Familias



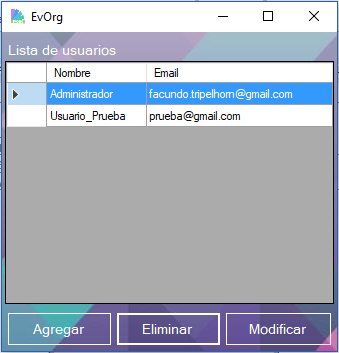
ABM Patentes



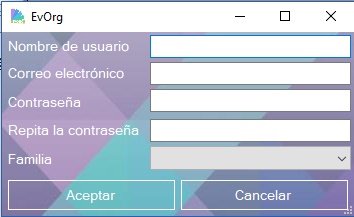
Nueva patente



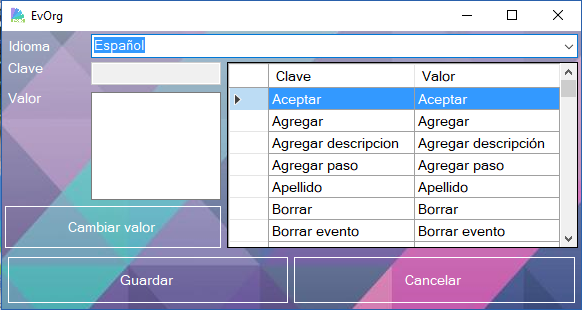
ABM Usuarios



Nuevo usuario



Gestionar idiomas



Anexo

Arquitectura en 6 capas

Para utilizar la arquitectura de las 6 capas utilizaremos un ejemplo de ABM de Usuario.

Capa Framework

Capa DAL\_ABMC

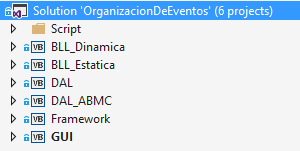
Capa GUI

Capa BLL\_Dinamica

Capa BLL\_Estatica

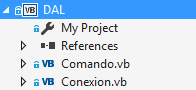
Capa DAL

BD



Capa DAL

Esta capa consta de dos clases: Conexión y Comando. Se encarga de realizar las conexiones a la base de datos y de realizar las consultas o sentencias necesarias.



* Clase Conexión

Public Class Conexion

Private Shared vObjConexion As SqlConnection

Public Shared Function ObjConexion() As SqlConnection

If vObjConexion Is Nothing Then

vObjConexion = New SqlConnection

End If

Dim vString As ConnectionStringSettings = ConfigurationManager.ConnectionStrings("MiConexion")

vObjConexion.ConnectionString = vString.ConnectionString

Return vObjConexion

End Function

End Class

Posee un método que devuelve un objeto de tipo SqlConnection configurado

* Clase Comando

Public Class Comando

Private Shared vObjComando As SqlCommand

Private Shared Function ObjComando(pSelectCommand As String, pConexion As SqlConnection) As SqlCommand

vObjComando = New SqlCommand

vObjComando.CommandText = pSelectCommand

vObjComando.CommandType = CommandType.Text

vObjComando.Connection = pConexion

Return vObjComando

End Function

Public Shared Function GetDataTable(pTablaNombre As String) As DataTable

Dim DAdapter As New SqlDataAdapter

Dim DTable As New DataTable

DAdapter.SelectCommand = Comando.ObjComando("SELECT \* FROM " & pTablaNombre, Conexion.ObjConexion())

DAdapter.Fill(DTable)

Return DTable

End Function

Public Shared Function GetData(pSelectCommand As String) As DataTable

Dim DAdapter As New SqlDataAdapter

Dim DTable As New DataTable

DAdapter.SelectCommand = Comando.ObjComando(pSelectCommand, Conexion.ObjConexion())

DAdapter.Fill(DTable)

Return DTable

End Function

Public Shared Sub ActualizarBD(pTablaNombre As String, pDTable As DataTable)

Dim DAdapter As New SqlDataAdapter

DAdapter.SelectCommand = Comando.ObjComando("SELECT \* FROM " & pTablaNombre, Conexion.ObjConexion())

Dim CBuilder As New SqlCommandBuilder(DAdapter)

DAdapter.InsertCommand = CBuilder.GetInsertCommand

DAdapter.UpdateCommand = CBuilder.GetUpdateCommand

DAdapter.DeleteCommand = CBuilder.GetDeleteCommand

DAdapter.Update(pDTable)

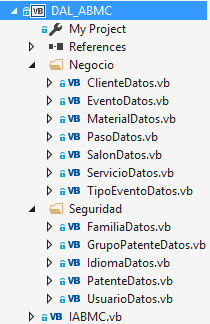
End Sub

End Class

Esta clase posee métodos que leer y escribir la base de datos.

Capa DAL\_ABMC

Esta capa consta de las clases mappers (mapeadores de objetos relacionales) y de una interfaz IABMC para realizar altas, bajas, modificación y consultas. En el ejemplo tenemos el mapper de Usuario.



* Clase UsuarioDatos

Public Class UsuarioDatos

Implements IABMC

Dim vUsuario As New Usuario

Public Sub Alta(Optional pObjeto As Object = Nothing) Implements IABMC.Alta

vUsuario = DirectCast(pObjeto, Usuario)

Try

Dim DTUsuario As DataTable = Comando.GetData("SELECT \* FROM Usuario WHERE Id = '" & vUsuario.IdUsuario & "'")

Dim DTFamilia As DataTable = Comando.GetData("SELECT \* FROM Familia WHERE Id = '" & vUsuario.Familia.Id & "'")

If DTUsuario.Rows.Count = 0 Then

Dim DRow As DataRow = DTUsuario.NewRow

DRow.ItemArray = {vUsuario.IdUsuario, vUsuario.Email, vUsuario.Contraseña, DTFamilia.Rows(0).Item(0)}

DTUsuario.Rows.Add(DRow)

Comando.ActualizarBD("Usuario", DTUsuario)

End If

Catch ex As Exception

End Try

End Sub

Public Sub Baja(Optional pObjeto As Object = Nothing) Implements IABMC.Baja

vUsuario = DirectCast(pObjeto, Usuario)

Try

Dim DTUsuario As DataTable = Comando.GetData("SELECT \* FROM Usuario WHERE Id = '" & vUsuario.IdUsuario & "' AND Password = '" & vUsuario.Contraseña & "'")

If DTUsuario.Rows.Count > 0 Then DTUsuario.Rows(0).Delete()

Comando.ActualizarBD("Usuario", DTUsuario)

Catch ex As Exception

End Try

End Sub

Public Sub Modificacion(Optional pObjeto As Object = Nothing) Implements IABMC.Modificacion

vUsuario = DirectCast(pObjeto, Usuario)

Try

Dim DTUsuario As DataTable = Comando.GetData("SELECT \* FROM Usuario WHERE Id = '" & vUsuario.IdUsuario & "' AND Password = '" & vUsuario.Contraseña & "'")

Dim DRow As DataRow = DTUsuario.NewRow

DRow.ItemArray = {vUsuario.IdUsuario, vUsuario.Email, vUsuario.Contraseña, vUsuario.Familia.Id}

If DTUsuario.Rows.Count > 0 Then DTUsuario.Rows(0).ItemArray = DRow.ItemArray

Comando.ActualizarBD("Usuario", DTUsuario)

Catch ex As Exception

End Try

End Sub

Public Function InicioSesion(pUsuario As Usuario, pContraseña As String) As Boolean

Dim vEncriptar As Encriptador = Encriptador.GetInstance

Try

Dim DT As DataTable = Comando.GetData("SELECT \* FROM Usuario WHERE Id = '" & pUsuario.IdUsuario & "' AND Password = '" & pUsuario.Contraseña & "'")

If vEncriptar.ValidarHash(pContraseña, DT.Rows(0).Item(2).ToString) Then

Return True

Else

Return False

End If

Catch ex As Exception

Return False

End Try

End Function

Public Function ConsultaTodo() As List(Of Object) Implements IABMC.ConsultaTodo

Dim vLista As New List(Of Object)

Try

Dim DT As DataTable = Comando.GetDataTable("Usuario")

If DT.Rows.Count > 0 Then

For Each DR As DataRow In DT.Rows

Dim DTF As DataTable = Comando.GetData("SELECT \* FROM Familia WHERE Id = " & DR.Item(3) & "")

vLista.Add(New Usuario(DR.Item(0), DR.Item(1), DR.Item(2), New Familia(DTF.Rows(0).Item(1))))

Next

End If

Catch ex As Exception

End Try

Return vLista

End Function

Public Function DevuelveUsuario(pIdUsuario As String) As Usuario

Try

Dim DTUsuario As DataTable = Comando.GetData("SELECT \* FROM Usuario WHERE Id = '" & pIdUsuario & "'")

If DTUsuario.Rows.Count > 0 Then

vUsuario.IdUsuario = pIdUsuario

vUsuario.Email = DTUsuario.Rows(0).Item(1)

vUsuario.Contraseña = DTUsuario.Rows(0).Item(2)

Dim DTFamilia As DataTable = Comando.GetData("SELECT \* FROM Familia WHERE Id = " & DTUsuario.Rows(0).Item(3))

Dim vFamilia As New Familia(DTFamilia.Rows(0).Item(0), DTFamilia.Rows(0).Item(1))

Dim vFamiliaDatos As New FamiliaDatos

Dim vLista As List(Of Object) = vFamiliaDatos.ListarPatentes(vFamilia)

For Each vPAbstracta As PatenteAbstracta In vLista

vFamilia.ListaPatentes.Add(vPAbstracta)

Next

vUsuario.Familia = vFamilia

Return vUsuario

Else

Return Nothing

End If

Catch ex As Exception

Return Nothing

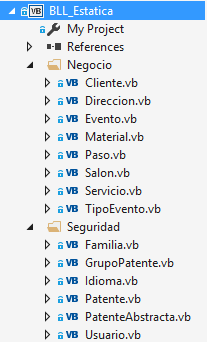
End Try

End Function

End Class

Capa BLL\_Estatica

Esta capa está formada por las entidades de negocio. Aquí se definen únicamente las estructuras de las entidades que maneja el sistema. En esta capa no suele haber métodos.



* Clase Usuario

Public Class Usuario

Property IdUsuario As String

Property Contraseña As String

Property Email As String

Property Familia As New Familia

Sub New()

End Sub

Sub New(pId As String)

IdUsuario = pId

End Sub

Sub New(pId As String, pContraseña As String)

Me.New(pId)

Contraseña = pContraseña

End Sub

Sub New(pId As String, pEmail As String, pContraseña As String)

Me.New(pId, pContraseña)

Email = pEmail

End Sub

Sub New(pId As String, pEmail As String, pContraseña As String, pFamilia As Familia)

Me.New(pId, pEmail, pContraseña)

Familia = pFamilia

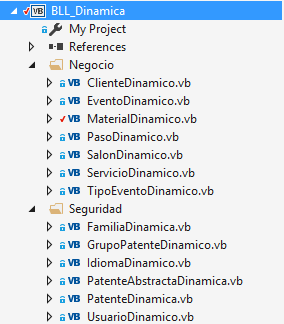
End Sub

End Class

Esta clase tiene las propiedades de la entidad Usuario y varios constructores para inicializar sus variables.

Capa BLL\_Dinamica

Esta capa está formada por la lógica de negocio. En esta capa se definen todos los comportamientos del sistema. Las clases suelen estar formadas únicamente por métodos.



* Clase Usuario\_Dinamico

Public Class UsuarioDinamico

Implements IABMC

Dim UsuarioDatos As New UsuarioDatos

Public Sub Alta(Optional pObjeto As Object = Nothing) Implements IABMC.Alta

UsuarioDatos.Alta(pObjeto)

End Sub

Public Sub Baja(Optional pObjeto As Object = Nothing) Implements IABMC.Baja

UsuarioDatos.Baja(pObjeto)

End Sub

Public Sub Modificacion(Optional pObjeto As Object = Nothing) Implements IABMC.Modificacion

UsuarioDatos.Modificacion(pObjeto)

End Sub

Public Function ConsultaTodo() As List(Of Object) Implements IABMC.ConsultaTodo

Return UsuarioDatos.ConsultaTodo()

End Function

Public Function InicioSesion(pUsuario As Usuario, pContraseña As String) As Boolean

Return UsuarioDatos.InicioSesion(pUsuario, pContraseña)

End Function

Public Function DevuelveUsuario(pIdUsuario As String) As Usuario

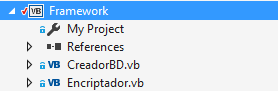
Return UsuarioDatos.DevuelveUsuario(pIdUsuario)

End Function

End Class

Capa Framework

Esta capa está formada por aspectos extra que no pertenecen a las demás capas. Posee dos clases “CreadorBD”, que es la que se utiliza al inicializar el sistema y se encarga de crear la base de datos en caso de que esta no exista en la máquina en la cual se desee ejecutar el sistema, y “Encriptador”, que es la clase que se encarga de encriptar la contraseña del usuario y de validar en caso de login que la contraseña ingresada sea correcta.



* Clase Encriptador

Public Class Encriptador

#Region "Singleton"

Private Shared vEncriptador As Encriptador = Nothing

Private Sub New()

End Sub

Public Shared Function GetInstance() As Encriptador

If vEncriptador Is Nothing Then

vEncriptador = New Encriptador

End If

Return vEncriptador

End Function

#End Region

Public Function Encriptar(pTexto As String) As String

Dim vMD5Hash As MD5 = MD5.Create

Dim vBytes As Byte() = vMD5Hash.ComputeHash(Encoding.UTF8.GetBytes(pTexto))

Dim vStringBuilder As New StringBuilder

For i = 0 To vBytes.Length - 1

vStringBuilder.Append(vBytes(i).ToString("X2"))

Next

Return vStringBuilder.ToString

End Function

Public Function ValidarHash(pTexto As String, pEncriptado As String) As Boolean

Dim vNuevoHash As String = Encriptar(pTexto)

If vNuevoHash = pEncriptado Then

Return True

Else

Return False

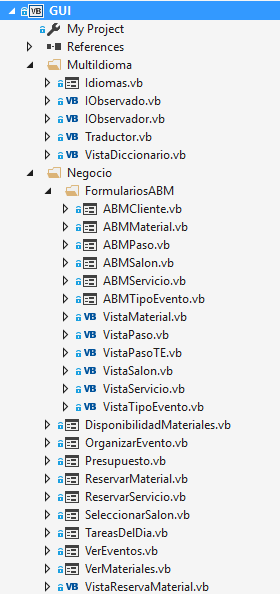
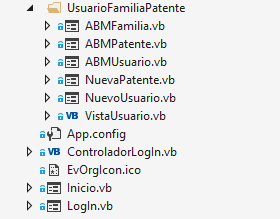
End If

End Function

End Class

Capa GUI

Esta capa es la capa de presentación, el ejecutable. Está formada por los formularios que verá el usuario, las clases requeridas para el multiidioma y clases “Vista” para que el usuario no pueda ver toda la información de las entidades, solo las necesarias. Es la encargada de controlar las entradas y salidas.



* Clase VistaUsuario

Public Class VistaUsuario

Property Nombre As String

Property Email As String

Sub New(pNombre As String, pEmail As String)

Nombre = pNombre

Email = pEmail

End Sub

End Class

Esta clase se usa para mostrar en pantalla solamente el nombre y el email del usuario, de esta forma nunca se visualiza la contraseña guardada en la base de datos.

* Formulario ABMUsuario

Public Class ABMUsuario

Implements IObservador

Dim vTraductor As Traductor = Traductor.GetInstance

Property Usuario As New Usuario

Property UsuarioDinamico As New UsuarioDinamico

Property FamiliaDinamica As New FamiliaDinamica

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

vTraductor.Registrar(Me)

Actualizar()

ActualizarObservador(Me)

End Sub

Public Sub Actualizar()

Try

Dim vLista As New List(Of VistaUsuario)

For Each vUsuario As Usuario In UsuarioDinamico.ConsultaTodo

vLista.Add(New VistaUsuario(vUsuario.IdUsuario, vUsuario.Email))

Next

GrillaUsuario.DataSource = Nothing

GrillaUsuario.DataSource = vLista

GrillaUsuario.Columns(1).AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill

ActualizarObservador(GrillaUsuario)

Catch ex As Exception

End Try

End Sub

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

Dim vNuevoUsuario As New NuevoUsuario

vNuevoUsuario.Modificacion = False

vNuevoUsuario.ShowDialog()

Actualizar()

End Sub

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

If GrillaUsuario.SelectedRows.Count > 0 Then

Usuario.IdUsuario = DirectCast(GrillaUsuario.SelectedRows(0).DataBoundItem, VistaUsuario).Nombre

For Each vUsuario As Usuario In UsuarioDinamico.ConsultaTodo

If vUsuario.IdUsuario = Usuario.IdUsuario Then Usuario = vUsuario

Next

UsuarioDinamico.Baja(Usuario)

Actualizar()

Else

MessageBox.Show(vTraductor.Traducir("Seleccione el usuario que desea borrar"), "EvOrg")

End If

End Sub

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

If GrillaUsuario.SelectedRows.Count > 0 Then

Usuario.IdUsuario = DirectCast(GrillaUsuario.SelectedRows(0).DataBoundItem, VistaUsuario).Nombre

For Each vUsuario As Usuario In UsuarioDinamico.ConsultaTodo

If vUsuario.IdUsuario = Usuario.IdUsuario Then Usuario = vUsuario

Next

For Each vFamilia As Familia In FamiliaDinamica.ConsultaTodo

If vFamilia.Nombre = Usuario.Familia.Nombre Then

Usuario.Familia = vFamilia

End If

Next

Dim vNuevoUsuario As New NuevoUsuario

vNuevoUsuario.Modificacion = True

vNuevoUsuario.Usuario = Usuario

vNuevoUsuario.ShowDialog()

Else

MessageBox.Show(vTraductor.Traducir("Seleccione el usuario que desea modificar"), "EvOrg")

End If

End Sub

Public Sub ActualizarObservador(Optional pControl As Control = Nothing) Implements IObservador.ActualizarObservador

For Each vControl As Control In pControl.Controls

Try

vControl.Text = vTraductor.IdiomaSeleccionado.Diccionario.Item(vControl.Tag.ToString)

Catch ex As Exception

Finally

If vControl.Controls.Count > 0 Then

ActualizarObservador(vControl)

End If

If TypeOf vControl Is DataGridView Then

For Each vColumna As DataGridViewColumn In DirectCast(vControl, DataGridView).Columns

Try

vColumna.HeaderText = vTraductor.IdiomaSeleccionado.Diccionario.Item(vColumna.Name)

Catch ex As Exception

End Try

Next

End If

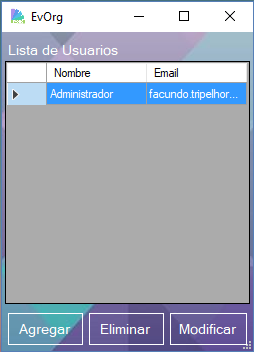
End Try

Next

End Sub

End Class

Este formulario muestra los usuarios guardados en la base de datos y te permite dar de alta un nuevo usuario y borrar o modificar un usuario existente. En caso de que se desee dar de alta o modificar un usuario se abrirá el formulario NuevoUsuario.



* Formulario NuevoUsuario

Public Class NuevoUsuario

Implements IObservador

Dim vTraductor As Traductor = Traductor.GetInstance

Property Familia As New Familia

Property FamiliaDinamica As New FamiliaDinamica

Property Usuario As New Usuario

Property UsuarioDinamico As New UsuarioDinamico

Property Modificacion As New Boolean

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

vTraductor.Registrar(Me)

Dim vLista As List(Of Object) = FamiliaDinamica.ConsultaTodo

For Each vFamilia As Familia In vLista

FamiliaCombo.Items.Add(vFamilia.Nombre)

Next

If Modificacion = True Then

IdUsuarioTxt.Text = Usuario.IdUsuario

IdUsuarioTxt.Enabled = False

EmailTxt.Text = Usuario.Email

End If

ActualizarObservador(Me)

End Sub

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

Try

Dim vEncriptar As Encriptador = Encriptador.GetInstance

If IdUsuarioTxt.Text <> "" And ContraseñaTxt.Text <> "" And EmailTxt.Text <> "" And RContraseñaTxt.Text <> "" Then

If ContraseñaTxt.Text = RContraseñaTxt.Text Then

If Regex.IsMatch(ContraseñaTxt.Text, "[a-zA-Z0-9]{6,16}") Then

If Regex.IsMatch(IdUsuarioTxt.Text, "[a-zA-Z0-9]{6,18}") Then

If Regex.IsMatch(EmailTxt.Text, "^[\w]+@{1}[\w]+\.[a-z]{2,3}$") Then

Usuario.IdUsuario = IdUsuarioTxt.Text

Usuario.Email = EmailTxt.Text

Usuario.Contraseña = vEncriptar.Encriptar(ContraseñaTxt.Text)

Dim vLista As List(Of Object) = FamiliaDinamica.ConsultaTodo

For Each vFamilia As Familia In vLista

If vFamilia.Nombre = FamiliaCombo.Text Then Familia = vFamilia

Next

Usuario.Familia = Familia

If Modificacion Then

UsuarioDinamico.Modificacion(Usuario)

Else

Dim vCheckUsuario As Usuario = UsuarioDinamico.DevuelveUsuario(Usuario.IdUsuario)

If vCheckUsuario Is Nothing Then

UsuarioDinamico.Alta(Usuario)

Else

Throw New Exception("El nombre de usuario ingresado ya existe")

ContraseñaTxt.Text = ""

RContraseñaTxt.Text = ""

End If

End If

Me.Close()

Else

Throw New Exception("El email ingresado es incorrecto")

ContraseñaTxt.Text = ""

RContraseñaTxt.Text = ""

End If

Else

Throw New Exception("El nombre de usuario tiene que tener al menos 6 caracteres y solo puede contener letras y números")

ContraseñaTxt.Text = ""

RContraseñaTxt.Text = ""

End If

Else

Throw New Exception("La contraseña tiene que tener al menos 6 caracteres y solo puede contener letras y números")

ContraseñaTxt.Text = ""

RContraseñaTxt.Text = ""

End If

Else

Throw New Exception("Las contraseñas no coinciden")

ContraseñaTxt.Text = ""

RContraseñaTxt.Text = ""

End If

Else

Throw New Exception("Ingrese los datos del usuario")

End If

Catch ex As Exception

MessageBox.Show(vtraductor.traducir(ex.Message), "EvOrg")

End Try

End Sub

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

Me.Close()

End Sub

Public Sub ActualizarObservador(Optional pControl As Control = Nothing) Implements IObservador.ActualizarObservador

For Each vControl As Control In pControl.Controls

Try

vControl.Text = vTraductor.IdiomaSeleccionado.Diccionario.Item(vControl.Tag.ToString)

Catch ex As Exception

Finally

If vControl.Controls.Count > 0 Then

ActualizarObservador(vControl)

End If

If TypeOf vControl Is DataGridView Then

For Each vColumna As DataGridViewColumn In DirectCast(vControl, DataGridView).Columns

Try

vColumna.HeaderText = vTraductor.IdiomaSeleccionado.Diccionario.Item(vColumna.Name)

Catch ex As Exception

End Try

Next

End If

End Try

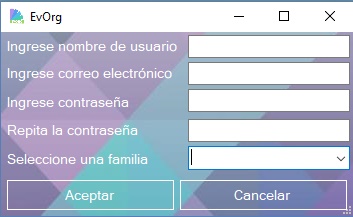
Next

End Sub

End Class

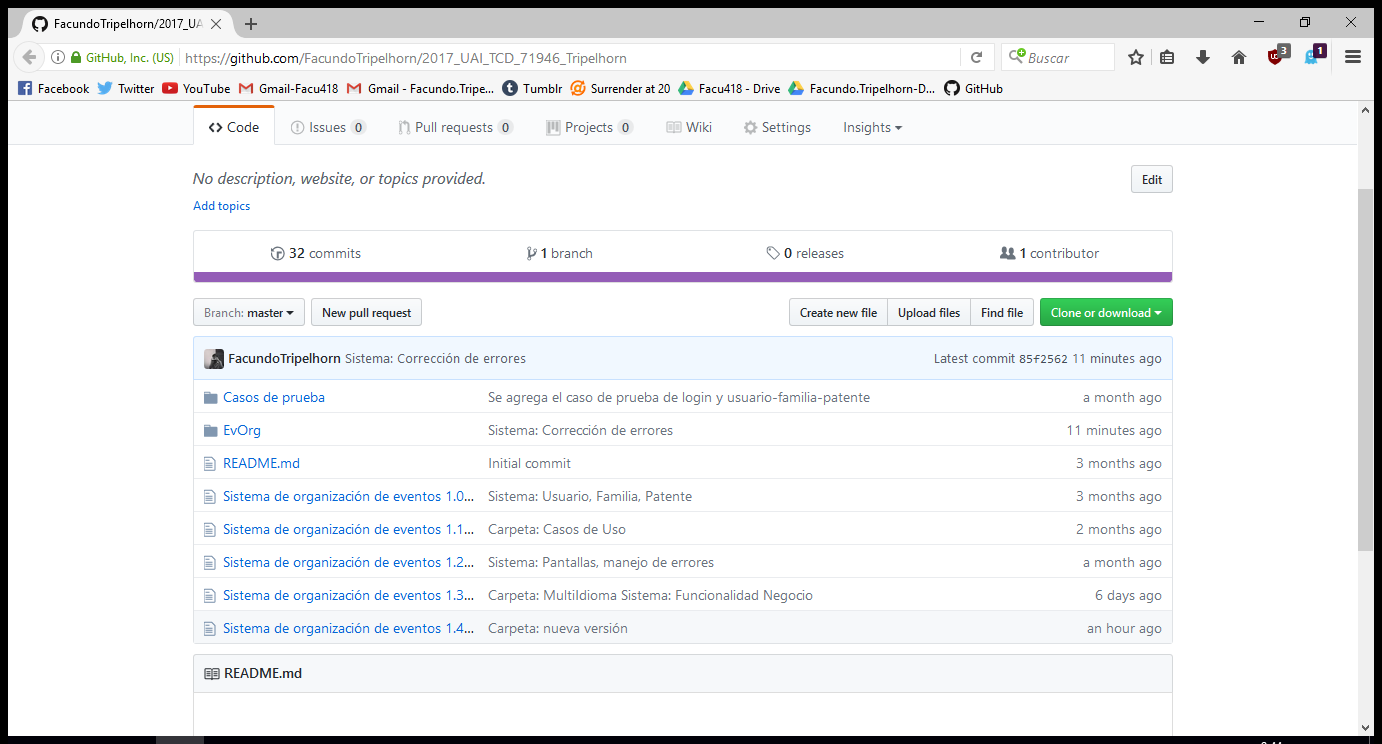
Este formulario en caso de que se haya abierto por que se desea dar de alta un usuario se abrirá con los campos vacío para que se ingrese la información del usuario, y al hacer “click” en el botón Aceptar se dará de alta el usuario nuevo.

En caso de que se haya abierto porque se desea modificar un usuario, el formulario se abrirá con los campos rellenados con la información del usuario a modificar y al hacer “click” en el botón Aceptar se modificará el usuario.

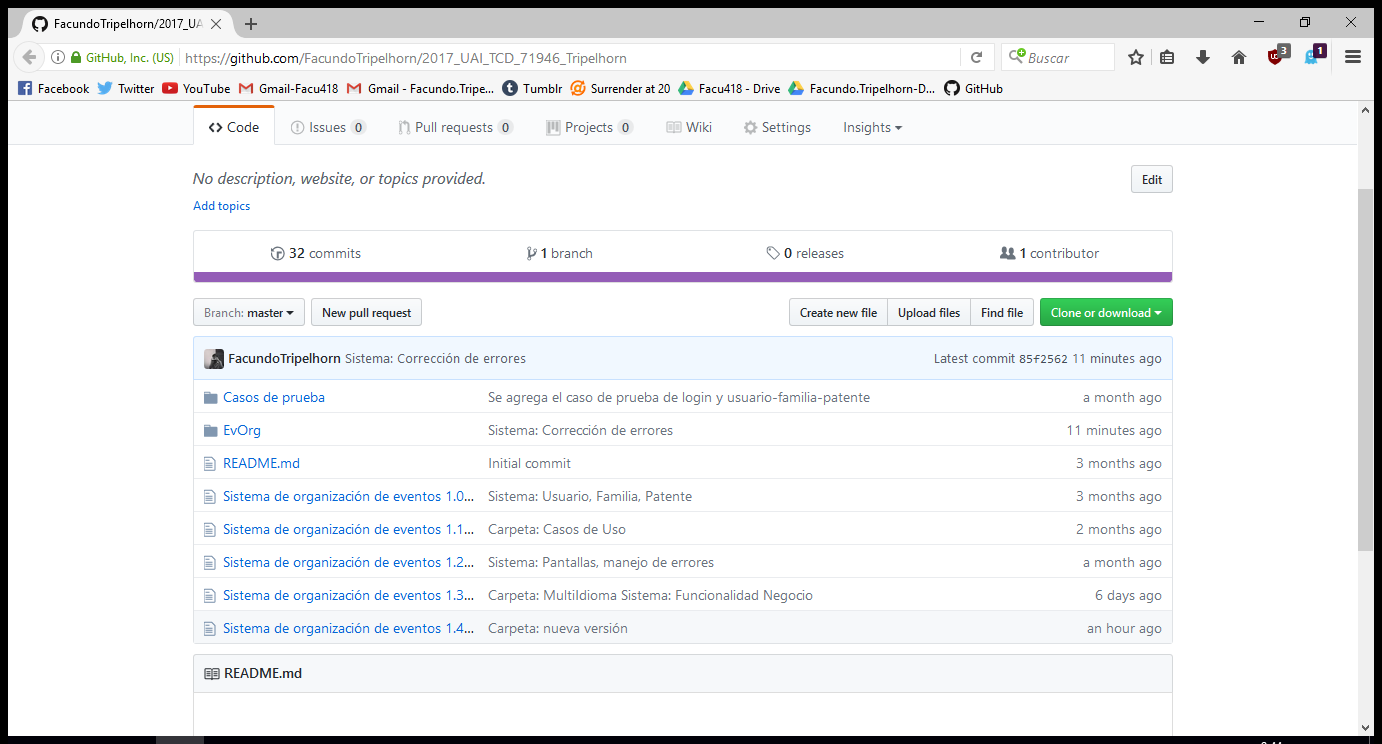


Cómo bajar el proyecto de GitHub

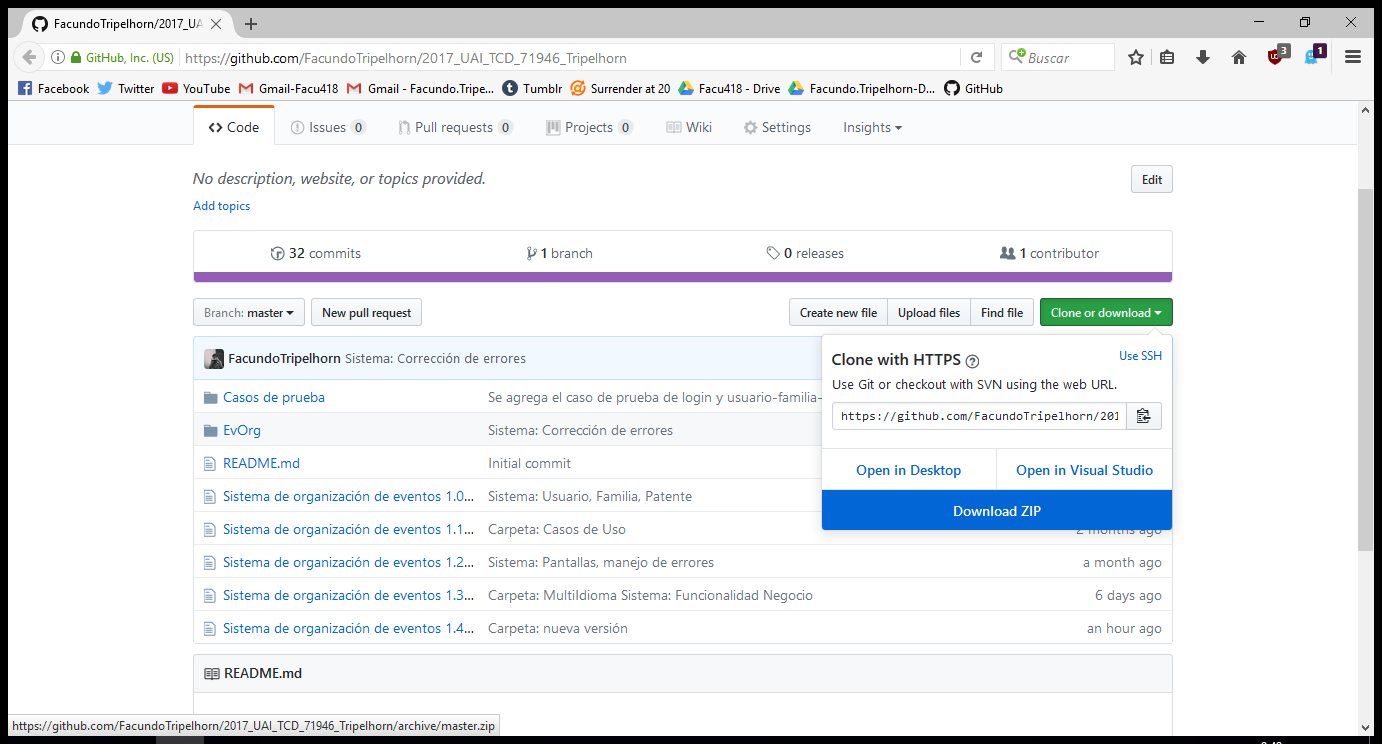
1. Ingresar al siguiente link <https://github.com/FacundoTripelhorn/2017_UAI_TCD_71946_Tripelhorn.git>
2. Se mostrará la siguiente página



1. Hacer “click” en “Clone or Download”



1. Luego hacer “click” en “Download ZIP”



1. Guardar el archivo

