Xabier San Martín

Repositorio:

* <https://github.com/xabiersanmartin/ProyectoTest>
* ***Comentarios en verde para indicar que está arreglado. Pero mira la lista de tareas con comentarios TODO***

Entrega inicial de BD

* **Preguntas**: No tiene clave. El orden de los campos sería más lógico: IdTest, Npregunta (así la consulta agrupará mejor las preguntas de un test)
* Sube los cambios ya en el repositorio, incluyendo los datos (recuerda que pueden ser 2 archivos sql diferentes, uno como el que has subido, y otro, que solo se encargue de insertar datos en las tablas.
* Curiosamente las relaciones que tienen que ver con testcategorias son en cascada (hablaremos de esto también en la videoconferecia) pero de Test a preguntas no.

Comentarios 23-04.

* Los nombres de las entidades son en singular: Categorias suena a varias y solo define una categoría.
* Las entidades no están pensadas como tal, sino que defines la tabla de la BD, pero eso no es entidad.

Por ejemplo: ¿qué es un Test? Pues un objeto que tiene un identificador, una descripción, una categoría (o al menos el identificador de la categoría) y varias preguntas.

Revisa por tanto las entidades.

* Que los métodos **devuelvan 0, 1, -1** es muy difícil de mantener, o se devuelven valores boolean relacionados con el nombre del método en cuestión: por ejemplo, ExisteTest puede responder true/false. Si necesitas más valores, o te inventas un tipo enum con nombres que lo clarifiquen, o un mensaje, o…
* Siempre debe llegar al usuario/a final la máxima información posible de los errores, y en tu caso, pudiendo tener bastante, la pierdes en el camino.

Ejemplo: los catch

catch (Exception)

return "No se a podido conectar con la base de datos"; // Controla la ortografía **a podido**

* Sigues pensando desde la capa de presentación hacia la de datos 🡪 ¿en un método AnadirCategoria de la capa de datos qué sentido tiene este mensaje: Como puedes ver en el recuadro de al lado, la categoria ya existe ¿De qué cuadro está hablando? Estabas haciendo referencia al lugar concreto del formulario en el que usas este método, pero ¿no puede servir luego para otros lugares?
* Si estás haciendo de forma que, por ejemplo, el nombre de la categoría no pueda repetirse, ¿qué significa este while cuando vas a buscarla? while (dataReader.Read()) ¿Puede haber varias? Y ya es más descarado cuando la búsqueda la haces por Id
* A la hora de borrar una categoría compruebas que tenga o no Test para sacar el posible mensaje (bien) pero luego si te dice que te permite borrar la categoría no compruebas que los test pueden tener preguntas (y en este caso encima sin borrado en Cascada, por lo que, si esto ocurriese, crearías un conflicto. Piensa y comenta la solución a este tema, puede haber varias diferentes
* Procura que los formularios tengan un botón de vuelta al principal.
* Añadir Test no tiene lógica como lo haces.

Lo lógico en este caso es permitir que teclee un nombre de test y varias categorías (que ha podido ir eligiendo de un combo de categorías). Cuando haya introducido todos los datos, un botón de Añadir será el encargado de crear el nuevo test (con lo que eso significa en diferentes tablas).

Y en ningún caso aporta información ver todos los test.

* Borrar Test, en cambio sí debería comenzar eligiendo la categoría. Al elegirla que apareciesen los test de esa categoría y cuando seleccionemos un test de esos, permitir el borrado.

Recuerda además que puede tener preguntas (que encima están sin borrar en cascada).

Cuando hayas arreglado estos errores me envías un correo para que los revise

Comentarios pendientes de entrega anterior

Gestor

* Siempre debe llegar al usuario/a final la máxima información posible de los errores, y en tu caso, pudiendo tener bastante, la pierdes en el camino. Observa que si declaras la variable de excepción y devuelves también el exc.Message como parte del mensaje, la información es mayor, ya que puede que nos diga por qué no ha podido conectar. Prácticamente siempre debemos recoger la variable Exception.

Ejemplo: los catch

catch (Exception)

return "No se ha podido conectar con la base de datos";

* ¿Para qué necesitas un Initialize y un OpenConnection que prácticamente hacen lo mismo?
* Si estás haciendo de forma que, por ejemplo, en ExisteCategoria el nombre de la categoría no pueda repetirse, ¿qué significa este while cuando vas a buscarla? while (dataReader.Read()) ¿Puede haber varias? Y ya es más descarado cuando la búsqueda la haces por Id, por ejemplo, en EliminarCategoria

28 de abril

* Debes cuidar la ortografía: se a eliminado, Piensa que tambien eliminaras, categorías, No se elimino la categoria
* Procura que siempre los nombres y valores devueltos sean coherentes. ExisteCategoria ¿cuándo debería devolver true con este nombre, cuando existe o cuando no? Respóndete a la pregunta y luego mira lo que hace.
* Cmd.ExecuteNonQuery devuelve un valor (el número de filas afectadas) que nunca recoges en una variable y que se debe hacer para ver si realmente se ha ejecutado o no. Ejemplo, si es un alta, luego preguntaré si realmente ha habido 1 fila añadida (o al menos si el valor devuelto no es 0).

Formularios

FrmAnadirTest

* No tiene sentido la forma en que se plantea. Se trata de añadir un test y a este asignarle **las** categorías a las que va a pertenecer, no que al crearlo lo hagamos sin categorías, y luego para añadirlas tenga que elegirlo varias veces y me pueda saltar a otro, … Un follón. Y encima, en ningún momento estoy viendo las categorías del test…

FrmBorrarTest

* Funciona, pero de nuevo con mucho código repetido: Llama a EliminarTest y si detecta que tiene preguntas luego tiene que llamar a EliminarTestConPreguntas que parte del código es común, …

FrmPrincipalPreguntas y Frm

* Bastante bien, ver algún comentario (incluyendo FrmPrincipalPreguntas)

Entidades

* No son entidades realmente lógicas, se parecen más a lo que son las tablas, por ejemplo, si en un test tienes las preguntas ¿para qué querrías tener en pregunta el idTest? En bd tiene sentido, pero como entidad, se supone que nunca necesitaremos trabajar con una pregunta como tal, sino dentro de un test, y ahí ya conocemos su id si queremos luego grabar la pregunta. *Si no entiendes esto, me lo preguntas en videoconferencia*
* ¿Por qué Categoria tiene una lista global (no es propiedad, y debería serlo) que se llama testCategorias (no debe comenzar en minúscula) y que en realidad es una lista de Test? Y curiosamente no tiene constructor para ella.

Gestor

* Los métodos OpenConnection y CloseConnection modifican una variable error que no sirve para nada. Aunque en ese momento se pueda saber el motivo de porqué no se puede conectar, se pierde ahí, luego las funciones que la usan solo devuelven true o false o cosas semejantes.
* ¿Para qué el constructor llama a Initialize? ¿qué sentido tiene si lo único que hace es intentar abrir y cerrar la conexión? Y encima, si no puede, solo pierde tiempo intentándolo, pero luego nadie se entera de que no ha podido. Vaya que no tiene ningún sentido ese Initizalize.
* No usas parámetros en las consultas (cmd.Parameters)

Así no cometerías el error de considerarlo de tipos erróneos, como haces en este, que IdCategoria es int (en la BD) y tú lo pones como String en la consulta. Dependiendo del valor que tuviese podría no funcionar la consulta

string comprobarTest = "SELECT \* FROM CATEGORIASTESTS WHERE (((IdCategoria) = '" + categoriaEliminar.idCategoria + "'))"; // No se deben hacer las consultas así (con + ) sino con parámetros. Así son más fáciles de hackear (está en los apuntes)

List<Categoria> listCat = new List<Categoria>(); // TODO Esto es copia de otros lugares, pero en esta función no tiene sentido

while (dataReader.Read()) // ¿pueden ser varias?

ExisteCategoria

* No tiene sentido el código, con aparente lista de categorías, ExecuteReader, while, .. solo para saber si existe o no una categoría

private bool ExisteCategoria(string consulta) // TODO Si solo necesita saber si exiswte o no, ¿para qué usar una consulta de tipo ExecuteReader y si solo puede haber 1, ¿por qué while? Y ¿que sentido tiene este parámetro? Lo lógico es que se haga dentro la sql

EliminarCategoria

* No tiene sentido que reciba toda la categoría, con el id debería bastar

public String EliminarCategoria(Categoria categoriaEliminar)

* Si lo único que quieres saber es la cantidad de test que tiene, para qué necesitas hacer esa Select y recorrer todas sus CATEGORIASTEST. Recuerda BD, con un Count te debería bastar (esta consulta es mucho más rápida y lógica)
* Si lo que quieres saber es si tiene o no test, ¿para qué necesita recorrerlos todos? Bastaría con una consulta Scalar (Count) o sino con HasRows, pero el while que haces carece de sentido
* Ver en while la instrucción: result = (dataReader["IdCategoria"].ToString()); // ¿?¿?¿? Es una lógica muy extraña

ModificarCategoria

* Cantidad de código idéntico al de EliminarCategoria. Como siempre comento: no se debe copiar código, si algo se repite habrá que ver la forma de que no sea así.

DevolverTestAsociadoCategoria

* Los test se devuelven con el idCategoria sin dato.
* ¿Para qué necesitas dos listas testAsoc y testDevolver?

DevolverCategorias

* public List<Categoria> DevolverCategorias() // TODO Si hay errores de ejecución, no salen al exterior

ara que los revise