Universidad Tecnológica Nacional – Facultad Regional Córdoba

Cátedra de Programación de Aplicaciones Visuales I

Curso: 3K1

Documento de Arquitectura

Proyecto: LPC Facturas

Integrantes Grupo N° 2

Nombre y Apellido	Legajo	Correo electrónico
Joaquin Costamagna	80383	jncostamagna@gmail.com
Agustín Maubecín	79637	maubecinagustin@gmail.com
Gastón Muñoz	80384	gfranciscom99@gmail.com
Jano Paschetti	80195	janopaschetti@gmail.com

Docentes

- Romero, María Soledad
- Figueroa, Rodolfo Alfredo
- Botta, Oscar Ernesto

Fecha de Entrega 13/11/2020

Índice

Introducción 2
Base de Datos 3
Programación y estructuras de código 5
Interfaces del proyecto 6
Aprendizajes 8
Problemas encontrados a lo largo del proyecto 10
Distribución de roles en el grupo 12
Conclusión 13

Introducción

El propósito de este documento es poder definir y explicar las distintas decisiones arquitectónicas realizadas a lo largo del proyecto, respecto a las interfaces utilizadas, los alcances del proyecto, la forma de trabajo compartida, las resoluciones de los problemas presentes a lo largo de la implementación, el manejo de base de datos, y demás.

Base de Datos

¿Cómo compartimos base de datos?

En esta situación, se nos presentaron dos opciones para poder realizar el manejo de datos en forma compartida:

1) Cada integrante del grupo utiliza su propia base de datos, generada a base del mismo script.
2) Generar una única base de datos, y gestionar el acceso a la misma para que cada integrante pueda acceder a la misma simultáneamente.

En un principio, habíamos pensado en la segunda alternativa, ya que esto permite evitar la redundancia innecesaria de la base de datos, además de que, si algún miembro del grupo generaba una nueva entrada en alguna de las tablas de la base de datos a través de los ABMs del proyecto, esta nueva entrada sería visible para todos los integrantes del proyecto.

Sin embargo, encontramos que las opciones posibles para poder trabajar de este modo nos limitaban a tener que pagar servidores remotos o en su defecto utilizar motores de bases de datos de capacidad de procesamiento limitada, lo cual demoraba las tareas del día a día. En base a esto, decidimos seguir la primera opción, y generar cada uno en su estación de trabajo su propia base de datos.

El inconveniente de esta alternativa, aparte de no generar los mismos datos al utilizar los ABMs de la aplicación, era el manejo de la conexión de cada terminal a su base de datos, ya que las cadenas de conexión son distintas para cada base de datos. Esto puede ser una tarea simple si son pocos integrantes (uno o dos), pero el hecho de tener 4 cadenas de conexión, y comentar y descomentar cada una a medida que se necesitaba acceder a la base de datos mostró que, en un proyecto a mayor escala, la utilización de una única base de datos en común es recomendada, si no necesaria.

Por lo que, en futuros proyectos, optaremos trabajar con una base de datos única, a pesar de que en la alternativa tomada hubiésemos generado la misma base de datos con el mismo script.

¿Qué base de datos utilizamos, y qué tablas utilizamos?

La base de datos utilizada en el proyecto fue la suministrada por la cátedra, ya que al haber optado por realizar uno de los proyectos propuestos por la cátedra, la base de datos suministrada contenía las tablas básicas necesarias para el desarrollo del proyecto.

Sin embargo, esta base de datos contenía además de las tablas necesarias para el proyecto otras tablas que eran utilizadas por otras propuestas de proyecto, que eran innecesarias para nuestro proyecto. Por lo tanto, se decidió modificar

esta base de datos, para que solamente contenga las tablas relevantes para nuestro trabajo, eliminando el resto de la base de datos.

Además, las tablas que permanecieron en la base de datos tuvieron que ser modificadas por distintas razones, siendo las principales las siguientes:

1) Mala generación de las tablas en el script. Algunas tablas poseían clave primaria autoincremental, lo cual era necesario para nuestro trabajo, pero otras no tenían esta característica, por lo que debimos modificar este detalle para que funcionen correctamente a la hora de insertar nuevos registros en la BD.
2) Atributos innecesarios. Como la base de datos originalmente daba soporte a varias propuestas de proyecto, algunas de las tablas relevantes para nuestro proyecto contenían atributos que eran necesarios para otras propuestas, pero que eran irrelevantes para nuestro trabajo, por lo que debimos modificar las tablas para poder eliminar estos atributos, con el consecuente problema de eliminar las claves foráneas de las tablas.
3) Atributos faltantes. Esto fue específicamente necesario a la hora de trabajar con los reportes y estadísticas, ya que normalmente éstos se realizan en base a *fechas*. Como algunas tablas (por ejemplo, la tabla Productos, que solamente contenía id y descripción) no contenían el atributo de fecha (por ejemplo, la fecha de alta del producto), los reportes generados no podían filtrar la información correctamente, por lo que debimos agregar estos atributos a las tablas.

Programación y estructuras de código

¿Cómo compartimos el código?

Para poder trabajar los 4 integrantes simultáneamente en el proyecto, y para evitar redundancias en el código, decidimos generar un repositorio en GitHub, en el cual subimos todo el proyecto, para así facilitar el desarrollo del mismo.

Sin embargo, los problemas que se nos presentaron tuvieron que ver más que nada con conflictos que se generaban en los metadatos del proyecto, cuya solución era significativamente más compleja que los conflictos en el código de las clases, aunque hablaremos de estos conflictos más adelante.

¿Cómo estructuramos el código?

Con respecto a la estructura del código, tuvimos que tomar la decisión más importante del inicio del proyecto, ya que esto iba a definir cómo íbamos a armar el proyecto en su totalidad, y cómo íbamos a distribuir la responsabilidad de las clases.

Se nos presentaron 2 opciones:

1) Realizar el proyecto de forma común, sin subdividir las clases, y contener todo lo relacionado a cada una en ella misma, permitiendo que maneje sus responsabilidades respecto a la presentación, la lógica de la clase, y el manejo de sus datos y persistencia.
2) Realizar el proyecto en capas, es decir, mantener el proyecto con una estructura en donde están bien separadas las capas de presentación, lógica de negocio y acceso a base de datos, y también manteniendo las clases de entidad que permitan enlazar estas tres capas.

En un principio, evaluamos seriamente realizar el proyecto según la primera alternativa, ya que solamente habíamos realizado ejemplos de trabajos de esa forma, y sabíamos que íbamos a necesitar realizar un esfuerzo adicional para poder hacer el trabajo en capas, ya que se requería que nosotros analicemos ejemplos de trabajos en capas suministrados por la cátedra para luego poder aplicar la metodología a nuestra aplicación. Sin embargo, finalmente decidimos programar nuestra aplicación en capas, ya que sabíamos que es una buena práctica de programación, y notamos que era una buena oportunidad comenzar desde cero con esta forma de trabajar para luego poder aprender a base de prueba y error.

Interfaces del proyecto

En general, no debimos pensar demasiado en cómo realizar las interfaces del proyecto, ya que como progresábamos en ejemplos prácticos a lo largo de la materia, las interfaces que implementábamos eran similares a las vistas en la cátedra.

Sin embargo, hubo algunas decisiones en las interfaces que debimos tomar. La primera de ellas fue cómo íbamos a realizar las interfaces para los formularios de alta, baja, modificación y consulta de nuestro proyecto, para las distintas entidades de soporte. Esto se debió a que vimos dos formas de realizar estas interfaces:

1) La primera de ellas fue vista en las clases prácticas, que constaba de un solo formulario, en el cual disponíamos los campos relacionados a la entidad de soporte, y dependiendo de los botones de “Nuevo”, “Editar” y “Borrar”, cada campo del formulario cumplía una función diferente, ya sea mostrar la información, editarla o generarla desde cero.
2) La segunda de ellas constaba de dos formularios, el primero siendo de consulta, donde a partir del mismo se podía acceder a un formulario extra que administraba la alta, baja y modificación de los registros de la base de datos. Esta opción es más prolija e intuitiva, ya que brinda inmediatamente la búsqueda de un registro de la base de datos, lo cual normalmente es lo que más se realiza en los formularios ABMC.

A pesar de lo previamente explicado, decidimos realizar las interfaces de los formularios ABMC según la primera alternativa, ya que cuando realizamos las interfaces no se nos había mostrado la otra opción para realizar dichos formularios.

Sin embargo, se evaluó la posibilidad de reestructurar estos formularios y realizarlos según la segunda alternativa, ya que dicha alternativa permitía una realización más directa de las consultas en el formulario, detalle que habíamos pasado por alto en las primeras implementaciones de los ABMCs. Pero luego se descartó esta alternativa, ya que por razones de esfuerzo y principalmente tiempo, no podíamos generar nuevamente todos los formularios de vuelta, ya que eran demasiado numerosos. Por lo que decidimos agregar un botón de búsqueda en el mismo formulario, y pensamos la forma en la que se pueda realizar la búsqueda borrando y permitiendo la gestión de los campos del formulario ya existentes.

Otra decisión que tomamos con respecto a las interfaces del proyecto fue respecto a la interfaz de usuario de la transacción principal de nuestro proyecto, la cual permitía generar una nueva factura de productos y/o proyectos. Nuestro problema surgió con respecto a las funcionalidades que brindaba este formulario, ya que tuvimos que decidir si en una misma factura podíamos permitir que se facturen proyectos y productos en simultáneo, o si solamente íbamos a permitir

que se puedan facturar solamente un tipo de éstos al mismo tiempo. Finalmente, decidimos permitir que se incluyan ambos en una misma factura, permitiendo al usuario que seleccione uno u otro antes de cargarlo a la factura mediante un botón, y transformando el formulario acorde a la selección del usuario.

Otro problema en esta interfaz fue decidir si se podían editar o no las facturas ya generadas. Dado que era una transacción, decidimos no permitir la edición de las facturas, aunque sí permitimos “copiar” una factura, es decir, tomar los datos de una factura previamente realizada para poder utilizarlos en una nueva factura, ya que, consultando con la profesora a cargo, esta es una práctica que se suele realizar en ambientes normales de gestión de pedidos, por lo que decidimos implementar esta funcionalidad.

Con respecto a los reportes y estadísticas generadas en la aplicación, solamente decidimos los formatos en los que íbamos a mostrar el proyecto.

Aprendizajes

A lo largo del proyecto, se realizaron distintos aprendizajes, ya que nunca habíamos trabajado en un proyecto que requiriera la participación constante de todos los integrantes del grupo en los aspectos de programación. Además, tampoco habíamos trabajado con el framework de .NET, por lo que también fue iniciar en un campo en donde no habíamos tenido experiencia.

Sin embargo, hay varios aspectos importantes relacionados al aprendizaje que tuvimos en el proyecto que entre los integrantes de nuestro trabajo consideramos fueron los más destacados:

1) **La utilización de los repositorios de GitHub y el manejo de la herramienta Git:** Aprender a compartir código es quizás la parte más importante de trabajar en conjunto en el desarrollo de una aplicación, más en las circunstancias actuales con respecto a la pandemia. Dado esto, aprender a compartir el código con la herramienta de Git, y utilizar los repositorios de GitHub nos ayudó mucho para poder seguir trabajando en grupo a pesar de que no se nos presentó la oportunidad de poder juntarnos a realizar las actividades del proyecto. Además, no solamente fue relevante para el proyecto realizado en esta materia, sino que también pudimos aplicar esto para el desarrollo de otros trabajos de programación requeridos en otras materias, como Diseño de Sistemas y Tecnología de Software de Base, en donde también tuvimos que realizar la implementación de programas a distancia.
2) **La programación en capas:** El poder estructurar el proyecto en capas nos facilitó mucho distribuir las distintas responsabilidades de las clases a lo largo del proyecto, así como la corrección de errores y el agregar nuevas funcionalidades a la aplicación, ya que permitió mantener una estructura limpia y cohesiva del proyecto. Es un proceso de aprendizaje complicado, porque al no tener experiencia realizando este tipo de trabajos, debimos empezar desde cero y por nuestra cuenta, pero rápidamente vimos las ventajas de realizar esto desde un principio, ya que pudimos dividir todo el proyecto en partes mucho más simples de mantener y de interrelacionar.
3) **La utilización de base de datos:** A pesar de que en otras materias vimos todos los fundamentos con respecto al manejo de base de datos, así como los distintos comandos SQL para la creación y actualización de las mismas, nunca habíamos implementado un programa que hiciera uso de la misma para mantener la información relevante, por lo que nos sirvió mucho aplicar estos conceptos en nuestro proyecto, primariamente para poder ejercer en la práctica lo que habíamos visto en la teoría, y para poder practicar algo que sabíamos que se utiliza mucho en los ámbitos laborales, y que de otra forma tendríamos que haber aprendido por nuestra cuenta, lo cual hubiese requerido un esfuerzo adicional.

4) **El manejo de los distintos componentes de interfaz en los formularios de Windows:** A pesar de que no todas las herramientas de creación de interfaces usan el mismo framework que el utilizado en el proyecto, y que otros lenguajes requieren distintos métodos y componentes para el manejo de las interfaces, los componentes de las interfaces suelen ser similares en varias herramientas, como los combobox, los checkbox, los textbox, las gridviews, y demás, por lo que lo aprendido en la creación de los formularios de la aplicación facilitó muchísimo el entendimiento de otras herramientas, como por ejemplo JavaFX para la creación de interfaces en Java. Además, como sabíamos que se suelen utilizar bastante los formularios de Windows para las aplicaciones utilizadas en sistemas de escritorio, lo realizado en el proyecto fue considerado una buena práctica para lo que un día vayamos a aplicar en un ámbito laboral.

Problemas encontrados a lo largo del proyecto

A lo largo de este documento se explicaron varios problemas que se nos presentaron, ya sea con las bases de datos, las interfaces, los conflictos generados en Git y su solución, y demás. Sin embargo, hubo otros problemas considerables que ocurrieron en el desarrollo del trabajo que consideramos notables de remarcar.

Uno de ellos fue un problema generado a nivel estructural del proyecto. Como creamos el proyecto inicialmente con otro nombre (ProyectoBugs), cuando cambiamos el nombre del proyecto, pudimos cambiar todos los namespaces de las clases, pero cuando uno creaba una clase desde cero, el namespace predeterminado era de ProyectoBugs en vez de LPCFacturas. Este problema era considerablemente menor a los otros generados en el proyecto, pero, sin embargo, no podíamos encontrar una solución para ello. Además, el IDE no nos permitía cambiar el nombre de la carpeta contenedora del proyecto, quedando la aplicación en general desprolija. Por lo que decidimos crear un proyecto nuevo desde cero, en donde copiamos todos los componentes de nuestro proyecto en el mismo, arreglando las dependencias y las referencias del mismo para que todo funcionase como corresponde. Sin embargo, la tarea fue bastante pesada, y demoró varias horas hasta que se solucionasen todos los errores provocados como consecuencia del traspaso de las clases del proyecto anterior en el nuevo proyecto.

Otro problema que encontrábamos era al momento de modificar los íconos en el proyecto, ya que los mismos para poder ser utilizados, tenían que ser referenciados en el proyecto, guardándose estas referencias en los metadatos del trabajo. Cuando modificábamos los íconos, y se generan conflictos en las salidas del proyecto, solíamos tener problemas al realizar el merge entre las distintas versiones del proyecto, ya que el merge realizado automáticamente por la herramienta normalmente generaba errores. Por lo tanto, tuvimos varias veces que volver a las versiones anteriores del trabajo para poder arreglar los errores generados en las referencias del proyecto.

Algo similar ocurría cuando generábamos nuevas clases o modificamos la estructura de las clases, ya que el IDE solía quitar las clases del proyecto, es decir, las clases seguían existiendo en las carpetas del proyecto, pero no eran tenidas en cuenta por el IDE al momento de mostrar el Explorador de Soluciones. Buscamos cómo resolver este problema, y no encontramos solución de forma inmediata, hasta que pudimos consultar con la cátedra acerca del mismo, y encontramos la forma de poder agregar las clases al proyecto sin necesidad de crearlas de vuelta para arreglar el problema.

Uno de los problemas más complejos que tuvimos fue en las últimas etapas del trabajo, con respecto a los paquetes nugget utilizados para la creación de los reportes, más específicamente los ReportViewer. Cuando creábamos un

ReportViewer, no nos mostraba los distintos DataSource del reporte, entonces no permítia cargar los datos en el reporte. La razón de esto fue que el IDE, Visual Studio, tenía problemas de compatibilidad en sus versiones más nuevas con las librerías de ReportViewer. Para solucionar este problema, cada vez que deseábamos modificar el DataSource del reporte, debíamos abrir el reporte con xml, y cambiar la versión del reporte, modificar el DataSource, y luego restaurar la versión original del reporte. Si bien es un problema de compatibilidad del IDE, decidimos notarlo ya que se puede tener en cuenta en futuros proyectos a la hora de elegir las versiones del IDE con las cuales trabajar.

Distribución de roles en el grupo

Con respecto a la distribución de roles en el grupo, no hicimos una distinción clara entre las responsabilidades de cada integrante, sino que cada integrante colaboraba en todos los aspectos del proyecto de una u otra forma. Si bien algunos integrantes eran más dados en las construcciones de interfaces, y otros eran más dados en el manejo de la base de datos, decidimos que era ideal que todos participaran en el desarrollo de todos los componentes del proyecto, ya que lo fundamental de este trabajo era poder practicar la implementación de los componentes que son fundamentales para el desarrollo de principio a fin de una aplicación.

En la mayoría de las actividades, distribuíamos las tareas entre los 4 integrantes, pero como varias de estas tareas requerían que solo un integrante modifique un componente a la vez (para poder evitar los conflictos en la herramienta Git), normalmente compartíamos pantalla entre varios integrantes, e íbamos aportando lo necesario para dejar funcionando correctamente el componente.

Conclusión

A lo largo del proyecto, aprendimos cómo utilizar muchas herramientas, ya sea para poder realizar los formularios, como también poder trabajar de una manera mucho más eficiente como equipo y a la distancia, en especial dadas las circunstancias en las que nos tocó cursar la materia.

Si bien tuvimos varias dificultades a lo largo del proyecto, explicadas previamente en este documento, nos vimos beneficiados por estos hechos, ya que pudimos aprender más sobre lo que se podía realizar y lo que no se podía realizar en el proyecto, en especial también el cómo poder encontrar las alternativas necesarias para poder realizar las funcionalidades que propusimos en la aplicación, ya sea en el código de la aplicación como también en la base de datos utilizada para el proyecto.