Juego Educativo de patrones Matemáticos.

Requerimientos funcionales y no funcionales.

Derlys Romero Graciani

Kevin Armando Garzón Forero

Juan Mora

Practica aplicada a sistemas.

06 de febrero 2026

Contexto:

- El sistema es un juego interactivo educativo en el que el usuario introduce valores numéricos iniciales. Además, utiliza una modificación matemática fundamentada en operaciones (resta) y presenta el resultado que se ha obtenido.
- El objetivo del usuario es observar los resultados, determinar el modelo de modificación y confirmar su entendimiento a través de una prueba.

Requerimientos funcionales:

Sección: Información y guía del usuario.

- El sistema debe mostrar una pantalla de bienvenida al iniciar la aplicación.
- El sistema debe presentar las instrucciones y reglas del juego antes de comenzar.
- El sistema debe permitir al usuario iniciar el juego mediante un botón de acción.
- El sistema no debe permitir interactuar con el juego antes de visualizar las instrucciones.

¿Como podemos hacer esto?: **Este requerimiento podemos lograrlo implementando una pantalla inicial de bienvenida en la interfaz gráfica desarrollada en JavaFX, donde se muestren las instrucciones y reglas del juego antes de iniciar. Se puede agregar un botón "Iniciar Juego" que permita continuar únicamente después de leer la información. Esto garantiza que el usuario comprenda el funcionamiento del sistema antes de interactuar con él.**

Ingreso y validación de datos.

- El sistema debe permitir al usuario ingresar valores numéricos iniciales para ejecutar la operación.
- El sistema debe validar que los datos ingresados sean numéricos antes de procesarlos.
- El sistema no debe permitir campos vacíos.

- El sistema debe mostrar un mensaje de error si el usuario ingresa caracteres no numéricos.
- El sistema debe permitir confirmar el ingreso de datos mediante un botón de acción.

¿Como podemos hacer esto?: **Podemos lograr este requerimiento utilizando campos de texto (TextField) en la interfaz y aplicando validaciones desde el controlador. Se debe verificar que los campos no estén vacíos y que los valores ingresados sean numéricos utilizando métodos de conversión y manejo de excepciones. En caso de error, el sistema mostrará mensajes en color rojo, evitando que la operación se ejecute hasta que los datos sean correctos.**

Procesamiento matemático.

- El sistema debe aplicar la regla de sustracción definida a los valores ingresados.
- El sistema debe generar automáticamente el resultado después de presionar el botón de ejecución.
- El sistema no debe permitir ejecutar la operación si los datos no han sido validados correctamente.

¿Como podemos hacer esto?: **Podemos cumplir este requerimiento mostrando el resultado en un Label visible dentro de la interfaz y registrando cada operación en una estructura de datos o tabla visual (TableView). Además, se debe capturar la fecha y hora mediante clases del sistema (como LocalDateTime) para almacenarlas junto con la operación, permitiendo así un historial organizado y numerado.**

Visualización de resultados.

- El sistema debe mostrar el resultado de la operación en la interfaz gráfica de manera clara y visible.
- El sistema debe mostrar los resultados en un historial ordenado y numerado.
- El sistema debe registrar la fecha y hora de cada operación realizada.

¿Como podemos hacer esto?: **Podemos cumplir este requerimiento mostrando el resultado en un Label visible dentro de la interfaz y registrando cada operación en una estructura de datos o tabla visual (TableView). Además, se debe capturar la fecha y hora mediante clases del sistema (como LocalDateTime) para almacenarlas junto con la operación, permitiendo así un historial organizado y numerado.**

Validación del aprendizaje

- El sistema debe indicar si la respuesta es correcta o incorrecta.
- El sistema debe permitir un máximo de tres intentos para responder la prueba.
- El sistema debe mostrar la respuesta correcta después del tercer intento fallido.

¿Como podemos hacer esto?: **El sistema debe evaluar la respuesta ingresada por el usuario e indicar si es correcta o incorrecta. El usuario podrá realizar un máximo de tres intentos. Si después del tercer intento la respuesta continúa siendo incorrecta, el sistema deberá mostrar la respuesta correcta.**

Control del juego

- El sistema debe permitir reiniciar el juego mediante un botón de reinicio.
- El sistema debe solicitar confirmación antes de reiniciar para evitar pérdida accidental de datos.
- El sistema debe permitir ejecutar múltiples operaciones sin cerrar la aplicación.

¿Como podemos hacer esto?: **Podemos lograr este requerimiento implementando un botón de reinicio que limpie los campos y resultados visibles. Antes de realizar esta acción, se mostrará una ventana de confirmación para evitar pérdida accidental de información. Además, el sistema no debe cerrarse después de cada operación, permitiendo múltiples ejecuciones continuas.**

Persistencia de datos

- El sistema debe almacenar el historial de operaciones en una base de datos relacional.
- El sistema debe recuperar el historial almacenado al reiniciar la aplicación.
- El sistema debe permitir limpiar el historial almacenado.

¿Como podemos hacer esto?: **Este requerimiento se cumple conectando el sistema a una base de datos relacional donde se almacenen las operaciones realizadas. Se puede crear una tabla que guarde los valores ingresados, el resultado y la fecha. Al iniciar la aplicación, el sistema debe consultar la base de datos para cargar el historial almacenado. También se debe implementar una opción para eliminar los registros si el usuario lo solicita.**

Requerimientos no funcionales

Usabilidad

- La interfaz debe desarrollarse en JavaFX.
- El sistema debe utilizar fuente Arial.
- El tamaño mínimo de letra debe ser 14 px para textos y 18 px para títulos.
- El fondo debe ser color blanco (#FFFFFF).
- El texto principal debe ser color negro (#000000).
- Los resultados correctos deben mostrarse en verde (#00FF00).
- Los mensajes de error deben mostrarse en rojo (#FF0000).
- Los títulos deben mostrarse en azul oscuro (#003366) y en mayúscula.

¿Como podemos hacer esto?: **Podemos lograr este requerimiento diseñando la interfaz en JavaFX y aplicando estilos mediante hojas CSS. Se configurará la fuente Arial, tamaños mínimos de texto, colores específicos para títulos, errores y resultados correctos, además de fondo blanco y texto negro. Esto asegura una interfaz clara, legible y visualmente coherente.**

Rendimiento

- El tiempo de procesamiento de cada operación no debe superar de 2 a 5 segundos.
- El sistema debe permitir ejecutar al menos 20 operaciones consecutivas sin degradar el rendimiento.

¿Como podemos hacer esto?: **Este requerimiento se logra asegurando que la operación matemática sea simple y eficiente, evitando procesos innecesarios o ciclos largos. Además, se deben cerrar correctamente las conexiones a la base de datos para evitar sobrecarga, garantizando que el sistema pueda ejecutar múltiples operaciones consecutivas sin afectar el desempeño.**

Confiabilidad

- El sistema debe garantizar que la misma entrada produzca siempre el mismo resultado.
- El sistema no debe generar resultados inconsistentes.

¿Como podemos hacer esto?: **Podemos garantizar este requerimiento implementando una lógica matemática determinística en el modelo, donde la misma entrada siempre produzca el mismo resultado. También se deben realizar pruebas repetidas con los mismos valores para verificar consistencia en los resultados.**

Compatibilidad

- El sistema debe ejecutarse en cualquier sistema operativo que soporte Java 17 o superior.

¿Como podemos hacer esto: **Este requerimiento se logra desarrollando el sistema en Java 17 o superior y evitando dependencias incompatibles con otros sistemas operativos. Además, el sistema puede empaquetarse como archivo ejecutable (.jar) para facilitar su ejecución en diferentes entornos.**

Seguridad y validación

- El sistema debe validar los datos antes de almacenarlos en la base de datos.
- El acceso directo a la base de datos no debe estar disponible para el usuario final.

¿Como podemos hacer esto?: **Podemos cumplir este requerimiento validando todos los datos antes de almacenarlos en la base de datos y separando la capa de acceso a datos dentro del modelo o paquete de persistencia. De esta manera, el usuario no tendrá acceso directo a la base de datos y se protege la integridad de la información.**

Mantenibilidad

- El sistema debe estar organizado bajo una arquitectura basada en el patrón MVC.
- El código debe estar modularizado en paquetes claramente definidos.

¿Como podemos hacer esto?: **Este requerimiento se logra organizando el proyecto bajo el patrón MVC, separando claramente el modelo (lógica y datos), la vista (interfaz gráfica) y el controlador (gestión de eventos). Además, el código debe dividirse en paquetes bien definidos, lo que facilita futuras modificaciones y mantenimiento del sistema.**

Aspectos adicionales

Objetivo del juego: Fortalecer la comprensión de la operación de sustracción mediante la resolución de ejercicios y la identificación de la regla matemática aplicada.

1. El usuario debe ingresar dos números válidos.

El sistema solo permitirá valores numéricos para realizar la operación de sustracción.

2. El sistema calculará automáticamente el resultado.

Se aplicará la operación de resta (primer número menos segundo número).

3. El usuario debe identificar la regla aplicada.

Después de ver el resultado, el jugador debe indicar cuál fue la regla matemática utilizada.

4. El usuario tiene máximo tres intentos.

Si responde incorrectamente, podrá intentarlo nuevamente hasta un límite de tres oportunidades.

5. Si falla los tres intentos, el sistema mostrará la respuesta correcta.

Esto permite reforzar el aprendizaje.

6. Los resultados de la partida serán almacenados.

Cada operación realizada quedará registrada para su consulta posterior.

7. El usuario puede reiniciar la partida.

Podrá comenzar nuevamente cuando lo desee.