

**Segunda tarea programada.**

**Uno. Objetivo.**

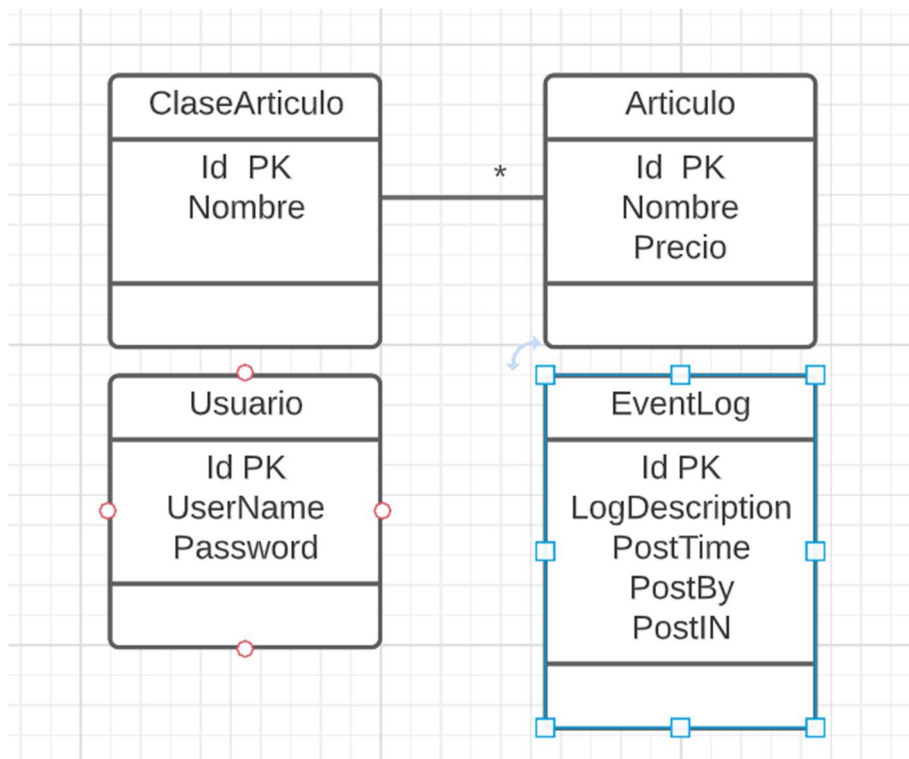
Implementar una aplicación sencilla que realiza validaciones de usuario, consultas sobre varias tablas en la BD con despliegue de información en una interfaz de usuario, la captura y edición de valores que salvan en una tabla (CRUD), el registro de actividades en la BD a través de una bitácora de eventos, la implementación de manejo de errores en procedimientos almacenados y la escritura de transacciones de BD.

**Dos. Descripción.**

Se trata de implementar una aplicación sencilla para consultar y hacer CRUD de una tabla de artículos (como parte de un sistema de inventarios o un POS), y realizar algunas consultas con filtros por clase de articulo, cantidad o nombre del artículo. La aplicación incluye validación de ingreso de usuarios mediante clave y contraseña, así como un registro en una bitácora de todas las operaciones que un usuario realiza en la BD.

**Dos. 1. Modelo Conceptual**

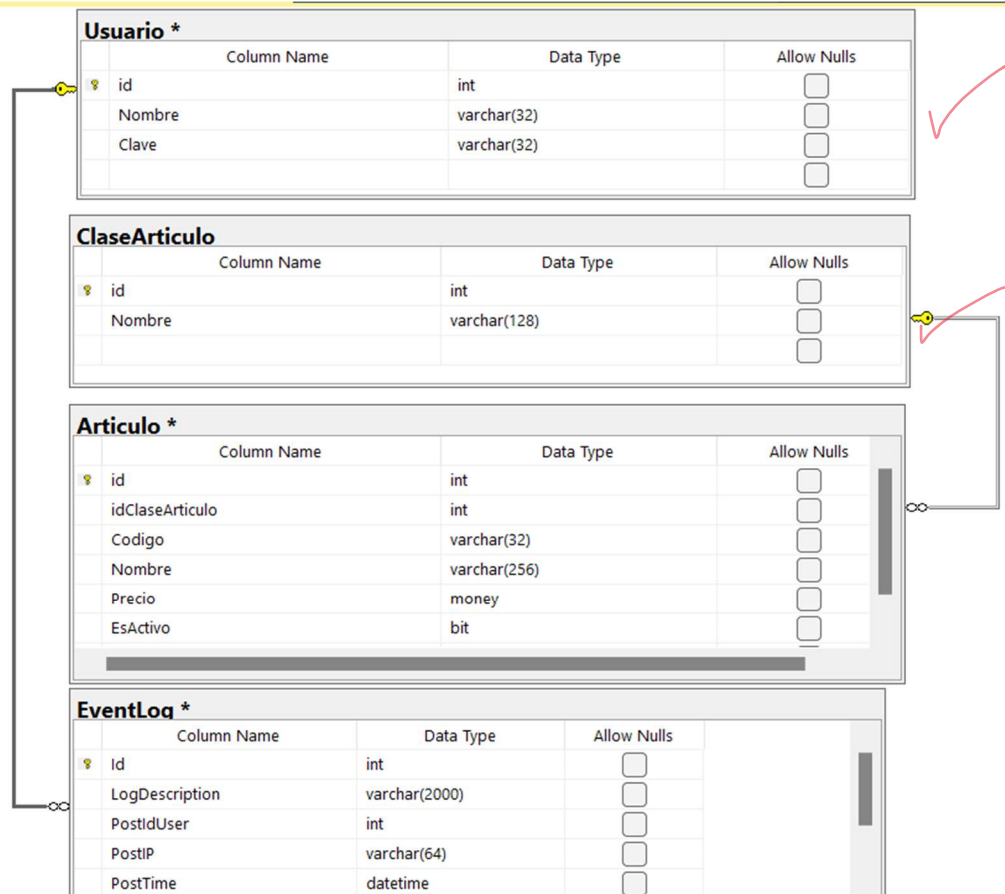
El modelo conceptual en el que se basa la tarea es este:



(En la tabla Artículo, falta el campo código de artículo VARCHAR (32) y EsActivo BIT default 1).

## Dos.2 Modelo físico.

En el modelo físico (en la base de datos) se implementarán las siguientes tablas:



Lo anterior puede declararse, de manera no gráfica, mediante un script.

```
CREATE TABLE dbo.Usuario
(
    id INT IDENTITY (1, 1) NOT NULL PRIMARY KEY
    , UserName VARCHAR(16) NOT NULL
    , Password VARCHAR (16) NOT NULL
);

CREATE TABLE dbo.EventLog (
    Id int IDENTITY(1,1) NOT NULL PRIMARY KEY
    , LogDescription varchar(2000) NOT NULL
    , PostIdUser int NOT NULL
    , PostIP varchar(64) NOT NULL
    , PostTime datetime NOT NULL
);

ALTER TABLE [dbo].[EventLog] WITH CHECK ADD CONSTRAINT
[FK_EventLog_Usuario] FOREIGN KEY([PostIdUser])
REFERENCES [dbo].[Usuario] ([id])
```

```
CREATE TABLE dbo.ClaseArticulo
(
    id INT IDENTITY (1, 1) PRIMARY KEY
    , Nombre VARCHAR(64)
);
```

```
CREATE TABLE dbo.Articulo
(
    id INT IDENTITY (1, 1) PRIMARY KEY
    , IdClaseArticulo INT
    , Codigo VARCHAR(32) NOT NULL
    , Nombre VARCHAR(128) NOT NULL
    , Precio MONEY NOT NULL
);
```

```
ALTER TABLE [dbo].[Articulo] WITH CHECK ADD CONSTRAINT
[FK_Articulo_ClaseArticulo] FOREIGN KEY([IdClaseArticulo])
REFERENCES [dbo].[ClaseDeArticulo] ([id])
```

Además, se agrega una nueva tabla, que no pertenece propiamente a la aplicación aunque es importante para el registro de errores en la BD captados desde Try Catch.

```
CREATE TABLE [dbo].[DBErrors](
    [ErrorID] [int] IDENTITY(1,1) NOT NULL,
    [UserName] [varchar](100) NULL,
    [ErrorNumber] [int] NULL,
    [ErrorState] [int] NULL,
    [ErrorSeverity] [int] NULL,
    [ErrorLine] [int] NULL,
    [ErrorProcedure] [varchar](max) NULL,
    [ErrorMessage] [varchar](max) NULL,
    [ErrorDateTime] [datetime] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

### **Dos.3 Interfaz de Usuario.**

En la capa presentación, la primera página es de ingreso y consiste en capturar el UserName y el Password de un usuario, si la combinación de ambos valores no está en la BD, se muestra un error que dice “Combinación de usuario/password no existe en la BD”.

Luego se muestra una página que mostrará todos los códigos y nombres de articulo, el nombre de clase de articulo y su precio en orden alfabético ascendente por nombre de artículo.

En la parte superior mostrará: 5 botones, con las siguientes funciones:

- 2 botones corresponden a la realización de búsquedas de artículos que responden a un patrón que se especifica en una caja de texto,
- 1 botón se asocia a una búsqueda de artículos que corresponden a una clase de artículo la cual se selecciona de una lista.
- Tres botones que se asocian con un CRUD sobre la tabla de artículos; y,
- finalmente, un botón de salir.

#### **Botón de filtrar por nombre:**

<Caja de texto que captura el filtro> [Filtrar nombre]

Si la caja de texto está vacía y se da clic en “Filtrar nombre”, se despliegan todos los artículos y su precio en orden alfabético ascendente por nombre del artículo.

Si la caja de texto NO está vacía se da clic en “Filtrar nombre”, se despliegan todos los artículos cuyo nombre contengan el texto en la caja.

Por ejemplo: si el filtro consiste en “la”, se desplegarán todos los artículos que contengan en su nombre la cadena “la”, tales como “la escalera”, “llaves”, “taladro”, se ignoran mayúsculas y minúsculas.

En la caja de texto (el filtro) puede aparecer cualquier carácter válido del teclado, no hay que validar nada.

Los valores que se listan deben ser Código del artículo, Nombre del artículo, nombre de la clase del artículo y el precio.

#### **Botón de filtrar por cantidad:**

<Caja de texto que captura el filtro> [Filtrar Cantidad]

Si la caja de texto está vacía y se da clic en “Filtrar Cantidad”, se despliegan todos los artículos y su precio en orden alfabético ascendente por nombre del artículo.

Si la caja de texto NO está vacía y se da clic en “Filtrar Cantidad”, se despliegan los primeros N artículos y su precio según su orden alfabético ascendente por nombre del artículo. Donde N es valor capturado en la caja de texto (filtro).

Debe validarse que el valor ingresado en la caja sea siempre un entero, el botón solo se habilita si la caja está vacía o el valor ingresado es un entero.

Los valores que se listan deben ser Código del artículo, el nombre de la clase de artículo, nombre del artículo, y el precio.

### **Botón de filtrar por clase de artículo.**

Cuando la aplicación sube, debe llenarse una lista con los nombres de clases artículos, esta lista debe estar ordenada alfabéticamente. Cuando se da clic en este botón, se mostrará la lista de los artículos cuya clase de artículo corresponda a aquella seleccionada por el usuario en la lista de clases de artículo; por default, en la lista se selecciona la primera clase de artículo, en orden alfabético.

Los valores que se listan deben ser Código del artículo, nombre del artículo, nombre de la clase del artículo y el precio.

### **Botón de insertar artículo.**

Ya sea en la página principal o en una nueva página, se habilitan una lista y tres cajas de texto, la lista contendrá todos los nombres de clase de artículo en orden alfabético, y tres cajas de texto son para capturar el código del artículo, el nombre del artículo y otra para capturar el precio.

Cuando la lista de clase de artículo está seleccionada y las tres cajas de texto (código, nombre de artículo y precio) están llenas y validadas, se habilitará un botón que se llama OK, y al dar clic en él, se insertan una nueva instancia en la tabla Artículo. Habrá un botón con nombre “cancelar” o “regresar” que anula la operación y regresa el control a la página principal.

Desde la interfaz de usuario (o sea NO desde la capa de datos). Se debe validar que la caja de texto que captura el precio valide que el precio es un valor monetario bien formado.

No pueden haber 2 o más artículos que se llamen igual, si al insertar se detecta que ya existe un artículo con ese nombre, debe enviar un mensaje de error (“Artículo con nombre duplicado”) y anular la inserción.

No pueden haber 2 o más artículos con el mismo código de artículo, si al insertar se detecta que ya existe un artículo con mismo código, debe enviar un mensaje de error (“Artículo con código duplicado”) y anular la inserción.

### **Botón de actualizar artículo.**

La selección del artículo se hará por la captura del código del artículo desde una caja de texto, se debe validar que artículo exista. Si sí existe, se muestran los campos (nombre de la clase de artículo, código y nombre de artículo, y precio), se permite la actualización de todos estos campos, la selección de la clase de artículo debe hacerse desde una lista, se debe validar que el nuevo nombre de artículo no sea repetido, así como el código tampoco puede ser repetido, en este caso se debe enviar un mensaje de error que diga “No puede actualizar nombre de artículo con un nombre ya existente”

o “No puede actualizar código de artículo con un código ya existente”. Estas validaciones deben realizarse en capa de datos.

Un intento de actualizar un artículo que no existe, debe generar una entrada en la tabla de EventLog, así como un intento de actualizar el nombre con un nombre que ya existe, o el intento de actualizar el código de artículo con uno ya existente.

La validación de que la captura del precio sea un valor monetario bien formado debe hacerse en capa de presentación (mediante html y javascript).

### **Botón de borrar artículo.**

La selección del artículo que se hará por la captura del código del artículo desde una caja de texto, se debe validar que el artículo exista. Si sí existe, se muestra los campos de código y nombre del artículo, nombre de la clase de artículo y precio, todos estos campos en modo solo lectura, y una alerta indicando, “Confirme que desea eliminar este artículo (S/N).” El borrado exitoso o no exitoso, así como el intento de borrado no confirmado debe ser registrado en la bitácora.

El borrado del artículo es lógico, o sea no es físico. Se actualiza el campo EsActivo, de 1 (default) a 0.

### **Botón de salir.**

El control se devuelve a la página que solicita el usuario y password.

### **Refrescamiento luego de ejecutar cualquier acción.**

Una vez que se da clic a cualquier botón y se realiza una operación (consultar con filtro, insertar, modificar o borrar), se debe refrescar el despliegue de la lista artículos, y las cajas de filtros estarán vacías.

### **Tres. Sketch de la interfaz de usuario.**

La interfaz principal, muestra dos áreas, la superior contiene los encabezados, las cajas de texto, la lista de clases de artículo y los botones. La inferior muestra el resultado de las consultas para cuando se dan botones de consulta (consulta por nombre, por cantidad o por clase de artículo). Los botones correspondientes al CUD (crear, actualizar-update y borrar-delete), levantan otra interfaz de usuario.

Sección de encabezado, cajas de texto  
para filtros, botones para consultas o CRUD

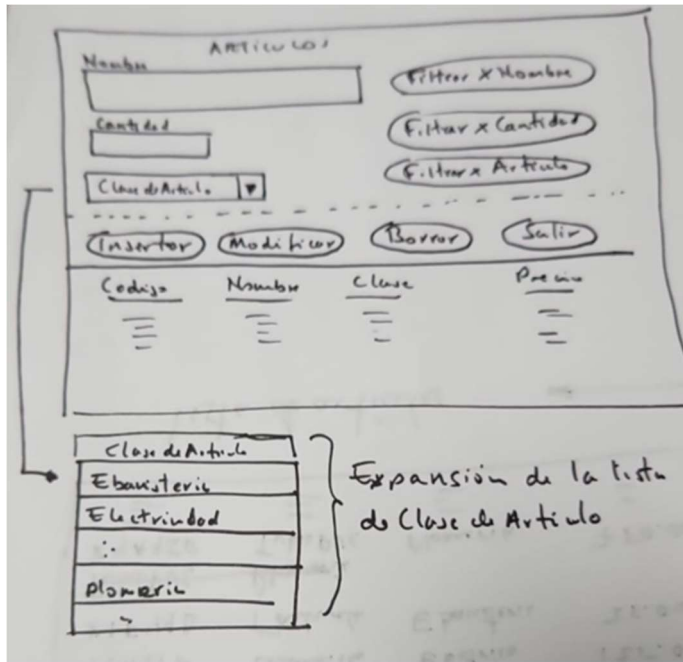
<u>Código</u>	<u>Nombre</u>	<u>Clase</u>	<u>Precio</u>
75A25F	Bombillo	Eléctrico	125.00
81F49D	Tachuela	Ebanistería	25.00
<del>Tubo PVC</del>	<del>Plomería</del>		
84A95F	Tubo PVC	Plomería	750.00
=	=	=	=

lista de artículos

El id del artículo, no se muestra, por razones de seguridad.

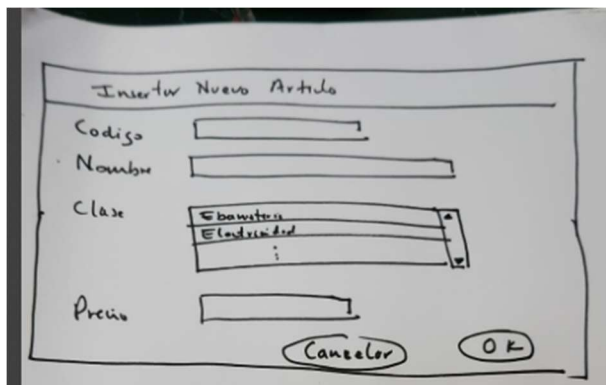
El área superior, que muestra botones y cajas, es como lo que sigue.



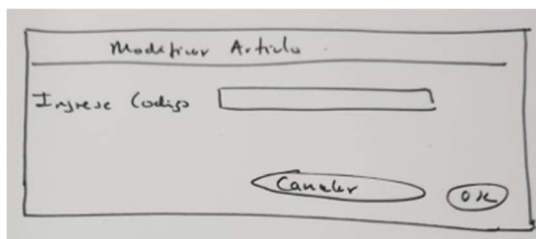


Las figuras cuadradas son cajas de texto o listas, las figuras redondeadas son botones, la lista se llena con los nombres de todas las clases de articulo en orden alfabético, por default estará seleccionado el primer nombre.

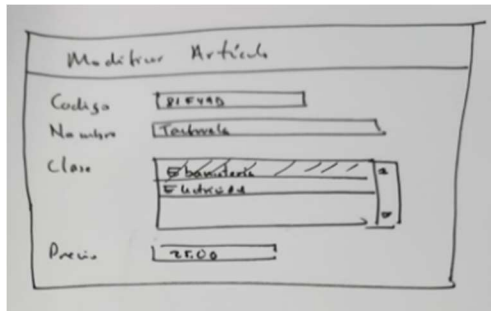
La inserción de artículos levanta una interfaz de usuario similar a la siguiente:



La actualización se hace mediante una interfaz similar a la siguiente:



Si el artículo no se pudo localizar se envía un mensaje de error y el intento de modificación se registra en la bitácora (tabla EventLog), si sí se pudo localizar se presenta esta pantalla:



Modificar Artículo

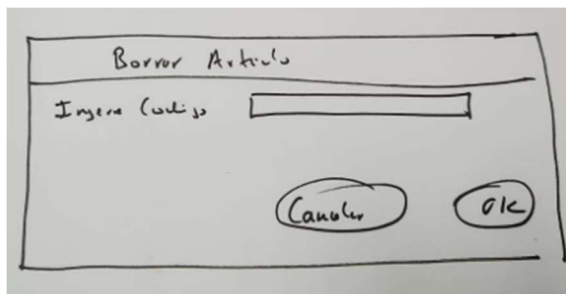
Codigo: 81F49D

Nombre: Tachuela

Clase: Ebanisteria

Precio: 25.00

El borrado levanta una interfaz similar a la siguiente:

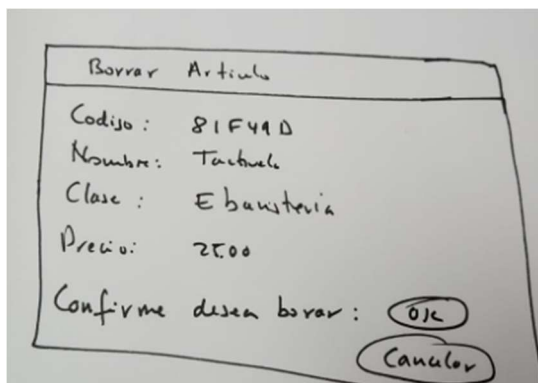


Borrar Artículo

Ingresa Codigo:

Cancelar OK

Si el artículo no se pudo localizar se envía un mensaje de error y el intento de modificación se registra en la bitácora (tabla EventLog), si sí el artículo se pudo localizar se presenta la siguiente alerta:



Borrar Artículo

Codigo: 81F49D

Nombre: Tachuela

Clase: Ebanisteria

Precio: 25.00

Confirme desea borrar: OK Cancelar

#### Cuatro. La Bitácora de actividades, el EventLog.

Por cada acción ya sea login, logout, alguna consulta o cualquier operación del CRUD de un artículo (ya sea que la inserción, actualización o borrado independientemente de si fue exitosa o no exitosa, y su resultado), debe guardarse en la tabla EventLog una instancia que describa la acción del usuario que activó la operación.

En los campos PostTime se almacena una estampa de tiempo (fecha, hora, minuto, segundo y milésima de segundo, función getdate del MSSQL) del acceso, PostByUserId el id del usuario, sacado de la tabla user, que lo realiza; y el PostIn, la IP de la estación que realiza el acceso (el cual se obtiene del contexto del browser). En el campo de LogDescription, ira un json que será con la siguiente estructura :

{TipoAccion=<Un Tipo de accion> Description=<Valor del Descripcion>}

Los tipos de acción y descripción tendrán los siguientes valores

<b>Tipo de Acción</b>	<b>Valor de Descripción</b>
“Login exitoso”	<nombre de usuario>
“Login no exitoso”	<nombre de usuario>
“Consulta por Nombre”	<Valor del filtro>
“Consulta por cantidad”	<Valor de la cantidad>
“Consulta por clase de articulo”	<Valor de Clase de articulo>
“Insertar articulo exitoso”	String de valores separados por coma que en este orden <IdArticulo>,<IdClaseArticulo>,<Codigo de articulo>,<NombreArticulo>,<Cantidad>
“Insertar articulo no exitoso” Descripción del error: nombre de articulo duplicado, código de articulo duplicado o error de la BD.	<IdArticulo>,<IdClaseArticulo>,<CodigoArticulo>,<NombreArticulo>,<Cantidad>,<descripcion del error>
“Intento de modificar articulo” Comentario: hubo intento si al final por alguna razón la inserción no se hizo debido a que se genero algún error o porque el usuario canceló	<Codigo de articulo>,<Resultado:”articulo existe” o “articulo no existe”>
“Modificación de articulo exitoso”	<IdArticulo>,<IdClaseArticulo anterior>,<Codigo de articulo anterior>,<NombreArticulo anterior>,<Cantidad anterior>,<IdClaseArticulo nuevo>,<Codigo de articulo nuevo>,<NombreArticulo nuevo>,<Cantidad nueva>
“Modificación de articulo no exitoso”	<IdArticulo>,<IdClaseArticulo anterior>,<Codigo de articulo anterior>,<NombreArticulo anterior>,<Cantidad anterior>,<IdClaseArticulo nuevo>,<Codigo de articulo nuevo>,<NombreArticulo nuevo>,<Cantidad nueva>,<Descripcion del error>

"Intento de borrar articulo"	<Codigo de articulo>, <Resultado:"articulo existe" o "articulo no existe" o "usuario no confirmo borrado">
"Borrado de articulo exitoso"	<IdArticulo>,<IdClaseArticulo>, <CodigoArticulo>, <NombreArticulo>, <Cantidad>
"Borrado de articulo no exitoso"	<IdArticulo>,<IdClaseArticulo>, <CodigoArticulo>, <NombreArticulo>, <Cantidad>, <descripcion del error>
"Logout"	""

### Cinco. Transacción de base de datos y manejo de errores.

Ya que todas las operaciones, incluso las consultas harán una inserción en la tabla de Eventlog en la BD; será necesario que el código de los SP haga manejo de errores, mediante Try-Catch. En caso del SP para los CRUD que harán alguna operación sobre los artículos y además harán una inserción en el EventLog, o sea que se están actualizando al menos 2 objetos en la BD; será necesario codificar una transacción de base de datos. No olvidar que cuando se actualiza un solo objeto en la BD, iniciar una transacción de BD (Begin tran, commit tran o rollback tran) es redundante, pues el SABD crea una transacción implícita.

Para el manejo de errores en el catch, deben crear una tabla con la siguiente estructura:

```
CREATE TABLE [dbo].[pBErrors](
    [ErrorID] [int] IDENTITY(1,1) NOT NULL,
    [UserName] [varchar](100) NULL,
    [ErrorNumber] [int] NULL,
    [ErrorState] [int] NULL,
    [ErrorSeverity] [int] NULL,
    [ErrorLine] [int] NULL,
    [ErrorProcedure] [varchar](max) NULL,
    [ErrorMessage] [varchar](max) NULL,
    [ErrorDateTime] [datetime] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

### Seis. Datos de prueba.

Los datos de prueba se cargan desde un archivo XML que contiene la siguiente estructura.

```
<Usuarios>
  <usuario Nombre="PRojas" Password="Mipass123*" />
  <usuario Nombre="LPerez" Password="Nadie124**" />
  <usuario Nombre="AVindas" Password="adiVINE12&" />
  <usuario Nombre="JRamirez" Password="Mefui44" />
</Usuarios>
```

```

<ClasesdeArticulos>
  <ClasesdeArticulos Nombre="Plomeria"/>
  <ClasesdeArticulos Nombre="Electricos"/>
  <ClasesdeArticulos Nombre="Ebanisteria"/>
</ClasesdeArticulos>

<Articulos>
  <ArticuloCodigo="34A67E" Nombre="Tachuela" ClasedeArticulo="Ebanisteria"
Precio="100.00"/>
  <ArticuloCodigo="45X89E" Nombre="Bombillo" ClasedeArticulo="Electricos"
Precio="350.00"/>
  <ArticuloCodigo="93P97X" Nombre="Tubo PVC" ClasedeArticulo="Plomeria"
Precio="500.20"/>
</Articulos>

```

En este XML, el mapeo entre las Articulo y ClaseArticulo es por medio del nombre, sin embargo, en la base de datos física, el mapeo debe hacerse por id, o sea mediante FK que refiere al id.

**Siete.** ¿Qué se pide?

El código en capa lógica para el funcionamiento de las páginas.

La base de datos creada, las tablas con información cargada y los store procedures para realizar las funcionalidades solicitadas.

Un script con los estatutos en MSSQL para cargar los datos de prueba.

La documentación que son dos documentos, una bitácora y un análisis de resultados.

## La bitácora

Debe ser escrita en una herramienta para hacer blogs, por ejemplo [www.blogger.com](http://www.blogger.com), para cada entrada se indica la cantidad de horas trabajadas por el equipo de trabajo durante una sesión de trabajo, se hace un relato de los avances acerca de los problemas encontrados, como fueron resueltos; dudas, divergencias de criterio, forma en que trabajó el equipo de trabajo, problemas con la instalación del software, problemas de aprendizaje del framework, investigaciones, pruebas de concepto, experiencias, moralejas, ayuda recibida, consejos a dar, buenas prácticas descubiertas, incluir preguntas que se hagan al profe en el foro o en comunicaciones privadas, etc.; La entrada en el blog describe el proceso de solución de la tarea programada; una descripción sincera y detallada será bien evaluada. Incluya referencias externas a recursos utilizados en internet para solventar dudas o resolver problemas, debe incluir los mensajes de error y como fueron resueltos.

Debe escribir las entradas en la bitácora teniendo en mente que serán la prueba de que el equipo de trabajo, laboro regularmente (o sea que no hizo la tarea la noche previa), que muestre el proceso de aprendizaje (paulatino e incremental), en otras palabras, que sea evidencia de que la tarea NO fue copiada, y de que el equipo de trabajo ha trabajado desde que se entregó la especificación y lo hecho regularmente.

## **Análisis de resultados**

Es un documento word, que debe ser formal o sea profesionalmente bien presentado, su objetivo es indicar que funciona o no, de los requerimientos y mostrar las métricas del proyecto.

El documento debe tener la siguiente estructura:

- Portada,
- índice de contenido,
- introducción,
- descripción del ambiente de desarrollo,
- un análisis de resultados que es una tabla donde para cada elemento que se evalúa de la tarea, según la plantilla de evaluación, se muestra el nombre del elemento (requerimiento, artefacto de diseño, etc), una valoración de si está implementado exitosamente, o si no un % de implementación, y si no es un 100% un comentario que explique que falta para que ese elemento esté un 100%
- Métricas del proyecto: Luego a través de una tabla se indican las métricas del proyecto, por ejemplo: horas trabajadas, cantidad de líneas de código, cantidad de entradas en el git hub, cantidad de datos de pruebas procesados, cantidad de pruebas realizadas, tiempo de duración de las pruebas, cantidad de tablas creadas, cantidad de procedimientos almacenados, etc., etc., todas las métricas que se puedan derivar del proyecto. Son obligatorias: cantidad de entradas en la bitácora, cantidad de entrada en el git, fecha de la primera entrada en la bitácora y fecha de la primera entrada en el git.

Se pueden agregar gráficos que ofrezca el github, o cualquier elemento que mejore la calidad del documento.

Toda tabla o gráfico debe tener una referencia textual. Por ejemplo, referido al Análisis de resultados: “En la siguiente tabla se indica el porcentaje de implementación de cada uno de los elementos de evaluación del proyecto y se comenta su estado final.”

El uso del github es obligatorio, será una prueba de trabajo constante y colaborativo.

**Ocho.** Reglas.

Todo el código de programación referido a base de datos, debe ser un procedimiento almacenado. No puede haber SQL incrustado en capa lógica, lo único permitido desde capa lógica es invocar un procedimiento almacenado. Todas las validaciones, excepto aquellas que tienen que ver con validar valores monetarios, deben hacerse en los store procedures.

Grupos de 2 personas. Motor de base de datos: MS SQL cualquier versión superior a 2014. Código en capa lógica, en el lenguaje o framework de su preferencia.

El valor de esta tarea programada es 13.3% de la nota.

Fecha de entrega: Cercana al lunes 25 de Setiembre.