Documento de arquitectura

Sistema del complejo de cines CINETOP

Índice de contenido

[Decisiones referidas a la base de datos y al trabajo en equipo: 3](#_Toc55152555)

[Método por el cuál compartimos la estructura y contenido de la base de datos del sistema entre los desarrolladores: 3](#_Toc55152556)

[Método que se utilizó para compartir el código fuente y recursos del proyecto: 3](#_Toc55152557)

[Base de datos utilizada y host: 3](#_Toc55152558)

[Cómo se distribuyeron los roles en el grupo: 3](#_Toc55152559)

[Decisiones referidas al proyecto en sí: 4](#_Toc55152560)

[Acuerdos de interfaz: 4](#_Toc55152561)

[Acuerdos de programación: 5](#_Toc55152562)

[Problemas y su solución: 5](#_Toc55152563)

[Aprendizajes: 7](#_Toc55152564)

Decisiones referidas a la base de datos y al trabajo en equipo:

## Método por el cuál compartimos la estructura y contenido de la base de datos del sistema entre los desarrolladores:

La base de datos fue compartida y la consistencia entre ambas partes mantenida a través de archivos .sql con los scripts que realizaban los cambios requeridos a medida que crecía el proyecto. Estos archivos fueron compartidos por la vía de comunicación que fuera conveniente en el momento.

## Método que se utilizó para compartir el código fuente y recursos del proyecto:

Para esto se hizo uso de un repositorio GIT con host en github.com, lo que facilitó el trabajo remoto y la división de tareas.

## Base de datos utilizada y host:

Se hizo uso del DBMS de Microsoft SQL Server.

Se decidió que ambos integrantes utilizaran un host local para la base de datos. Esto no resultó un problema al coordinar bien la distribución de los cambios del código junto con la de los cambios en la estructura de la base de datos.

## Cómo se distribuyeron los roles en el grupo:

Benjamín asumió el rol de administrador de la base de datos, programador de la aplicación (front-end y back-end) y también fue el encargado de mantener un estilo visual común entre todos los formularios del sistema. Además, se encargó de la confección de este documento.

Agustín se desenvolvió como programador de la aplicación (front-end y back-end), tester y se encargó de revisar la coherencia, claridad y consistencia de mensajes de error y demás información presentada al usuario. Además, se encargó de la confección del manual de usuario.

Decisiones referidas al proyecto en sí:

## Acuerdos de interfaz:

* Todos los formularios poseen el mismo color de fondo gris (“ControlDark”).
* Los botones y opciones de la barra de menú possen bordes negros, fondo gris oscuro (“ControlDarkDark”) y letra con una fuente de color claro (“ControlLight”).
* Cada formulario que poseyera algún tipo de opción de filtrado lo mostrará en la misma ventana, en la parte superior de este.
* Los reportes y grillas aparecerán en la parte inferior de la ventana de los formularios que los posean.
* Todos los formularios “acaparan” el control al ser abiertos desde el menú principal, no permitiendo así que se puedan abrir 2 formularios a la vez.
* No hay convención respecto del tamaño de los formularios, estos pueden ser del tamaño que requieran por el motivo que fuese.
* En los formularios de ABMC siempre se mostrarán los mismos botones en la parte inferior, con los mismos iconos y funcionalidades, y agrupados de la misma manera:
  + Grupo 1: Nuevo registro, modificar registro seleccionado y eliminar registro
  + Grupo 2: Cancelar operación y guardar cambios.
  + Grupo 3: Salir del formulario.
* Todos los formularios de ABMC presentan una grilla donde se muestran todos los registros disponibles en la base de datos.
* En los campos donde se pueda seleccionar más de una opción disponible y sin límite de cantidad se hace uso de un combobox para mostrar las opciones, una grilla para almacenar las seleccionadas y 2 botones (+ y -), uno para agregarlas a la grilla y otro para eliminar lo seleccionado en esta.
* Todos los formularios poseen el logo de la empresa como icono y un título descriptivo.
* Cada vez que ocurriera una validación errónea o un error referente a la conexión con la base de datos se informa al usuario mediante un mensaje en una ventana emergente.
* Todos los reportes e informes deben incluir el logo de la empresa y la fecha de su generación en las esquinas superiores.

## Acuerdos de programación:

* Acciones como la carga de grillas y combos con datos, así como la adaptación del formato de estos en las grillas, son manejados haciendo uso de una clase con métodos auxiliares (FrmHelper).
* Se trabajó con una programación en capas, con la capa de GUI agrupando formularios, la capa de datos agrupando clases que trabajan directamente con la base de datos haciendo uso de sentencias SQL, la capa de negocio con clases intermedias que implementan algo de la lógica de negocio y hacen de nexo entre capa de datos y capa de GUI.
* Las entidades de negocio se agruparon en un fichero aparte para facilitar su acceso generalizado en todas las capas.
* Se uso un mismo objeto DataSet para todos los reportes y estadísticas.
* Para el acceso a la base de datos se hizo una clase adaptadora (DBHelper) que encapsula funcionalidades como la conexión y desconexión con el host, ejecución de consultas y manejo de transacciones.
* Validaciones muy grandes y/o repetidas en los formularios se agrupan en un método privado aparte para mayor reutilización y legibilidad.
* Los archivos .rldc de los reportes y estadísticas se agruparon en un fichero aparte.

## Problemas y su solución:

En un par de casos de uso necesitamos hacer usos de campos que tuvieran que permitir la selección de varios datos predefinidos (géneros, directores, actores y premios). La solución fue hacer uso de una grilla, la cual puede llenarse y vaciarse mediante un combo y botones de agregar y quitar filas, que al momento de asignar sus datos a un objeto se traduce a un array con un elemento por fila (sus id, encontrados en una columna no visible al usuario, u objetos construidos con los datos disponibles en ellas).

Otro problema fue que en un principio las consultas a la base de datos eran atómicas y autoconclusivas, al hacer una consulta o ejecución de sentencia el mismo método del DBHelper cerraba la conexión con la BD. Esto complicaba mucho el manejo de transacciones, ya que la conexión debía mantenerse a lo largo de varios métodos. La solución fue simplemente quitar esa desconexión de los métodos y delegar la responsabilidad de realizarla a las clases DAO que hacen uso de la DBHelper.

Un par de problemas aún más complejos surgieron al momento de manejar la transacción de registrar películas.

Uno de ellos consistía en que el caso de uso, además de permitir agregar actores ya registrados, permitía ingresar nuevos actores (escribiendo directamente su nombre y apellido en vez de seleccionarlos), lo que implicaba que debía verificarse para que actores debía crearse una nueva fila en la tabla Actores de la BD a la hora de grabar la transacción. La solución a esto fue que al momento de crear un array de objetos actor a partir de la grilla aquellos que fueran nuevos se les asignara un id con un valor ilegal (0), así luego al momento de grabar cada uno de los elementos del array en la BD habría un condicional if que verificara ese valor, creará un nuevo registro para aquellos cuyo valor fuera 0 y asignara el nuevo id autogenerado al objeto en sí, pudiendo luego insertarse en la tabla ActoresXPelicula como el resto.

El otro surgió de la necesidad de poder modificar las películas ya guardadas, que implicaba tener que verificar que datos se agregaron, quitaron y/o se mantuvieron en los campos que aceptaban múltiples entradas al momento de tener que grabarlos en las tablas que correspondían a cada uno en la BD (ActoresXPelicula, GenerosXPelicula, PremiosXPelicula). Esto pudo solucionarse agregando una consulta previa a la inserción de estos datos, que obtendría los id y valor de borrado de los registros de la tabla con la que se quisiera trabajar y que tuvieran el id de la película que se quería modificar. Así, estos valores quedarían contenidos en un array para que antes de insertar un nuevo registro primero se verificara el id ya estaba contenido en este, si fuera el caso y el valor de borrado fuera 0 el objeto no se registraría, y si el valor de borrado fuera 1 simplemente se haría un UPDATE cambiando ese valor a 0. Cada vez que un id validara como contenido en el array, el elemento que correspondiera a este se quitaría y, al final de las inserciones, se realizaría una sentencia UPDATE para cada elemento restante cuyo valor de borrado fuera 0 para cambiarlo a 1, pues significaría que el id de ese dato fue quitado de la película por el usuario desde el formulario.

Aprendizajes:

De este proyecto hemos sacado unos cuantos aprendizajes valiosos tanto a nivel personal como profesional.

Benjamín tuvo su primera experiencia desarrollando software con un grupo colaborativo y de manera remota, adaptándose a los tiempos y capacidades de cada uno, trabajando de manera síncrona o asíncrona según se requiriese. También tuvo la oportunidad de aplicar los conocimientos aprendidos en la materia Gestión de Datos al desarrollo de un software de escritorio.

Benjamín también pudo profundizar en algunos de los aprendizajes adquiridos cuando trató de aprender a desarrollar software con interfaces gráficas en java por su cuenta, llegando a tener una mayor compresión sobre el tema y quedando satisfecho con ello.

Agustín también tuvo su primera experiencia de desarrollo en grupo, teniendo aportes de todas las partes y aprendiendo a buscar y expandir los conocimientos a través de herramientas de búsqueda y foros.

Ambos integrantes utilizamos la herramienta de git por primera vez y hemos adquirido un conocimiento básico de esta y su funcionamiento.

Ambos nos familiarizamos un poco con el entorno de .net y al desarrollo con winforms, como también al uso del IDE de Visual Studio.

Ambos integrantes tuvimos una buena oportunidad de aprender y usar el lenguaje C#, aplicando el paradigma orientado a objetos y aprendiendo a través de las clases, la práctica y material obtenido en la web.

Logramos aplicar una aproximación al desarrollo dividido en capas y un ejemplo de aplicación de un patrón de diseño, ambos temas concernientes a la materia Diseño de Sistemas.

En general hemos podido aplicar contenidos de diversas materias, así como contenido nuevo y aprendizajes adquiridos en el pasado de manera individual, a un proyecto medianamente complejo de software de forma satisfactoria y exitosa.