**FACULTAD DE INGENIERÍA INDUSTRIAL**

**E.A.P. INGENIERÍA INDUSTRIAL**

**Análisis y diseño de un sistema de información para la gestión de reuniones de docentes universitarios**

Integrantes:

* Armas Benavides Raúl Marcelo 15170111
* Oliva Moya Ronnie Osmar 15170068

Docente:

Ing Ballón

Asignatura:

Ingeniería de Software

Ciudad Universitaria, Lima. de 2019

**Contenido**

[1 Resumen 6](#_Toc488188697)

[2 Introducción 6](#_Toc488188698)

[3 Aspectos de la investigación 6](#_Toc488188699)

[3.1 Descripción del problema 6](#_Toc488188700)

[3.2 Justificación del proyecto de investigación 6](#_Toc488188701)

[3.2.1 Razones sociales 6](#_Toc488188702)

[3.2.2 Razones económicas 7](#_Toc488188703)

[3.2.3 Razones técnicas 7](#_Toc488188704)

[3.2.4 Razones organizacionales 7](#_Toc488188705)

[3.3 Delimitación 7](#_Toc488188706)

[3.3.1 Espacial 7](#_Toc488188707)

[3.3.2 Cronología 7](#_Toc488188708)

[3.3.3 Conceptual 7](#_Toc488188709)

[3.3.4 Metodología 7](#_Toc488188710)

[3.3.5 Metodología de trabajo 8](#_Toc488188711)

[4 Objetivos 9](#_Toc488188712)

[4.1 Objetivos generales 9](#_Toc488188713)

[4.2 Objetivos específicos 9](#_Toc488188714)

[5 Marco Teórico 9](#_Toc488188715)

[5.1 Antecedentes 9](#_Toc488188716)

[5.1.1 Legales 9](#_Toc488188717)

[5.2 Calidad en educación 9](#_Toc488188718)

[5.3 Modelo de Calidad 10](#_Toc488188719)

[5.4 Estándar de Calidad 10](#_Toc488188720)

[5.5 Calidad en la educación universitaria 11](#_Toc488188721)

[5.6 Instituciones de acreditación en el Perú 11](#_Toc488188722)

[5.7 Estándar de calidad para la reunión periódicas de docentes 12](#_Toc488188723)

[5.8 Teorías genéricas basadas en la ingeniería 12](#_Toc488188724)

[6 Diseño metodológico 13](#_Toc488188725)

[6.1 Análisis del Sistema 13](#_Toc488188726)

[6.1.1 Metodología para el análisis 13](#_Toc488188727)

[6.2 Diseño Lógico 14](#_Toc488188728)

[6.2.1 Requerimientos 14](#_Toc488188729)

[6.2.2 Casos de uso 14](#_Toc488188730)

[7 Análisis del Sistema. Modelamiento de datos en UML ( Unified Modelling Language ) 16](#_Toc488188731)

[8 Implementación del sistema 23](#_Toc488188732)

[8.1 Creación de la base de datos en SQL Server 2008 23](#_Toc488188733)

[8.2 Creación de aplicación de escritorio de Visual Studio C# 2015 con infraestructura ADO.NET 24](#_Toc488188734)

[9 Análisis de resultados 24](#_Toc488188735)

[9.1 Codificación del programa 24](#_Toc488188736)

[9.2 Banco de pruebas 24](#_Toc488188737)

[9.2.1 Pruebas de función 24](#_Toc488188738)

[9.2.2 Pruebas modulares 25](#_Toc488188739)

[9.2.3 Pruebas de documentación y ayuda 26](#_Toc488188740)

[9.2.4 Pruebas de seguridad y control 26](#_Toc488188741)

[9.2.5 Pruebas de Calidad 27](#_Toc488188742)

[9.3 Informe de pruebas 27](#_Toc488188743)

[10 Conclusiones y Recomendaciones 27](#_Toc488188744)

[11 Bibliografía 28](#_Toc488188745)

[12 Anexos 29](#_Toc488188746)

# Resumen

.

# Introducción

# Aspectos de la investigación

## Descripción del problema

## Justificación del proyecto de investigación

### Razones sociales

Entre las razones sociales que existen para la elaboración de este proyecto, se encuentran la importancia de gestionar las reuniones de los docentes para que discuten sobre temas de enseñanza. De esta manera, se incrementaría el compromiso por parte de los docentes en su labor crucial en la educación superior en el país.

### Razones económicas

Se busca crear el software con el fin de que las facultades que lo soliciten puedan adquirirlo a bajos costos, ya que fue realizado por alumnos de manera ad honorem y se utilizó software libre para su creación.

### Razones técnicas

La aplicación podrá ser utilizada en equipos con plataforma Windows, en el lenguaje Visual C# y con bases de datos Microsoft SQL Server 2014 Express Edition, ya que estas herramientas son de la misma casa de software, son compatibles entre ellas y son populares a nivel computacional en la actualidad.

### Razones organizacionales

Las facultades que cuenten con un software como este, tendrán la posibilidad, de agilizar los procesos, de tener una estricta organización de sus docentes. Así mismo, tendrán la facilidad de adquirirlo, pues los bajos costos del software otorgan una ventaja competitiva.

## Delimitación

### Espacial

El proyecto se realiza en las instalaciones de la Facultad de Ingeniería Industrial de la UNMSM, bajo la coordinación con el ingeniero Wiler Ponce docente del curso de Base de Datos y Programación Visual, en donde se entregarán actas semanales para dicho curso.

### Cronología

El proyecto se realiza en un lapso de tiempo de 4 meses. Teniendo presente la presentación de avances periódicos al docente.

### Conceptual

En el proyecto se realiza la investigación sobre las reuniones de docentes en la universidad peruana, así mismo se realiza el manejo de la base de datos Microsoft SQL Server 2014 Express Edition, ya que es una herramienta nueva para el equipo de trabajo.

### Metodología

A nivel metodológico se trabajó el modelo de Análisis y Diseño orientado a Objetos debido a la orientación del proyecto y también por el lenguaje en el cual se realizará, en todo el CVDS (Ciclo de Vida de Desarrollo de Software), dado que esta metodología tiene un conjunto de disciplinas que desarrollan y modelan el software y facilitan la construcción de sistemas complejos a partir de componentes.

Para la construcción y el modelado de todos los componentes del proyecto a nivel de software se utilizó la herramienta UML, las siglas corresponden a Lenguaje Unificado de Modelado (Unified Modeling Language), la definición de UML se puede obtener descomponiendo las palabras que la conforman.

Lenguaje: como todo lenguaje cuenta con una sintaxis y una semántica, por lo tanto, al realizar cualquier modelo es necesario conocer una serie de reglas de cómo es el manejo del elemento s en el lenguaje

Modelado: con él se modelan los aspectos del mundo real, los cuales permiten la correcta interpretación y entendimiento de este.

Unificado: es unificado ya que agrupa varias técnicas de modelado en un solo modelo.

El proyecto se desarrolló a nivel de programación siguiendo un patrón de diseño, modelo vista controlador, el cual es bastante acorde con el modelo de Análisis y diseño orientado a objetos.

**Patrón Modelo-Vista-Controlador**

Este patrón se utiliza en diseño de aplicaciones con sofisticadas interfaces. El fin es realizar un diseño que separe la vista del modelo, esto para aprovechar la reusabilidad de código, de tal forma que los cambios en las vistas no afecten el modelo del negocio y el modelo de datos, por otra parte, al utilizar la herramienta .Net para el proyecto, en futuro se podrá llevar a la web por medio de ASP.Net, utilizando la misma controladora y el mismo modelo, para entender mejor cada uno de los componentes del modelo, se describen a continuación.

* El modelo es el responsable de:

Acceder a la capa de almacenamiento de datos. Es ideal que el modelo sea independiente del sistema de almacenamiento. Define las reglas de negocio.

* El controlador es responsable de:

Recibe los eventos de entrada Contiene reglas de gestión de eventos, tanto de la vista como del modelo.

* Las vistas son responsables de:

Recibir datos del modelo y mostrarlos al usuario, así como hacer el proceso contrario recibir los datos del usuario y registrarlos en el modelo para que sean procesados y registrados.

### Metodología de trabajo

Se seguirá la metodología ágil de software SCRUM

El tamaño óptimo del Equipo de Desarrollo es lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para completar una cantidad de trabajo significativa. Tener menos de tres miembros en el Equipo de Desarrollo reduce la interacción y resulta en ganancias de productividad más pequeñas. Los Equipos de Desarrollo más pequeños podrían encontrar limitaciones en cuanto a las habilidades necesarias durante un Sprint, haciendo que el Equipo de Desarrollo no pudiese entregar un Incremento que potencialmente se pueda poner en producción. Tener más de nueve miembros en el equipo requiere demasiada coordinación. Los Equipos de Desarrollo grandes generan demasiada complejidad como para que pueda gestionarse mediante un proceso empírico.

En el caso particular de este trabajo, el trabajo fue realizado por dos estudiantes.

# Objetivos

## Objetivos generales

* Desarrollar un sistema que gestione las reuniones de docentes universitarios con el fin de promover su participación en la discusión sobre temas de enseñanza.

## Objetivos específicos

* Modelar la gestión de las reuniones.
* Implementar una base de datos que sirva como soporte del sistema.
* Implementar una API (Interfaz de programación de aplicaciones, por sus siglas en inglés) para el sistema.

# Marco Teórico

## Antecedentes

## Teorías genéricas basadas en la ingeniería

Para el desarrollo del proyecto se utilizó un motor de base de Microsoft SQL Server 2014 Express Edition, versión libre en internet, donde se implementaron la base de datos para el almacenamiento de la información, relacionada con la gestión de reuniones de los docentes.

La aplicación se realizó para plataformas Microsoft. El lenguaje de programación que se utilizó para el desarrollo de la aplicación es Visual C#.

**Microsoft SQL Server 2014 Express Edition**

Está diseñado para brindar una simplicidad de uso, permitiendo instalaciones rápidas. La facilidad de uso comienza con una instalación robusta de la interface del usuario grafica (GUI) que guía al usuario a través del proceso de instalación. Las herramientas GUI que se encuentran incluidas sin cargo con SQL Server Express, incluyen Express Manager (versión Alfa) y Computer Manager. Estas herramientas simplifican las operaciones de bases de datos. El diseño y desarrollo de aplicaciones de bases de datos se vuelve más sencillo con la integración de Visual Studio. Con todas estas ventajas de interface de usuario, en el transcurso del proyecto se presentó la necesidad de ejecutar sentencias en SQL, por complejidad en la realización de procesos.

**Visual Studio .NET**

Visual Studio .NET es un conjunto completo de herramientas de desarrollo para la construcción de aplicaciones Web ASP, servicios Web XML, aplicaciones para escritorio y aplicaciones móviles. Visual Basic .NET, Visual C++ .NET, Visual C# .NET y Visual J# .NET utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y servicios Web XML.

# Diseño metodológico

El presente proyecto de investigación es de tipo descriptivo, ya que en la investigación se busca describir y especificar las propiedades más importantes de los procesos computacionales para lograr requerimientos óptimos y acordes con las necesidades de la gestión de reuniones de los en la universidad.

Adicionalmente en la investigación, se describirán las características que identifican los diferentes elementos y componentes de la investigación.

## Análisis del Sistema

### Metodología para el análisis

#### Análisis orientado a objetos

#### El presente proyecto de investigación se trabaja con metodología AOO, en donde se brindan los medios para mejorar la reutilización de los componentes software. El objetivo de la programación orientada a objetos es permitir que los sistemas informáticos sean fácilmente extendidos para mejorar su funcionalidad, o reutilizados en otros sistemas que requieran sus servicios.

Las características que integran la Programación orientada a objetos son la: abstracción, encapsulación, polimorfismo y herencia.

La herramienta de diseño será UML, Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software.

Se utilizan los siguientes diagramas para su conceptualización:

• Diagrama de Casos de Uso: mostrando los procesos relevantes en un escenario con sus respectivos actores.

• Diagrama de Secuencia: Muestran a los diferentes objetos y las relaciones que pueden tener entre ellos, los mensajes que se envían entre ellos.

• Diagrama de Actividades: Es un caso especial del diagrama de estados. Muestra el flujo entre los objetos. Se utilizan para modelar el funcionamiento del sistema y el flujo de control entre objetos.

• Diagramas de Clases: muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Los diagramas de clases son diagramas estáticos porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ella.

## Diseño Lógico

### Requerimientos

El sistema establecerá un mecanismo de gestión para reuniones de docentes. De modo que es necesario identificar cuáles son requerimientos. En ese sentido identificamos los requisitos funcionales.

**Requerimientos funcionales**: Definen qué debe hacer el sistema.[[1]](#footnote-2)

1. Los Jefes de Departamento, programarán reuniones periódicas, donde se discutan con los docentes temas relacionados con actividades de enseñanza, aprendizaje y sus resultados.

2. Registrar los acuerdos consensuados en actas para su seguimiento y cumplimiento.

3. Establecer los mecanismos para verificar el cumplimiento de los acuerdos.

### Casos de uso

ECU 01 – Logueo del sistema

ECU02– Registrar Cliente

ECU03– Registrar Venta

ECU04– Realizar Cotización

ECU05– Registrar Productos

|  |  |
| --- | --- |
| Actor |  |
| Rol |  |

|  |  |
| --- | --- |
| Actor |  |
| Rol |  |

|  |  |
| --- | --- |
| Actor |  |
| Rol |  |

|  |  |
| --- | --- |
| Actor |  |
| Rol |  |

**MODELAMIENTO DE LA GESTIÓN DE REUNIONES DE DOCENTES**

# Análisis del Sistema. Modelamiento de datos en UML ( Unified Modelling Language )

**Diagrama 1.Diagrama de casos de uso**[[2]](#footnote-3)



**Diagrama 2.Diagrama de secuencia de Asignación de tareas**



**Diagrama 3.Diagrama de secuencia de registro de Asistencia**



**Diagrama 4.Diagrama de secuencia de invitación a reuniones**



**Diagrama 5.Diagrama de secuencia de Redacción de documentación de actas**



**Diagrama 6. Diagrama de envío de actas de reuniones al director de escuela**



**Diagrama7. Diagrama de secuencia de Revisión de actas de reuniones por el directo**

# Implementación del sistema

## Creación de la base de datos en SQL Server 2008

Posteriormente, se creó la base de datos en el motor SQL Server 2008.Se crearon tablas[[3]](#footnote-4). El diagrama de relaciones en SQL Server Management Studio (Figura 1) detalla los campos de cada tabla.

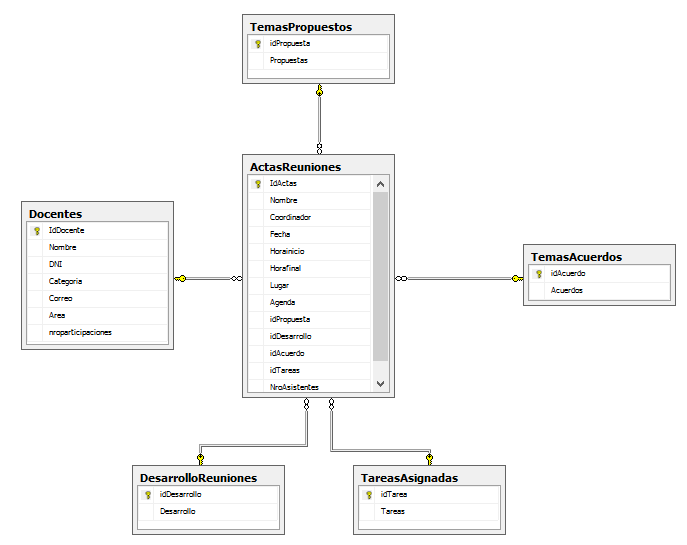


Figura 2. Diagrama de relaciones en SQL Server Management Studio (Elaboración propia)

## Creación de aplicación de escritorio de Visual Studio C# 2015 con infraestructura ADO.NET

Finalmente, se desarrolló la aplicación de escritorio en Visual Studio 2015 en el lenguaje de programación C#. Se programó mediante el patrón de software MVC(Modelo-Vista-Controlador). Esto permitió programar con mayor facilidad.

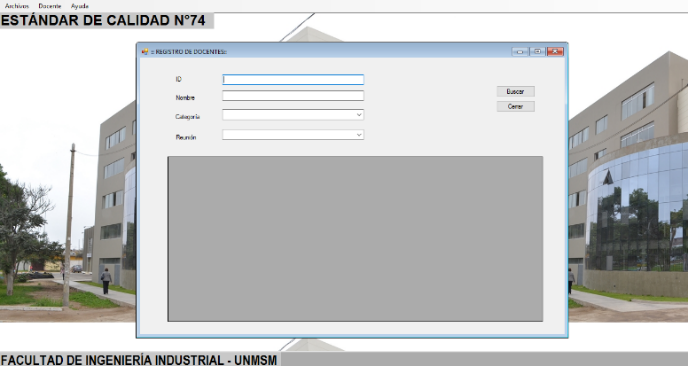


Figura 3. Aplicación de escritorio en Visual Studio C# (Elaboración propia)

# Análisis de resultados

## Codificación del programa

Es el proceso de trascripción del diseño funcional realizado en UML (Diagramas de Casos de Usos, Diagramas de Secuencia) a Visual C#.NET, para la creación de las interfaces, diseño y pruebas del monitoreo del aplicativo.

## Banco de pruebas

Las pruebas realizadas al desarrollo al finalizar fueron pruebas de caja negra debido a la carencia de un agente de pruebas de funcionamiento del desarrollo y una entidad de implementación del desarrollo, donde se harían pruebas de integración.

### Pruebas de función

Objetivo: Verificar que cada una de las funciones cumpla con el propósito para el cual fueron diseñados.

Criterio de decisión: De acuerdo a los requerimientos internacionales que exige el modelo del CVDS, dentro de su primera prueba de funcionalidad, exige la verificación de la validación de los caracteres de digitación, entre ellos están las letras, los números y los espacios en blanco en cada uno de los campos que se consideran obligatorios en el diseño de los respectivos formularios.

|  |  |  |
| --- | --- | --- |
| PRUEBAS | SÍ | NO |
| La función de validación de letras es correcta | X |  |
| La función de validación de números es correcta | X |  |
| La rutina de validación de espacios en blanco es correcta | X |  |

### Pruebas modulares

Objetivo: Detección de errores y verificación de la integración de los menús.

Criterio de decisión: Las pruebas modulares corresponden al segundo ítem mencionado en la aplicación de pruebas de la metodología del CVDS, y su función consiste en evaluar cada una de las rutinas que se integran al todo de la aplicación y que exista interacción adecuada con el usuario.

**MENÚ APLICACIÓN**

|  |  |  |
| --- | --- | --- |
| PRUEBAS | SÍ | NO |
| Se generan errores al momento de cargar un Submenú |  | X |
| Los Submenús son complejos |  | X |
| La interacción con el usuario es fácil | X |  |
| La selección de menús y submenús funciona | X |  |
| Realiza el proceso de inserción de nuevo entorno | X |  |
| Realiza el proceso de salir del entorno actual | X |  |

El siguiente pantallazo muestra la funcionalidad de los menús y los submenús los cuales son de fácil acceso al usuario y divididos según la funcionalidad a solicitar, anexo a los submenús están un cuadro de botones que tienen un texto de ayuda para saber a qué funcionalidad se está accediendo al hacer clic en el botón.

### Pruebas de documentación y ayuda

Objetivo: Verificar el contenido de la documentación, manuales y ayudas.

Criterio de decisión: Para las pruebas de documentación, igualmente se tomó la referencia de la metodología del CVDS, en donde el código debe ser explicativo como referencia para que el desarrollador se ubique dentro del entorno del análisis.

|  |  |  |
| --- | --- | --- |
| PRUEBAS | SÍ | NO |
| Existe una documentación interna (código) | X |  |
| Los nombres de las variables y etiquetas son significativos | X |  |
| Los comentarios son fáciles de entender | X |  |
| Existe una documentación externa | X |  |
| La ayuda es fácil de entender y manejar | X |  |

### Pruebas de seguridad y control

Objetivo: Verificación del control de acceso, validación de información y la realización correcta de las acciones sobre la aplicación.

|  |  |  |
| --- | --- | --- |
| PRUEBAS | SÍ | NO |
| Existe perfiles de usuario para el funcionamiento del sistema | X |  |
| Solicita contraseña al ingresar a la aplicación | X |  |

### Pruebas de calidad

Objetivo: Verificar la capacidad operativa y de adaptación a los cambios en la aplicación.

|  |  |  |
| --- | --- | --- |
| PRUEBAS | SÍ | NO |
| Existe portabilidad |  | X |
| El código es reutilizable | X |  |
| Es fácil de utilizar | X |  |
| Existe integridad | X |  |

## Informe de pruebas

* Comprobar la validación de información.
* Comprobar la lógica de la aplicación.
* Comprobar el nivel de integración de los menús.
* Probar el manejo de errores.

## Material revisado

**Formulario principal**

**Menú Archivos**

**Menú Docentes**

# Conclusiones y Recomendaciones

* Las herramientas de trabajo Visual C# .NET 2015 y SQL server 2010 Express, se interrelacionan y complementan bien en el desarrollo de proyectos de software, generando confianza en el proceso de construcción, existe buena documentación con respecto a ambas en Internet y por parte de Microsoft, son de fácil uso y manejo, el reporte de errores y al documentación de ellos es bastante buena y hace que el trabajo del desarrollador sea ágil al encontrar solución rápida y oportuna a los inconvenientes que se puedan presentar.
* Se propone este sistema, en aras de mejorar con la calidad educativa en distintas facultades de la universidad peruana.
* La experiencia específica en la UNMSM señala que en diversas facultades no existe documentación de reuniones de los docentes.
* Se recomienda, reutilizar los sistemas informáticos en las distintas facultades (Por ejemplo, Campus Virtual FII).

# Bibliografía

AEC.*Asociación Española para la Calidad*. Obtenido de Modelos de Calidad: https://www.aec.es/web/guest/centro-conocimiento/modelos-de-calidad

Ferguson,J. Patterson,B.Beres, J.Boutquin,P.Gupta,M. (2003). La Biblia de C# . Edit Anaya Multimedia. Madrid,España.

Kendall, K.Kendall,J.(2005). Análisis y diseño de sistemas.6ta. Ed., Edit. Pearson, México.

MINEDU.*Ministerio de Educación.* Obtenido de Estándares de Calidad Educativa: https://educacion.gob.ec/wp-content/uploads/downloads/2013/03/estandares\_2012.pdf

Parodi, C. (2013). Calidad de la Educación en el Perú.Diario *Gestión*.

SINEACE. (2016). *SINEACE.* Obtenido de Modelo de Acreditación para Programas de: https://www.sineace.gob.pe/wp-content/uploads/2014/08/Anexo-1-nuevo-modelo-programas-Resolucion-175.pdf

# Anexos

ACTA DE REUNIÓN

|  |  |  |  |
| --- | --- | --- | --- |
| **ACTA No. 24**  **COMITÉ DE CALIDAD** | **FECHA:**  **8 de julio de 2017** | **HORA INICIO**: 8 a.m.  **HORATERMINACIÓN:**  **10: 10 a.m.** | **LUGAR: FII. Piso 2°. Salón N° 1.** |
| **OBJETIVO DE LA REUNIÓN:**   * 1. A   2. B   3. C | | | |
| ***COORDINADOR DE LA REUNION:*** *Ing.Wiler Ponce* | | | |

CONVOCADOS / ASISTENTES

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRES Y APELLIDOS** | **FIRMA** | **ASISTIO** | |
| **SI** | **NO** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| **AGENDA** |
| 1. Verificación del quórum. 2. Lectura y aprobación del acta anterior. 3. Presentación situación actual del problema a tratar 4. Propuesta cronograma. 5. Revisión y aprobación de procedimientos. 6. Proposiciones y varios. |

|  |
| --- |
| **DESARROLLO DE LA AGENDA** |
| **1. Verificación del quórum.**  **2. Lectura y aprobación de las Actas # 22 y # 23 del 24 y 25 de Junio de 2017.**  SITUACIÓN ACTUAL:  PLAN DE ACCION PROPUESTO DEL PROBLEMA X:  Estrategia 1:  Estrategia 2:  Estrategia 3:  Estrategia 4:  **6. Revisión** y aprobación de procedimientos.  **7.** Proposiciones y varios. |
| **ACUERDOS** |

**TAREAS Y COMPROMISOS**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **TAREA / ACTIVIDAD** | **RESPONSABLE** | ***ENTREGA*** |
| **1.** |  | *Dr.Omar Álvarez* | *Inmediata.* |

Fecha de aprobación del acta: Julio de 2017

ELABORÓ: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

REVISÓ: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

En constancia firma:

|  |  |
| --- | --- |
| *NOMBRE – ROL* | *NOMBRE – ROL* |
| ***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*** | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
| *XXXXX* | *XXXX* |
| *Coordinador* | *Director* |

SCRIPT DE CREACIÓN DE BASE DE DATOS

use master

go

create database Bdproyecto

on primary

(

name=Bdproyecto,

filename='D:\BdProyecto\Bdproyecto.mdf',

size=10,

maxsize=50,

filegrowth=5

)

log on

(

name=Bdproyecto\_log,

filename='D:\BdProyecto\Bdproyecto.ldf',

size=5,

maxsize=unlimited,

filegrowth=10%

);

go

create table ActasReuniones

(

IdActas char(9) NOT NULL,

Nombre varchar(50),

Coordinador char(9) NOT NULL,

Fecha date not null,

Horainicio time not null,

Horafinal time not null,

Lugar varchar(20) NOT NULL,

Agenda varchar(200) NOT NULL,

idPropuesta char(9) NOT NULL,

idDesarrollo char(9) NOT NULL,

idAcuerdo char(9) NOT NULL,

idTareas char(9) NOT NULL,

NroAsistentes int null,

)

GO

--CREACION DE DESARROLLO DE REUNIONES

create table DesarrolloReuniones

( idDesarrollo char(9) NOT NULL,

Desarrollo varchar(200) NOT NULL,

)

GO

--CREACION DE TEMAS PROPUESTOS

create table TemasPropuestos

( idPropuesta char(9) NOT NULL,

Propuestas varchar(200) NOT NULL,

)

GO

--CREACION DE TABLA ACUERDOS

create table TemasAcuerdos

( idAcuerdo char(9) NOT NULL,

Acuerdos varchar(200) NOT NULL,

)

GO

--CREACION DE TAREAS ASIGNADAS

create table TareasAsignadas

( idTarea char(9) NOT NULL,

Tareas varchar(200) NOT NULL,

)

GO

--CREACION DE TABLA USUARIOS QUE USARAN EL SISTEMA

--create table UsuariosSistema

--(

--Usuario varchar(20) NOT NULL,

--Pass varchar(20) NOT NULL,

--)

use Bdproyecto

go

insert into Docentes values(77777771,'ACEVEDO BORREGO ADOLFO OSWALDO','99999991','user1@hotmail.com',0);

insert into Docentes values(77777772,'ANDIA VALENCIA WALTER','99999992','user2@hotmail.com',0);

insert into Docentes values(77777773,'ARROYO SALAZAR JORGE HUGO OMAR','99999993','user3@hotmail.com',0);

insert into Docentes values(77777774,'BELTRAN SARAVIA VICTOR ESTEBAN','99999994','user4@hotmail.com',0);

insert into Docentes values(77777775,'CACERES SALAZAR CESAR MARCELO','99999995','user5@hotmail.com',0);

insert into Docentes values(77777776,'CACHAY BOZA ORESTES','99999996','user6@hotmail.com',0);

insert into Docentes values(77777777,'CALSINA MIRAMIRA WILLY HUGO','99999997','user7@hotmail.com',0);

insert into Docentes values(77777778,'CAMPOS CONTRERAS CESAR','99999998','user8@hotmail.com',0);

insert into Docentes values(77777779,'CANCINO VERA NESTOR','99999999','user9@hotmail.com',0);

insert into Docentes values(77777710,'CEVALLOS AMPUERO JUAN MANUEL','99999910','user10@hotmail.com',0);

go

insert into Evaluadores values(66666661,'OLIVA MOYA RONNIE','88888881','fii1@hotmail.com','user1','pass1');

insert into Evaluadores values(66666662,'ARMAS BENAVIDES RAUL','88888882','fii2@hotmail.com','user2','pass2');

insert into Evaluadores values(66666663,'TACO ANTON RODRIGO','88888883','fii3@hotmail.com','user3','pass3');

insert into Evaluadores values(66666664,'PEÑA SERQUEN JORGE','88888884','fii4@hotmail.com','user4','pass4');

insert into Evaluadores values(66666665,'ALTEZ INCISO RICARDO','88888885','fii5@hotmail.com','user5','pass5');

go

insert into Administradores values(55555551,'CORREA DELGADO RAFAEL','33333331','adm1@hotmail.com','user6','pass6');

insert into Administradores values(55555552,'MULDER BEDOYA MAURICIO','33333332','adm2@hotmail.com','user7','pass7');

insert into Administradores values(55555553,'GARCIA BELAUNDE VICTOR','33333333','adm3@hotmail.com','user8','pass8');

insert into Administradores values(55555554,'VICENTE VALVERDE GRACE','33333334','adm4@hotmail.com','user9','pass9');

insert into Administradores values(55555555,'POMA LICETA LESLIE','33333335','adm5@hotmail.com','user10','pass10');

GO

insert into ActasReuniones values(1,'ACTA N1',77777778,55555551,66666661,'2017/02/5','UNI','CALIFICACIONES',0);

insert into ActasReuniones values(2,'ACTA N2',77777778,55555552,66666662,'2017/03/15','UNMSM','CAMPUS VIRTUAL',0);

insert into ActasReuniones values(3,'ACTA N3',77777778,55555553,66666663,'2017/03/20','UNAC','CALIFICACIONES',0);

insert into ActasReuniones values(4,'ACTA N4',77777779,55555554,66666664,'2017/03/31','ESAN','CAMPUS VIRTUAL',0);

insert into ActasReuniones values(5,'ACTA N5',77777779,55555555,66666665,'2017/04/20','ESAN','CALIFICACIONES',0);

GO

--Como asignar un propietario a la base de bd:

use Bdproyecto

go

exec sp\_changedbowner sa

ALTER TABLE ActasReuniones add primary key (IdActas)

go

ALTER TABLE Docentes add primary key (idDocente)

go

ALTER TABLE DesarrolloReuniones add primary key (idDesarrollo)

go

ALTER TABLE TareasAsignadas add primary key (idTarea)

go

ALTER TABLE TemasPropuestos add primary key (idPropuesta)

go

ALTER TABLE TemasAcuerdos add primary key (idAcuerdo)

go

ALTER TABLE ActasReuniones add foreign key (Coordinador) references Docentes

go

ALTER TABLE ActasReuniones add foreign key (idAcuerdo) references TemasAcuerdos

go

ALTER TABLE ActasReuniones add foreign key (idPropuesta) references TemasPropuestos

go

ALTER TABLE ActasReuniones add foreign key (idDesarrollo) references DesarrolloReuniones

go

ALTER TABLE ActasReuniones add foreign key (idTareas) references TareasAsignadas

go

Código en C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AppAdo03.Entity

{

public class ActaTO

{

//propiedades

public string IdActa { get; set; }

public string Nombre { get; set; }

public DateTime Fecha { get; set; }

public string Objetivo { get; set; }

public string Lugar { get; set; }

public string Coordinador { get; set; }

public int NroAsistentes { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AppAdo03.Entity

{

public class DocenteTO

{

public string IdDocente { get; set; }

public string DNI { get; set; }

public string Correo { get; set; }

public string Nombre { get; set; }

public int nroparticipa { get; set; }

}

}

Clase conexión a la base de datos

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using System.Configuration;

namespace AppAdo03.DataBase

{

public class AccesoDB

{

public static SqlConnection Conexion()

{

SqlConnection cn = new SqlConnection(ConfigurationManager.ConnectionStrings["Neptuno"].ConnectionString);

return cn;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AppAdo03.Service

{

public interface ICrudDao<T>

{

//definir las firmas

int create(T t);

int update(T t);

int delete(T t);

T findForId(object t);

List<T> readAll();

}

}

Model

Clase ActaDAO

using AppAdo03.Entity;

using AppAdo03.Service;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

using AppAdo03.DataBase;

namespace AppAdo03.Model

{

public class ActaDAO : ICrudDao<ActaTO>

{

// variables

SqlCommand cmd;

SqlDataReader dr;

int ok;

//crear actas

public int create(ActaTO t)

{

try

{

using (var cn=AccesoDB.Conexion())

{ //procedimiento almacenado para agregar una nueva acta

cmd = new SqlCommand("usp\_Acta\_Adicionar",cn);

cmd.CommandType = CommandType.StoredProcedure;

// prepar parametros con su valores para el sp

cmd.Parameters.AddWithValue("@Idacta",t.IdActa);

cmd.Parameters.AddWithValue("@Nombre", t.Nombre);

cmd.Parameters.AddWithValue("@Fecha", t.Fecha);

cmd.Parameters.AddWithValue("@Lugar",t.Lugar);

cmd.Parameters.AddWithValue("@Objetivo", t.Objetivo);

cmd.Parameters.Add("@IdActa", SqlDbType.Int).Direction = ParameterDirection.Output;

//ejecutar sp

cn.Open();

ok = cmd.ExecuteNonQuery() == 1 ? 1 : -1;

t.IdActa = (string)cmd.Parameters["@Idacta"].Value;

}

}

catch (SqlException ex)

{

throw ex;

}

return ok;

}

//borrar actas

public int delete(ActaTO t)

{

try

{

using (var cn = AccesoDB.Conexion())

{

cmd = new SqlCommand("usp\_Acta\_Eliminar", cn);

cmd.CommandType = CommandType.StoredProcedure;

// prepar parametro con su valor para el sp

cmd.Parameters.AddWithValue("@Idacta", t.IdActa);

//ejecutar sp

cn.Open();

ok = cmd.ExecuteNonQuery() == 1 ? 1 : -1;

}

}

catch (SqlException ex)

{

throw ex;

}

return ok;

}

//buscar codigo de actas

public ActaTO findForId(object t)

{

ActaTO pro = null;

try

{

using (var cn = AccesoDB.Conexion())

{

cmd = new SqlCommand("usp\_Acta\_Datos", cn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@IdProducto", Convert.ToInt32(t));

cn.Open();

dr = cmd.ExecuteReader();

if (dr.Read())

{

ActaTO ac = new ActaTO()

{

IdActa = dr[0].ToString(),

Nombre = dr[1].ToString(),

//Fecha= dr[2].ToString(), convertir a fecha falta

Objetivo = dr[3].ToString(),

Lugar = dr[4].ToString(),

NroAsistentes = Convert.ToInt32(dr[5]),

};

}

dr.Close();

}

}

catch (SqlException ex)

{

throw ex;

}

return pro;

}

//ver todas las actas

public List<ActaTO> readAll()

{

List<ActaTO> lista = new List<ActaTO>();

try

{

using (var cn = AccesoDB.Conexion())

{

cmd = new SqlCommand("usp\_Acta\_Listar", cn);

cmd.CommandType = CommandType.StoredProcedure;

cn.Open();

dr = cmd.ExecuteReader(CommandBehavior.SingleResult);

while (dr.Read())

{

ActaTO ac = new ActaTO()

{

IdActa = dr[0].ToString(),

Nombre=dr[1].ToString(),

//Fecha= dr[2].ToString(), convertir a fecha falta

Objetivo=dr[3].ToString(),

Lugar= dr[4].ToString(),

NroAsistentes= Convert.ToInt32(dr[5]),

};

lista.Add(ac);

}

dr.Close();

}

}

catch (SqlException ex)

{

//throw ex;

}

return lista;

}

Clase DocenteDAO

using AppAdo03.DataBase;

using AppAdo03.Entity;

using AppAdo03.Service;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AppAdo03.Model

{

public class DocenteDAO : ICrudDao<DocenteTO>

{

SqlCommand cmd;

SqlDataReader dr;

int okm;

// crear docente

public int create(DocenteTO t)

{

try

{

using (var cn = AccesoDB.Conexion())

{ //procedimiento almacenado para agregar una nueva acta

cmd = new SqlCommand("usp\_Docente\_Adicionar", cn);

cmd.CommandType = CommandType.StoredProcedure;

// prepar parametros con su valores para el sp

cmd.Parameters.AddWithValue("@IdDocente", t.IdDocente);

cmd.Parameters.AddWithValue("@Nombre", t.Nombre);

cmd.Parameters.AddWithValue("@DNI", t.DNI);

cmd.Parameters.AddWithValue("@Correo", t.Correo);

cmd.Parameters.Add("@IdDocente", SqlDbType.Int).Direction = ParameterDirection.Output;

//ejecutar sp

cn.Open();

okm = cmd.ExecuteNonQuery() == 1 ? 1 : -1;

t.IdDocente = (string)cmd.Parameters["@IdDocente"].Value;

}

}

catch (SqlException ex)

{

throw ex;

}

return okm;

}

//borrar docente

public int delete(DocenteTO t)

{

try

{

using (var cn = AccesoDB.Conexion())

{

cmd = new SqlCommand("usp\_Docente\_Eliminar", cn);

cmd.CommandType = CommandType.StoredProcedure;

// prepar parametro con su valor para el sp

cmd.Parameters.AddWithValue("@IdDocente", t.IdDocente);

//ejecutar sp

cn.Open();

okm = cmd.ExecuteNonQuery() == 1 ? 1 : -1;

}

}

catch (SqlException ex)

{

throw ex;

}

return okm;

}

//buscar codigo de docentes

public DocenteTO findForId(object t)

{

DocenteTO pro = null;

try

{

using (var cn = AccesoDB.Conexion())

{

cmd = new SqlCommand("usp\_Docente\_Datos", cn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@IdProducto", Convert.ToInt32(t));

cn.Open();

dr = cmd.ExecuteReader();

if (dr.Read())

{

pro = new DocenteTO()

{

//IdProducto = Convert.ToInt32(dr[0]),

//NombreProducto = dr[1].ToString(),

//IdProveedor = Convert.ToInt32(dr[2]),

//IdCategoria = Convert.ToInt32(dr[3]),

//Precio = Convert.ToDecimal(dr[4]),

//Stock = Convert.ToInt32(dr[5]),

};

}

dr.Close();

}

}

catch (SqlException ex)

{

throw ex;

}

return pro;

}

//ver todas las actas

public List<DocenteTO> readAll()

{

List<DocenteTO> lista = new List<DocenteTO>();

try

{

using (var cn = AccesoDB.Conexion())

{

cmd = new SqlCommand("usp\_Docente\_Listar", cn);

cmd.CommandType = CommandType.StoredProcedure;

cn.Open();

dr = cmd.ExecuteReader(CommandBehavior.SingleResult);

while (dr.Read())

{

DocenteTO doc = new DocenteTO()

{

IdDocente = dr[0].ToString(),

Nombre = dr[1].ToString(),

//Fecha= dr[2].ToString(), convertir a fecha falta

//Objetivo = dr[3].ToString(),

//Lugar = dr[4].ToString(),

//NroAsistentes = Convert.ToInt32(dr[5]),

};

lista.Add(doc);

}

dr.Close();

}

}

catch (SqlException ex)

{

//throw ex;

}

return lista;

}

//modificar o actualizar actas

public int update(DocenteTO t)

{

try

{

using (var cn = AccesoDB.Conexion())

{

cmd = new SqlCommand("usp\_Docentes\_Actualizar", cn);

cmd.CommandType = CommandType.StoredProcedure;

// prepar parametros con su valores para el sp

cmd.Parameters.AddWithValue("@Nombre", t.Nombre);

cmd.Parameters.AddWithValue("@Idacta", t.IdDocente);

//cmd.Parameters.AddWithValue("@Fecha", t.Fecha);

//cmd.Parameters.AddWithValue("@Objetivo", t.Objetivo);

//cmd.Parameters.AddWithValue("@Lugar", t.Lugar);

//cmd.Parameters.AddWithValue("@NroAsistentes", t.NroAsistentes);

//ejecutar sp

cn.Open();

okm = cmd.ExecuteNonQuery() == 1 ? 1 : -1;

}

}

catch (SqlException ex)

{

throw ex;

}

return okm;

}

}

}

Controller

using AppAdo03.Entity;

using AppAdo03.Model;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AppAdo03.Controller

{

public class ActaBLL

{

// variable de la clase ProductoDAO

ActaDAO dao;

//constructor

public ActaBLL()

{

dao = new ActaDAO();

}

// metodos de negocio

public int ProductoAdicionar(ActaTO p)

{

return dao.create(p);

}

public int ProductoActualizar(ActaTO p)

{

return dao.update(p);

}

public int ProductoEliminar(ActaTO p)

{

return dao.delete(p);

}

public ActaTO ProductoBuscar(object p)

{

return dao.findForId(p);

}

public List<ActaTO> ProductoListar()

{

return dao.readAll();

}

public DataTable CategoriaListar()

{

return dao.readAllCategorias();

}

public DataTable ProveedorListar()

{

return dao.readAllProveedores();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using AppAdo03.Model;

using AppAdo03.Entity;

namespace AppAdo03.Controller

{

public class DocenteBLL

{

DocenteDAO dao;

public DocenteBLL()

{

dao = new DocenteDAO();

}

// metodos de negocio

public int ProductoAdicionar(DocenteTO p)

{

return dao.create(p);

}

public int ProductoActualizar(DocenteTO p)

{

return dao.update(p);

}

public int ProductoEliminar(DocenteTO p)

{

return dao.delete(p);

}

public DocenteTO ProductoBuscar(object p)

{

return dao.findForId(p);

}

public List<DocenteTO> ProductoListar()

{

return dao.readAll();

}

}

}

INTERFAZ

Clase MDIMenu

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AppAdo03.View

{

public partial class MDImenu : Form

{

public MDImenu()

{

InitializeComponent();

}

private void productosToolStripMenuItem\_Click(object sender, EventArgs e)

{

ActaView p = new ActaView();

p.MdiParent = this;

p.Show();

}

private void salirToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void verDocentesToolStripMenuItem\_Click(object sender, EventArgs e)

{

DocenteView p = new DocenteView();

p.MdiParent = this;

p.Show();

}

private void menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

}

}

Clase ActaView

using AppAdo03.Controller;

using AppAdo03.Entity;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AppAdo03.View

{

public partial class ActaView : Form

{

public ActaView()

{

InitializeComponent();

listaProductos();

}

//instanciar objeto de la clase ProductoBLL

ActaBLL opro = new ActaBLL();

ActaTO pro;

private void listaProductos()

{

dgvProducto.DataSource = opro.ProductoListar();

}

private void ProductoView\_Load(object sender, EventArgs e)

{

try

{

cargaCombos();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void cargaCombos()

{

//cboProveedor.DataSource = opro.ProveedorListar();

//cboProveedor.DisplayMember = "NombreCompañía";

//cboProveedor.ValueMember = "IdProveedor";

// cboCategoria.DataSource = opro.CategoriaListar();

//cboCategoria.DisplayMember = "NombreCategoría";

//cboCategoria.ValueMember = "IdCategoría";

}

private void btnAdicionar\_Click(object sender, EventArgs e)

{

procesar(1);

}

private void btnActualizar\_Click(object sender, EventArgs e)

{

procesar(2);

}

private void btnEliminar\_Click(object sender, EventArgs e)

{

procesar(3);

}

private void btnCerrar\_Click(object sender, EventArgs e)

{

Dispose();

}

private void btnBuscar\_Click(object sender, EventArgs e)

{

procesar(4);

}

private void procesar(int opcion)

{

int ok=0;

string msg = "";

try

{

switch (opcion)

{

case 1:

ok = opro.ProductoAdicionar(leeProducto());

msg = "Producto registrado con exito";

break;

case 2:

ok = opro.ProductoActualizar(leeProducto());

msg = "Producto actualizado con exito";

break;

case 3:

ok = opro.ProductoEliminar(leeProducto());

msg = "Producto eliminado con exito";

break;

case 4:

consultarProducto();

return;

}

MessageBox.Show(msg, "exito");

listaProductos();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "error");

}

}

private void consultarProducto()

{

pro = opro.ProductoBuscar(txtCodigo.Text);

if (pro!=null)

{

//txtNombre.Text = pro.NombreProducto;

// cboProveedor.SelectedValue = pro.IdProveedor;

//cboCategoria.SelectedValue = pro.IdCategoria;

//txtPrecio.Text = pro.Precio.ToString();

// numCantidad.Value = pro.Stock;

}

else

{

MessageBox.Show("Producto no existe", "aviso");

txtCodigo.SelectAll();

txtCodigo.Focus();

}

}

private ActaTO leeProducto()

{

//crear objeto pro

pro = new ActaTO()

{

//IdProducto=int.Parse(txtCodigo.Text),

//NombreProducto=txtNombre.Text,

//IdProveedor=(int)cboProveedor.SelectedValue,

//IdCategoria=(int)cboCategoria.SelectedValue,

// Precio=decimal.Parse(txtPrecio.Text),

//Stock=(int)numCantidad.Value

};

return pro;

}

private void dgvProducto\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

}

1. También se tiene que considerar los requerimientos no funcionales, que por cuestiones de tiempo y de recursos no se ha podido realizar. Sin embargo, es necesario recalcar que cualquier sistema que se implemente debe OBLIGATORIAMENTE considerar elementos como tiempo de respuesta del sistema entre otros. [↑](#footnote-ref-2)
2. Este diagrama se desarrolló con el software IBM Rational Rose 2007 como elaboración propia [↑](#footnote-ref-3)
3. Cabe resaltar que para la creación de las tablas se pasó por el proceso de normalización. [↑](#footnote-ref-4)