

# Devolución 2da entrega: Heart Bits

Prof. Ismael Schellemborg

## Observaciones generales:

- ✓ ● La capa presentación es muy atractiva. Se destaca:
- ✓ ○ Uso de fuentes, colores de texto, imágenes, fondos.
- ✓ ○ Pantalla de carga.
- ✓ ■ Si bien no está cargando nada, la pantalla está interesante y el usuario no se dará cuenta.
- ✓ ○ Buena incorporación detectar el Enter para realizar la acción de Login.
- ✓ ● Excelente uso de herencia en clases de datos.
- ✓ ● Se nota y valora el esfuerzo en el pulido de la aplicación.

## Correcciones necesarias (requerimientos mínimos):

- Separar presentación de capa lógica
- ✓ ○ Algunos formularios utilizan ConnectionDB directamente, cuando deberían comunicarse únicamente con la capa lógica.
- ✓ ○ Si se necesita cargar un DataGridView, la persistencia puede devolver un DataSet o DataTable, el cual puede ser cargado en un data grid sin necesidad de que la presentación maneje un Recordset.
- ✓ ○ Manejar errores con exceptions.
- ✓ ■ La lógica es la que debe validar los campos y arrojar exceptions en caso de detectar un error.
- ✓ ■ La presentación también puede arrojar exceptions en caso que haya campos inválidos (y ella misma las atrapa).
- ✓ ■ Ejemplos de validaciones pertinentes a la presentación:
- ✓ ■ Letras en un campo numérico
- ✓ ■ No se seleccionó si es tratamiento de medicamento o quirúrgico.
- Ejemplos de validaciones pertinentes a la lógica:
- ✓ ■ Textos vacíos.
- ✓ ■ Existencia de al menos 1 tratamiento
- ✓ ■ Ejemplo en FrmLogin.BtnLogin\_Click, la presentación debería únicamente solicitar a la lógica que realice una acción de login, y atrapar las exceptions producidas con un try catch. En caso de ser exitosa mostrar el Home.
- ✓ ● Mejoras al obtener diagnóstico.
- ✓ ○ La aplicación crasha al intentar obtener diagnóstico. El error se produce en la consulta de obtener patologías.

- ✓
  - COMException: '[MySQL][ODBC 8.0(w) Driver][mysqld 5.7.31]Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'telediagnosticomedico\_heartbits.p.id' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql\_mode=only\_full\_group\_by'
- ✓
  - La consulta hace un GROUP BY por ps.id\_sintoma sin embargo no puede agrupar las patologías por síntoma.
- ✓
  - Las patologías ya están siendo seleccionadas una única vez gracias al DISTINCT, ¿es necesario el GROUP BY?
- ✓
  - Menciona un tema de configuración de base de datos sql\_mode, es posible que funcione con la configuración de su PC y no en la mía.
- ✓
  - Se arregla eliminando el GROUP BY.
    - Al seleccionar varios síntomas, aparecen muchos diagnósticos, todos con evidencia baja o media, alguno quizás con evidencia alta.
    - Es difícil para el usuario identificar cual es el diagnóstico más certero para lo que ingresó.
    - ¿Qué significa que haya evidencia alta o baja?
      - Uno podría entender que depende de la cantidad de síntomas presentes, sin embargo no parece funcionar así.
    - Realicé la siguiente prueba:
      - Seleccioné todos los síntomas de Bronquitis (Disnea, Tos, Espujo, Fatiga, Fiebre, Escalofríos, Dolor torácico).
      - Aparecen 27 resultados posibles.
        - 8 resultados con evidencia media (incluyendo Bronquitis)
        - 2 resultados con evidencia alta (Linfoma, Rabia)
          - Linfoma comparte síntomas Disnea, Fatiga, Fiebre
          - Rabia solo comparte el síntoma Fiebre
        - El resto evidencia baja.
        - La primer patología que aparece (que es lo primero que leerá el paciente) es Nefropatía crónica, la cual comparte varios síntomas con Bronquitis pero tiene diferencias.
        - Bronquitis es el sexto resultado.
      - ¿Habiendo seleccionado exactamente todos los síntomas de Bronquitis, cómo puede el usuario identificar entre todos los resultados qué diagnóstico es el más probable de ser acertado?
    - Este tipo de operación es muy compleja para resolverla con una única consulta SQL.
    - Sugerencia: La lógica debería obtener todas las patologías posibles como lo está haciendo ahora, pero en lugar de devolverlas a la presentación automáticamente, debería determinar para cada una qué tan probable es que tenga esa enfermedad.
      - Ejemplo Rabia, si solo tengo Fiebre, ¿qué tan probable es que tenga Rabia? La probabilidad es baja.

- Para cada patología encontrada, se podría asignar un número de probabilidad dependiendo de qué tantos síntomas presenta el usuario.
  - En el ejemplo anterior, el paciente tendría 20% de tener Rabia (porque solo hay 1 síntoma presente de los 5 que tiene Rabia), pero 100% de tener Bronquitis, porque presenta el 100% de los síntomas.
    - Quizás no sea necesario mostrar al usuario los porcentajes (¿o quizás sí?) pero al menos el sistema podría destacar o poner más arriba los más probables, con algún tipo de indicador “esto es más probable”, dejando las opciones poco probables más abajo.
    - Si hay varios resultados con el mismo porcentaje, quizás tomar el que tenga mayor cantidad de síntomas presentes.
  - Si se les ocurre otra forma de reducir la cantidad de resultados o al menos poder indicar cuales son los más probables también será válida.
- ✓ ● ~~No se están atrapando las exceptions correctamente.~~
- ✓ ○ ~~En el formulario de login si se intenta hacer login con la base de datos sin funcionar no sucede nada ni se muestra un diálogo de error.~~
- ✓ ○ ~~Podrían producirse otras exceptions que no se estarían atrapando.~~
- ✓ ● ~~Corregir acceso a lógica desde FrmAlertRemove.~~
- ✓ ○ ~~Si necesita acceder a la lógica debería acceder directo al controlador sin crear un formulario solo para llamar a la función que este contiene.~~
- ✓ ○ ~~Si FrmPath no utiliza la función de borrar, ¿por qué está ahí? Moverla a FrmAlertRemove.~~
- ✓ ● ~~Corregir error al eliminar una patología: se probó eliminando Cataratas~~
- ✓ ○ ~~Primero muestra mensaje de éxito.~~
- ✓ ○ ~~Luego la operación de eliminar la patología falla~~
- ✓ ■ ~~COMException: '[MySQL][ODBC 8.0(w) Driver][mysqld-5.7.31]Cannot delete or update a parent row: a foreign key constraint fails ('telediagnosticomedico\_heartbits`.`sintoma\_compone`, CONSTRAINT `sintoma\_compone\_ibfk\_2` FOREIGN KEY (`id\_patologia`) REFERENCES `patologia` (`id`))'~~
- ✓ ■ ~~Error por clave foránea, la patología está asociada a síntomas (y a tratamientos) por lo que no puede ser eliminada hasta que se eliminen las asociaciones.~~
- ✓ ○ ~~La persistencia debería eliminar primero las asociaciones de síntomas y los tratamientos de esa patología, y por último eliminar la patología en sí.~~
- ✓ ○ ~~El mensaje de éxito debe mostrarse luego de que todas las operaciones hayan sido completadas exitosamente y nunca antes (porque pueden fallar).~~

## Correcciones opcionales:

- ✓ ● Utilizar gris más oscuro para la letra en algunos lugares.
- ✓ ○ Utilizar distintos colores ayuda a organizar mejor el contenido, sin embargo un gris muy claro con fondo blanco es difícil de leer, especialmente para personas con problemas de visión.
- ✓ ○ Notorio en los siguientes lugares:
  - ✓ ■ Primeros 2 formularios en la aplicación de pacientes.
  - ✓ ■ Sección "Sobre el sistema"
  - ✓ ■ Sección "Acerca del administrador"
- ✓ ● Agregar botón de cerrar en la esquina superior derecha.
- ✓ ○ No todos los usuarios conocen alt+f4 u otras técnicas de cerrar una aplicación, especialmente la población mayor.
- ✓ ● Mejoras al selector de síntomas para obtener diagnóstico:
  - ✓ ○ Agrandar el espacio de selección de síntomas (en el ComboBox).
  - ✓ ■ El usuario debe encontrar sus síntomas de una lista enorme y solo puede ver de a 3. Esto dificulta mucho encontrar el síntoma apropiado.
  - ✓ ○ Si se presiona una tecla para buscar, se selecciona automáticamente el primer síntoma con esa letra.
  - ✓ ■ Presionar la primer letra de lo que buscamos es una acción común para buscar en un ComboBox con muchos elementos.
  - ✓ ■ No debería seleccionar elementos al usar el control de esta forma, especialmente considerando que la lista de síntomas seleccionados queda escondida y no veo que se agregó algo accidentalmente.
- ✓ ● Mejoras en formulario de administración de síntomas:
  - ✓ ○ Reorganizar la parte superior del formulario de listado de síntomas.
  - ✓ ■ Los controles de alta síntoma están sueltos a la derecha y sin un título que identifique esa sección del formulario. Al abrir la pantalla no queda claro para qué están estos controles ya que el título parece ser "Asociación a patologías".
  - ✓ ■ El alta de síntomas podría hacerse en el mismo formulario que la modificación de síntomas.
  - ✓ ○ Pueden diferenciarlos utilizando un objeto Symptom con Id= -1 y inicializado con todos los campos vacíos.
  - ✓ ○ Se eliminarían los controles de alta en el formulario de listado.
  - ✓ ○ El botón nuevo abre el formulario de alta/modificación en lugar de hacer el insert.
  - ✓ ■ Barra de búsqueda separada del listado.
  - ✓ ■ Al haber mucho espacio entre los controles (y considerando los controles de alta síntoma a la derecha) no queda claro que está relacionada a la tabla.

- ✓
  - Si mueven el alta síntoma al formulario de modificar, puede agrandarse la tabla para que abarque toda la pantalla y a la vez solucione este problema.
- ✓
  - Agregar botón de modificar síntoma.
- ✓
  - No es tan evidente que doble click sirve para editar (si bien es una excelente incorporación a la experiencia de usuario).
- ✓
  - Si necesitan espacio pueden eliminar la palabra “síntoma” de los botones ya que está implícita en el formulario de síntomas.
- ✓
  - Agrupar las 3 acciones de síntomas sin que esté el botón refrescar en el medio.
- ✓
  - Eliminar complejidad excesiva de operación de modificar síntoma.
- ✓
  - La presentación debería trabajar con un objeto Síntoma el cual envía a la lógica para guardar.
- ✓
  - El objeto síntoma es cargado desde el listado de síntomas al abrir el formulario (puede ser un parámetro del constructor del formulario).
- ✓
  - Dado que trabajan con el DataGridView, sería necesario una operación BuscarSíntoma en la cual se envía el ID y se obtiene el objeto Symptom con toda la información cargada.
- ✓
  - No debería importar qué campos fueron cambiados (incluso nombre) dado que identificamos al síntoma por su campo ID.
- ✓
  - Trabajando con el objeto Síntoma, siempre sabremos el ID de lo que estamos modificando por lo que no es necesario que la capa persistencia realice una consulta para averiguarlo.
- ✓
  - Lo mismo aplica a las regiones.
- ✓
  - No necesita recordar los valores anteriores de los campos ni diferenciar casos según qué propiedades fueron modificadas.
- ✓
  - Si síntoma tuviera 15 campos de datos sería imposible contemplar todas las combinaciones posibles.
- ✓
  - Debe realizar un único UPDATE donde actualice todos los campos (los modificados y los no modificados) identificando el objeto a modificar únicamente por su ID.
- ✓
  - Mejoras en formulario de administración de patologías:
    - Que el listado abarque toda la pantalla.
- ✓
  - Al tener tantas cosas para mostrar, conviene aprovechar al máximo el espacio del formulario y no dejar espacios vacíos innecesariamente.
- ✓
  - Evaluar necesidad de tener 2 barras de título “criterios de búsqueda” y “listado patologías”.
- ✓
  - Agregar botón de modificar patología.
- ✓
  - Aplican los mismos comentarios que para el botón de modificar síntoma
- ✓
  - Agregar y modificar patologías
- ✓
  - Evaluar la posibilidad de que agregar y modificar patologías utilicen el mismo formulario, dado que trabajan sobre la misma entidad y realizan las mismas validaciones, por lo que repiten mucho código entre ellas.

- ✓
  - Cambiar la forma de utilizar el DataGrid de tratamientos.
- ✓
  - En lugar de agregar tratamientos directamente al DataGrid y acceder a las celdas del DataGrid buscando información, los tratamientos deberían guardarse en una List (Of Treatment) y luego el DataGrid cargarse a partir de esa lista.
- ✓
  - No acceder a la información directo a la tabla. En lugar de eso buscar el objeto Treatment en la lista.
- ✓
  - El código quedará más sencillo y prolijo.
- ✓ ● Las propiedades de People y el enumerador Genre deberían ser públicas en lugar de protected.
- ✓
  - Al ser protected, solo pueden ser utilizadas por las clases que hereden de People.
- ✓
  - Eso implica que desde ningún formulario podremos acceder a los datos de una persona (por ejemplo su nombre) para mostrarlos en pantalla.
- ✓
  - Permitiría cargar la información en el panel “acerca del administrador”.
- ✓ ● Evitar diálogos de error todo en mayúsculas o con múltiples símbolos de exclamación.
- ✓
  - Utilizar mayúsculas solo en la primer palabra o en nombres para facilitar lectura.
- ✓ ● Eliminar funciones vacías
- ✓
  - En algunos formularios hay eventos configurados que no hacen nada. Si no son necesarios, eliminarlos.