Proyecto *FloppaWare*: Programa de gestión escolar

Aranda Solis Ricardo
Universidad Nacional Autonoma de Mexico
Facultad de Ingeniería,UNAM
Ciudad de México, México
richardo2463@gmail.com

Garcia Gonzalez Alejandro
Universidad Nacional Autonoma de Mexico
Facultad de Ingeniería,UNAM
Ciudad de México, México
alebaalgg@gmail.com

Rodriguez Rodriguez Arturo
Universidad Nacional Autonoma de Mexico
Facultad de Ingeniería,UNAM
Ciudad de México, México
arturo.rodriguez.enp5.123@gmail.com

Sosa Cortez Misael
Universidad Nacional Autonoma de Mexico
Facultad de Ingeniería,UNAM
Ciudad de México, México
misaelsosacortez@gmail.com

Abstract—Este documento es la síntesis del software computacional *FloppaWare* realizado a través de la asignatura Programación Orientada a Objetos impartida en la Facultad de Ingeniería, UNAM. Se detalla su funcionamiento y la aplicación de este software de código abierto destinado al manejo de un sistema de gestión escolar.

Palabras clave—proyecto, software, programación, objetos, scrum, sistema, github, javaFX, archivos, registros, datos, inscripción, promedio, escolaridad,dirección, grupo.

I. Objetivo

En este proyecto se intentó replicar el sistema de gestión de alumnos de la facultad de ingeniería, las funciones como lo son de generación, edición, exportación y cálculo de número de inscripción son necesarias para la administración correcta de los alumnos de la carrera de ingeniería en computación.
Por medio de la tecnología de software de JavaFX es posible brindarle al usuario una experiencia más sencilla para interactuar con el programa, así como ser suficientemente eficiente como para que este actualice la información que se muestra al usuario.
Particularmente los objetivos/ funcionalidades son:

- Generar datos del alumno (personales y académicos)
- Calcular los números de inscripción
- Modificar los datos del alumno (no relacionados con el número de inscripción ni de cuenta)
- Exportar a .csv la tabla mostrada

II. Desarrollo del proyecto

I-A. Requerimientos de software

Los requerimientos del software son las necesidades y expectativas que un usuario o cliente tiene respecto a una aplicación informática. Los requerimientos del software deben ser identificados y especificados de manera clara y detallada antes de iniciar el desarrollo de una aplicación, ya que estos requerimientos sirven como guía para el equipo de desarrollo y determinan el alcance y las características del software final.

Algo que se debe de tener en cuenta para implementar un proyecto de software es que conforme el proyecto vaya creándose y haciéndose las entregas de avances al cliente, pueden ir cambiando los requerimientos del proyecto de acuerdo a las necesidades, gustos y que vaya satisfaciendo las necesidades del cliente, siendo posible anexar algunos requerimientos extra o quitar alguno que no sea importante para el cliente.

Para este proyecto, en los requerimientos de software se plantea un panorama general de las acciones que llevará a cabo dicho proyecto y un glosario con términos que se usarán en el programa y son imprescindibles para el perfecto funcionamiento del proyecto de software. También teniendo un panorama de cómo sería la interfaz gráfica a grandes rasgos como un menú de opciones con el cual interactuar.

En el último paso se hace un análisis de la confiabilidad, la eficiencia y las restricciones de diseño, siendo necesario confirmarle al cliente si todo lo que se pide puede ser llevado a las normas de un software.

I-B. Diseño del Software

El diseño del software es el proceso de definir y planificar la estructura y el comportamiento de una aplicación informática. El diseño del software incluye la definición de la arquitectura del sistema, la selección de tecnologías y herramientas a utilizar, y la creación de modelos y diagramas que representen cómo se integran y comunican entre sí los distintos componentes del sistema.

El objetivo del diseño del software es asegurar que la aplicación cumpla con los requerimientos del usuario y sea fácil de desarrollar, mantener y actualizar. Para ello, es necesario tomar en cuenta aspectos como la escalabilidad, la flexibilidad, la mantenibilidad y la seguridad del sistema.

Bajo este panorama el software diseñado requirió priorizar las características antes mencionadas y por lo tanto implementó la arquitectura basada en el patrón *ModelViewConroller*. En este patrón, la aplicación se divide en tres componentes principales:

- La vista: es la parte de la aplicación que se encarga de presentar la información al usuario y recibir sus acciones. La vista es presentada a través de la interfaz de usuario diseñada con JavaFX.
  JavaFX es una biblioteca de Java para el desarrollo de aplicaciones gráficas de usuario (GUI, por sus siglas en inglés). JavaFX proporciona un conjunto de clases y componentes que permiten crear interfaces de usuario atractivas y funcionales de manera sencilla. Incluye un lenguaje de programación declarativo llamado FXML, que permite definir la interfaz de usuario de una aplicación de manera visual y separada del código de programación.

- El modelo: Es la parte de la aplicación que se encarga de almacenar y gestionar los datos y la lógica de negocio. El modelo puede incluir componentes como bases de datos, clases de objetos o librerías de funciones, en este caso particular, se trata del manejo y manipulación de la información como la generación de datos, edición, recopilación y organización de la misma.

- El controlador: es la parte de la aplicación que se encarga de gestionar la interacción entre la vista y el modelo. El controlador recibe las acciones del usuario a través de la vista, procesa la información y actualiza el modelo en consecuencia, y luego actualiza la vista para reflejar los cambios. Por lo tanto, para cada opción mostrada en la interfaz de usuario será necesario crear un controlador, por ejemplo controlador para la consulta de datos, controlador para la modificación de datos del alumno,etc.

I-C. Implementación de la metodología SCRUM

Para llevar un control al momento de ir elaborando el proyecto, se hace uso de la metodología SCRUM con el fin de realizar un mejor control y tener calidad al realizar el producto. Haciendo uso de la metodología esto para llevar un mejor orden, repartir tareas; todo esto con el fin de tener orden y ser más rápidos desarrollando el software, dejando en claro cuál sería el producto esperado desde un principio.

También elaboramos un Product Backlog para cada Sprint. Al final de cada retrospectiva evaluamos todas las tareas dejando en claro las que se realizaron de manera exitosa y las que no se concluyeron, esto con el fin de terminarlas para el siguiente sprint tratando de terminarlas en un solo sprint para no tener retrasos de tiempo en el diseño de software.

I-D. Control de versiones con Git y GitHub

Git es una plataforma que permite a los desarrolladores de software tener un

repositorio con las diferentes versiones del software en desarrollo y trabajar en equipo de manera más eficiente sin la necesidad de que el equipo de desarrollo esté en un mismo lugar.

Git también nos permite tener nuestra propia copia del repositorio lo cual hace más fácil trabajar con un versión y recuperar la anterior fácilmente.

I-E. Validación del Software

Validar un software es el proceso en el cual verifican y validan que una aplicación de software cumpla con los requerimientos que se dieron en un principio del proyecto y los que se fueron agregando. Se lleva a cabo con el fin de asegurarse que el cliente esté satisfecho con el producto entregado.

En *FloppaWare* hemos cumplido cada unos de los requerimientos y expectativas del programa, programando de manera avanzada para poder tener un producto final con resultados bastante favorables en el uso de este software.

III. Funcionalidad e implementación

I-A. Clases del patrón de diseño

- **Modelo:** Alumno, Calificación, Dirección, Edad, GeneradorNombre, GeneradorNumCuenta, Materia, Promedio.
- **Vista:** Están embebidas en la clase GUIvistaControlador, además que el archivo que contiene toda la información acerca de los objetos gráficos es el archivo GUIvista.fxml, el cual fue generado por un software de edición de GUI (JavaFX Scene Builder)
- **Controlador** GUIvistaControlador, ProyectoPOO.

I-A. Descripción del programa

El programa inicia con la muestra de varios objetos gráficos, como lo son:

- TableView: La tabla que contiene a los datos
- TableColumn: La columna especifica que contiene un tipo de dato
- Label: Un cuadro de texto
- Button: Un botón que al presionarlo algo ocurre
- ChoiceBox: Un selector de opciones

I-B. Generación de datos del alumno

La funcionalidad al momento de la ejecución del programa, es con el selector de tareas en la opción “Generar datos del alumno”, al momento de presionar el botón de inició saldrá una ventana emergente, la cual nos avisará de que se han generado los datos de los alumnos y se mostrará la información de los alumnos en la tabla.
En el código esto se logra con una clase (Alumno) la cual nos permite setear los datos de los alumnos, podemos replicar este molde de datos para todos los alumnos de nuestro modelo, es decir se inicializa un objeto alumno, se llenan los datos con los métodos del paquete modelo.
La fortuna del uso de esta clase constructora vendrá cuando se quiera modificar los datos, ya que nos permite modificarlos desde cada objeto y se actualizan casi automáticamente en la vista y en modelo.
Los datos son llenados, por los métodos que se encuentran en el paquete de modelo o en dado caso en generadores de números aleatorios en la misma función de la clase (GUIvistaControlador).

- Para la generación de nombres y apellidos se escoge de un arreglo aleatoriamente y se le asigna la cadena que se encuentre en el mismo.

- Para la edad se genera aleatoriamente con algunas restricciones, no puede haber alumnos tan jóvenes cursando semestres avanzados y viceversa, por lo que se evalúa el semestre generado, dentro de la clase modelo y se brinda la edad.

- Para el número de cuenta, se genera con números aleatorios de dos cifras que se concatenan para formar un número más amplio, un número de cuenta no puede empezar con 0
- El semestre es calculado con un número aleatorio del 1 al 10 en la clase controlador

- El promedio se calcula simulando el mapa curricular de la carrera de Ingeniería en Computación y las materias que contiene en cada semestre, las materias con calificación

reprobatoria no son contabilizadas en el promedio.

- La escolaridad se calcula según el coeficiente de materias aprobadas con las materias inscritas multiplicado por 100 aquí nuevamente se simula el mapa curricular y se ve el número de materias inscritas por semestre

- La velocidad es calculada según los créditos obtenidos del alumno, nuevamente se simula el mapa curricular con respecto a los créditos por semestre, si no se aprobó una materia no se obtienen los créditos, se obtiene el coeficiente de los créditos obtenidos y los reglamentados multiplicado por 100

- Indicador escolar: Multiplica todos los datos
- Dirección Se tiene una base de datos cargada en el archivo, la cual es leída linea por linea en un arreglo y luego seleccionada aleatoriamente de este arreglo

I-C. Calcular números de inscripción

Está opción nos permite generar un número de inscripción para cada alumno generado. Al presionar el botón "Iniciar" automáticamente se habrán generado números de inscripción únicos para cada alumno.

Se ordenan los números del indicador escolar de mayor a menor, y se le asigna al mayor el primer número de inscripción y así sucesivamente. Para calcular dicho número fue necesario programar el siguiente algoritmo matemático:

- El elemento principal para el cálculo es el Indicador escolar, el cual pondera el promedio de los alumnos respecto a su aprovechamiento y avance, con base en su semestre de ingreso y plan de estudios.

$$Indicador\,escolar = (Promedio) \times (Escolaridad) \times (Velocidad)$$

- Por otro lado, para el promedio aritmético se consideran exclusivamente las calificaciones numéricas (5 a 10) y los otros factores quedan definidos como:

$$Escolaridad = \frac{Asignaturas\ aprobadas\ en\ ordinario}{Asignaturas\ inscritas\ en\ ordinario} \times 100\%^{\#}$$

 # Los exámenes extraordinarios no son contabilizadas para este cálculo.

$$Velocidad = \frac{Créditos\ del\ alumno}{Créditos\ desde\ el\ ingreso^{*}} \times 100\%$$

** Los créditos que debería tener el alumno de acuerdo a los semestres desde su ingreso; descontando, de ser el caso, los semestres autorizados en suspensión temporal de estudios.*

- Finalmente, una vez obtenido el indicador de todos los alumnos que se inscribirán, se ordenan en forma descendente, asignando el primer número de inscripción al primero de la lista y así consecutivamente hasta llegar al alumno con el menor indicador.

I-D Modificar datos del alumno

Por medio del objeto *Tableview* se activa la edición de datos, posteriormente, por cada columna se espera el evento de hacer click dos veces en el nombre para hacerlo un cuadro de texto donde se pueda escribir, para aceptar los cambios solo basta con dar enter. Después de cambiar a cualquier otra opción se desactiva la edición de elementos de la tabla.
Los rubros que están prohibidos modificar son todos aquellos relacionados con el cálculo del número de inscripción, así como los números de cuenta, esto con el objetivo de evitar fraudes con la inscripción.

I-E. Exportar datos a un archivo .csv

Cuando se selecciona esta opción lo que se hace es retomar la lista que es el backend de la Tableview y recorre la misma, para que por cada elemento se pueda emplear una operación get, que por medio de un BufferedWriter y un PrintWriter, nos permite escribir en un archivo .csv separado por comas los elementos. Cabe resaltar que el archivo es creado si no existe, sin embargo sí existe y el archivo está utilizándose, existirá una excepción.
Dentro del archivo se podrá editar, pero no se cambiarán los datos del mismo a la tabla nuevamente, sin embargo si se hace alguna modificación en el programa y se vuelve a exportar, el cambio se exportará igualmente.

IV. Conclusiones

Aranda Solis Ricardo:
El proyecto se enfocó en aplicar la metodología SCRUM en nuestro contexto, por lo que se fueron evaluando las versiones incrementales de nuestra implementación, si bien no era necesario tener un modelo que tuviera en mente una interfaz gráfica, fue una manera de poder organizar nuestro código según nuestro patrón de diseño MVC, lo que nos permitió conocer nuevas formas de organización como lo son las clases constructoras de objetos, y de cómo podemos escalar un objeto general como lo son los datos de los alumnos a una serie de objetos que conformen datos en una tabla, y aun así de este mismo objeto tener objetos como las columnas que dependen de la tabla. Una estrategía que nos ahorró bastante tiempo en el diseño de la GUI fue el uso del scene builder que generó el archivo FXML ya que fue sencillo hacer el layout de todos los objetos.

García González Alejandro:
La creación de este proyecto fue de gran ayuda para entender cómo se lleva a cabo el desarrollo de un proyecto empezando por la planificación y requerimientos que se desean obtener. Aprendimos a elaborar un proyecto de manera profesional usando una metodología que nos permitió llevar un control de desarrollo más profesional y entendible, sirviendo está metodología para futuros proyecto de materias que lo requieran.
En conclusión fue un proyecto bastante complejo ya que para su elaboración se imprenta todo lo visto a lo largo del curso por lo que al finalizar este proyecto se ha adquirido mayor experiencia en el lenguaje orientado a objetos como lo es Java.

Rodríguez Rodríguez Arturo:

En conclusión, el proyecto de desarrollo de un programa de gestión escolar basado en el paradigma orientado a objetos ha demostrado ser un éxito en términos de alcanzar sus objetivos y superar las expectativas. La implementación del programa ha permitido una mayor flexibilidad y reutilización del código, lo que ha contribuido a una mayor eficiencia en el desarrollo y mantenimiento del software.

Además, la aplicación de SCRUM ha facilitado una mayor colaboración entre los desarrolladores y ha permitido un seguimiento más preciso del progreso del proyecto, lo que ha contribuido a una mayor efectividad en el proceso de trabajo. En general, se puede decir que la combinación del paradigma orientado a objetos y SCRUM ha demostrado ser una elección acertada para este proyecto de gestión escolar y se recomienda considerar esta opción para proyectos similares en el futuro.

Por otro lado, el enfoque orientado a objetos ha permitido una mejor representación de los conceptos y entidades involucrados en la gestión escolar, lo que ha facilitado la integración del programa con otras herramientas y sistemas como lo fue el diseño de una interfaz gráfica a través de javaFX.

En general, se puede decir que el uso del paradigma orientado a objetos combinado con el marco de trabajo SCRUM ha demostrado ser una elección acertada para este proyecto de gestión escolar, ya que ha permitido una mayor eficiencia y flexibilidad en el desarrollo y mantenimiento del software. Sin embargo, también es importante señalar que el enfoque orientado a objetos no es la única opción disponible y que cada proyecto debe evaluar cuidadosamente las diferentes opciones de diseño y arquitectura para determinar la mejor opción para sus necesidades y requisitos específicos.

Sosa Cortez Misael Ivan:
La creación de este proyecto se basó enteramente en la generación de un programa de gestión, creación y modificación de datos escolares a partir de los métodos aprendidos e implementados junto con la utilización de la interfaz gráfica por parte de JavaFX que permite un manejo idóneo de el propio programa, además de una colaboración que permite eficientar la corrección y desarrollo del proyecto

Para la realización de este proyecto, nos enfocamos en aplicar la metodología SCRUM aprendida durante el semestre con el objetivo de la optimización y manejo correcto de

periodos de tiempo y realización de tareas desde un periodo de planificación hasta la realización de iteraciones para cada punto de este mismo permitiéndonos un mejor control y gestión de proyecto realizandolo de una manera profesional

V. Github y Github Pages

Github:
https://github.com/ghostdevil24/POO-23-1-EquipoH-ProyectoFinal
Github pages:
https://ghostdevil24.github.io/POO-23-1-EquipoH-ProyectoFinal/