Desarrollo de Sistema de Gestión Escolar

Integrantes : Santos Carmona Valeria Sofia | Hernandez Reyes Rebeca Sarai | Almaguer Miranda Michel Santiago | Leyva Campos Eddie

Introducción:
En este proyecto se busca desarrollar un Sistema de Gestión Escolar que estará orientado hacia la administración de datos de alumnos pertenecientes a la facultad de Ingeniería de la UNAM, utilizando Java como entorno. Teniendo como propósito diferentes acciones, tales como la creación, el almacenamiento y la modificación de datos personales de los alumnos.
Con este programa no se busca únicamente el almacenamiento y la gestión de datos, si no, que también pretende simular procesos que la administración de dicha facultad requiere, como es la asignación del número de inscripción y la exportación de los datos a un archivo csv para que esos mismos datos puedan ser analizados fuera del programa.

Objetivo General:
Desarrollar un programa que sea capaz de gestionar los datos personales y académicos de los alumnos, así como su creación, actualización, eliminación y consulta, de forma clara.

Objetivos Específicos:
Implementar un sistema que ponga en práctica las bases de la Programación Orientada a Objetos, que logre la generación de 1000 alumnos con datos realistas y cuente con un módulo CRUD que permita la administración de sus datos, así como la simulación de un sistema de asignación de número de inscripción que trabaje acorde al algoritmo que se implementa en la Facultad de Ingeniería, y por último, que permita ser exportado a un formato csv para la visualización de los datos de los alumnos desde una hoja de cálculo.

Formación del equipo y definición de roles:
Para el inicio del proyecto se comenzó con la estructuración del mismo asignando las responsabilidades a cada uno de los integrantes y la creación de los perfiles en GitHub .
La distribución quedó de la siguiente manera:
Líder del proyecto: Almaguer Miranda Michel Santiago.
Desarrolladores: El equipo técnico está conformado por Hernández Reyes Rebeca Sarai, Santos Carmona Valeria Sofia y Leyva Campos Eddie.

Requerimientos de sistema:
Se establece la base de lo que nuestro Software tiene que hacer y cómo es que debe comportarse llegando a los siguientes puntos:
Requerimientos funcionales:

- Capacidad para crear 1000 alumnos.
- Cada alumno deberá contar con datos como son el nombre, edad, número de cuenta, semestre, dirección y número de inscripción.
- El administrador podrá crear, consultar, eliminar y actualizar información de los alumnos

inscritos, mediante un módulo CRUD.
- Se necesita generar números de inscripción teniendo en cuenta el sistema que utiliza la Facultad de Ingeniería de la UNAM que se basará en un indicador académico.
- El código tiene que exportar la información a formato .csv para poder usarlo de manera externa en excel.

Requerimientos no funcionales:
- El sistema deberá ser funcional en diversos sistemas operativos.
- El código deberá ser claro, comprensible y modular.
- Deberá permitir escalabilidad futura .
- Eficiencia en la gestión de datos.

Diseño de software:
El equipo en conjunto definió soluciones bajo el patrón de arquitectura MVC.
- El sistema prioriza facilidad de comprensión, modularidad y portabilidad, de modo que pueda ejecutarse en distintos entornos (Windows/Linux/macOS) sin dependencias pesadas. Para lograrlo, se adopta una arquitectura basada en el patrón Model–View–Controller (MVC), complementada con servicios de dominio y un repositorio de persistencia en archivos de texto.

Estructura :
- Modelo: Gestión lógica de negocio del sistema y sus entidades (Alumno , Dirección , Materia y Historial Academicpo) y las reglas para validar esta información encapsulando los atributos y sus comportamientos propios de cada una de las identidades.
- Vista: Se definen los paquetes encargados de la interacción con el usuario, los menús y el despliegue de los datos.
- Controlador: Coordina la generación de nombres , inscripciones y la exportación de los archivos. El controlador actúa como intermediario entre la vista y el modelo, garantizando que las operaciones solicitadas se realicen correctamente y los resultados se muestren al usuario.

También coordina la interacción entre los diferentes módulos del sistema, como la generación de datos, la gestión CRUD y la exportación de archivos.

Vista del sistema:
1. Vista de despliegue: Equipo local (cliente) : Computadora con entorno java y Servidor de archivos con nodo encargado de poder guardar los archivos .json1 y .csv para su edición en excel como herramienta externa.
2. Vista lógica:

Clases de Vista: menú, crud_alumno, listar_alumnos, Generar_datos, Asignar número de inscripción, Exportar_csv.

Clases Modelo: Alumno, Número de Inscripción, Registro de materias, Historial

académico, Dirección, Materias.

Clases de Controlador: menu_controller, Generar_datos_controller, crud_alumno_controller, Generar_nombres_controller,generar_número_controller, Exportar_csv_controller.

3. Vista de Datos: alumno,historial académico , calificaciones , materias , dirección y semestre.
4. Vista Dinámica : Utilizamos diagramas de secuencia para que se pueda modelar como es que interactúan los objetos definidos : Agregar Alumno , Generar número de inscripción , Generar nombre , Editar alumno y eliminar alumno.

Principios de POO aplicados:

Para realizar este proyecto fue necesario el uso de varios principios de POO, tales como:

- Encapsulamiento: Cada clase controla su propia información mediante métodos getters y setters.
- Abstracción: Se modelan entidades reales, como Alumno.
- Modularidad: Cada clase cumple una función específica.
- Reutilización: Las clases pueden ser reutilizadas en distintos módulos.

Funcionamiento del sistema:

Nuestro sistema inicia mostrando un menú principal, donde el administrador podrá elegir la operación que requiera entre las opciones disponibles. Al generar alumnos el sistema les asigna a cada uno de ellos datos como, nombre o nombres y apellidos aleatorios, edad y direcciones. Con estos primeros datos ya asignados se procede a generar sus registros académicos, que incluyen materias con calificaciones y estados que van entre aprobado e inscrito. Con estos datos se procede con el indicador académico que determinará el número de inscripción correspondiente a cada alumno. Y si se requiriera hacer modificaciones, actualizaciones, eliminaciones, creaciones o una simple visualización de los estudiantes, será posible gracias al módulo CRUD, y finalmente toda la información generada podrá ser exportada a un documento csv para su manipulación externa.

Sprints y metodología de desarrollo:

Se dividió la construcción del Software mediante la metodología de desarrollo ágil, implementando iteraciones conocidas como Sprints que permitieron dividir el proyecto en etapas controlables para administrar el tiempo y lo recursos de manera eficiente:

Sprint 1: Datos Personales donde nuestro enfoque fue establecer la estructura de la captura de datos con el desarrollo de HU01 Y HU02 para capturar datos personales , nombres y apellidos además de agregar interfaces de captura y validar campos.

Sprint 2: Se creó la clase para generar los nombre (GeneradorNombres) con los arreglos de nombres y apellidos además de implementar la lógica para que el 20% de los alumnos tengan dos nombres.
Se trabajó en la generación de manera aleatoria de edades (18 a 27) y dependiendo de la edad se asigna el semestre al que pertenezca junto con los números de cuenta.

Sprint 3: Generamos y definimos la lista de las 50 asignaturas y agregamos la lógica para que se asocie cada materia a los alumnos generados, integrando el historial académico a la clase Alumno. Además, se creó la clase RegistroAcademico, encargada de almacenar las materias cursadas, calcular el promedio y los créditos acumulados de cada alumno. En este mismo sprint se diseñó e implementó la clase CalculadoraNumeroInscripcion, la cual calcula un indicador escolar a partir del promedio, el porcentaje de materias aprobadas y el avance en créditos respecto a los créditos ideales del semestre, ordenando a los alumnos de forma descendente para asignar su número de inscripción siguiendo el esquema de la Facultad de Ingeniería de la UNAM. Finalmente, se desarrolló el módulo CRUD de alumnos y la funcionalidad de persistencia externa mediante archivos CSV, permitiendo listar, editar y eliminar alumnos, así como exportar toda la información a un archivo alumnos.csv e importarla nuevamente al sistema para continuar trabajando con los mismos datos.

Alcances del sistema:
El programa permite la simulación completa de un sistema de gestión de registros académicos, desde la creación de los alumnos, y sus datos personales y académicos, hasta la asignación de un número de inscripción basado en un indicador escolar y la exportación de estos mismos datos a un archivo tipo csv.

Conclusiones:
El desarrollo del Sistema de Gestión Escolar permitió aplicar de manera práctica los conceptos fundamentales de la Programación Orientada a Objetos, integrando teoría y práctica en un proyecto funcional que simula procesos reales utilizados en la administración académica. A lo largo del proyecto se logró modelar correctamente entidades como Alumno, Materia y RegistroAcadémico, así como implementar mecanismos para la generación automática de datos, cálculo de indicadores escolares y asignación de números de inscripción basados en el desempeño académico.
La implementación del sistema demostró la importancia de una correcta organización del código mediante el uso del patrón de arquitectura MVC, lo cual facilitó la separación de responsabilidades y permitió mantener un flujo claro entre la lógica del sistema, la interacción con el usuario y el manejo de los datos. Asimismo, la incorporación de un módulo CRUD proporcionó una herramienta eficiente para la gestión dinámica de la información, permitiendo la modificación de los

datos sin necesidad de reiniciar el proceso desde cero.

El Sistema de Gestión Escolar desarrollado cumple con los objetivos planteados, demostrando la aplicación efectiva de la Programación Orientada a Objetos y buenas prácticas de diseño de software. Su estructura modular permite escalabilidad y mejora constante, sirviendo como una base sólida para proyectos futuros de mayor complejidad.