



**UNIVERSIDAD DE LAS AMERICAS**

**INGIENERIA EN SOFTWARE**

**FACULTAD DE INGENIERÍA Y CIENCIAS  
APLICADAS**

**PROGRAMACION III**

**TUTOR**

**PAULO GUERRA**

**AUTORES**

**EMILY MULLO**

**LUIS MORALES**

**JOHAN GAMBOA**

**LUCAS KARLSSON**

**TAREA**

**TALLER S7**

## Formulación del problema

El sistema de gestión del rol de misiones de los Avengers requiere un sistema que permita gestionar de manera organizada la información relacionada con cada héroe, incluyendo su identificación, misión, nivel de peligrosidad y compensaciones económicas. Además, es necesario automatizar el cálculo del aporte al Fondo de Héroes, los impuestos gubernamentales y el pago neto mensual. Para resolver esta necesidad, se requiere desarrollar una aplicación basada en programación orientada a objetos (POO) que permita gestionar de manera eficiente los datos de cada Avenger, procesar cálculos financieros y presentar reportes mediante una interfaz gráfica accesible para el usuario.

## Entidades involucradas en el problema

Entidad	Descripción
Avenger	Miembro del equipo con sus datos personales y de misión, incluyendo cálculo de aportes, impuestos y pago neto.
Equipo	Gestiona la colección de Avengers, permitiendo registrar, editar y buscar miembros.
Ventana	Interfaz para interactuar con el sistema: registrar, editar, buscar y generar reportes de Avengers.

## Atributos identificados

Atributo	Tipo	Descripción
id	String o int	Identificador único del Avenger
nombre	String	Nombre del Avenger
misión	String	Misión asignada
nivelPeligrosidad	int	Nivel de riesgo del 1 al 5
pagoMensual	float	Pago mensual asignado

## Métodos necesarios

Método	Descripción
registrarAvenger()	Agregar un Avenger
editarDatos()	Actualizar la información
getLista ()	Imprimir la lista de Avengers
buscarDatos()	Buscar por ID
calcularAporte()	Calcular el 8% del pago mensual
calcularImpuesto()	Calcular los impuestos según la tabla anual
pagoNeto()	Pago mensual restado los impuestos

## Datos de entrada, procesos y salida

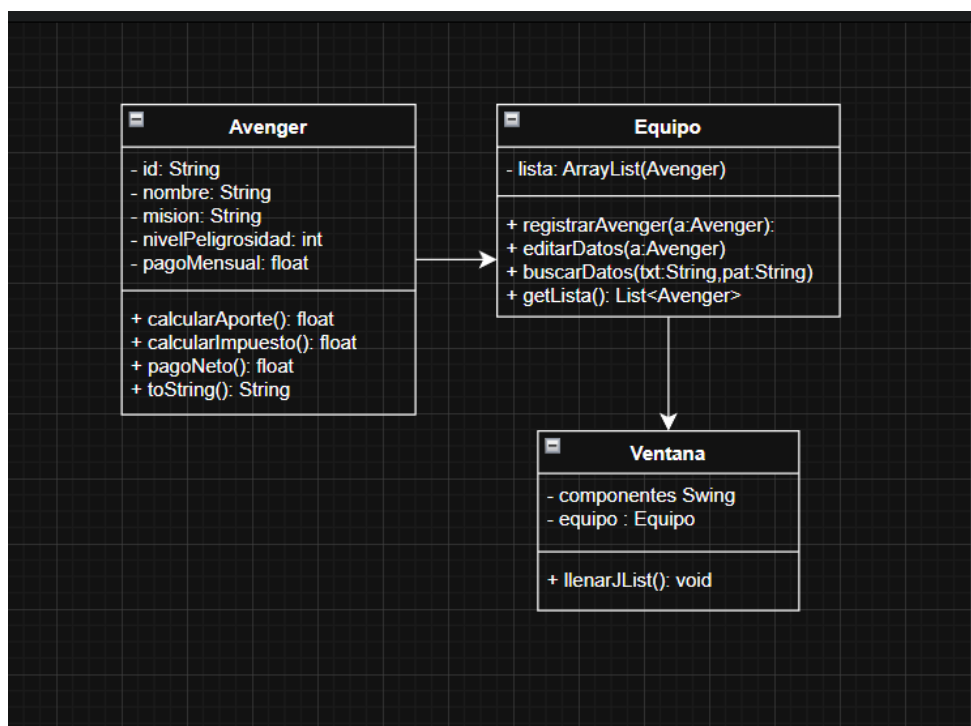
<b>Entrada</b>	<ul style="list-style-type: none"> <li>• Id</li> <li>• Nombre</li> <li>• Misión</li> <li>• Nivel de peligrosidad</li> <li>• Pago mensual</li> </ul>
<b>Proceso</b>	<ul style="list-style-type: none"> <li>• Calcular aporte al fondo de héroes</li> <li>• Impuesto anual</li> <li>• Pago neto mensual</li> </ul>
<b>Salida</b>	<ul style="list-style-type: none"> <li>• Listado de Avengers</li> <li>• Reporte</li> </ul>

## Alternativas de solución

### Solucion 1:

La primera alternativa propone utilizar un ArrayList de objetos Avenger, aplicando el algoritmo Naive String Matching para realizar búsquedas por ID. Este enfoque resulta especialmente útil cuando se necesita localizar elementos sin requerir que el usuario escriba el identificador completo, ya que el algoritmo compara secuencialmente el patrón de búsqueda con cada elemento de la lista. Como señalan Cormen et al. (2022), los algoritmos de fuerza bruta son apropiados para colecciones de tamaño reducido, ya que su complejidad  $O(nm)$  no representa un problema significativo en listas pequeñas. Además, su capacidad para manejar IDs alfanuméricos y permitir búsquedas parciales lo convierte en una solución práctica dentro del contexto del sistema de gestión de misiones de los Avengers.

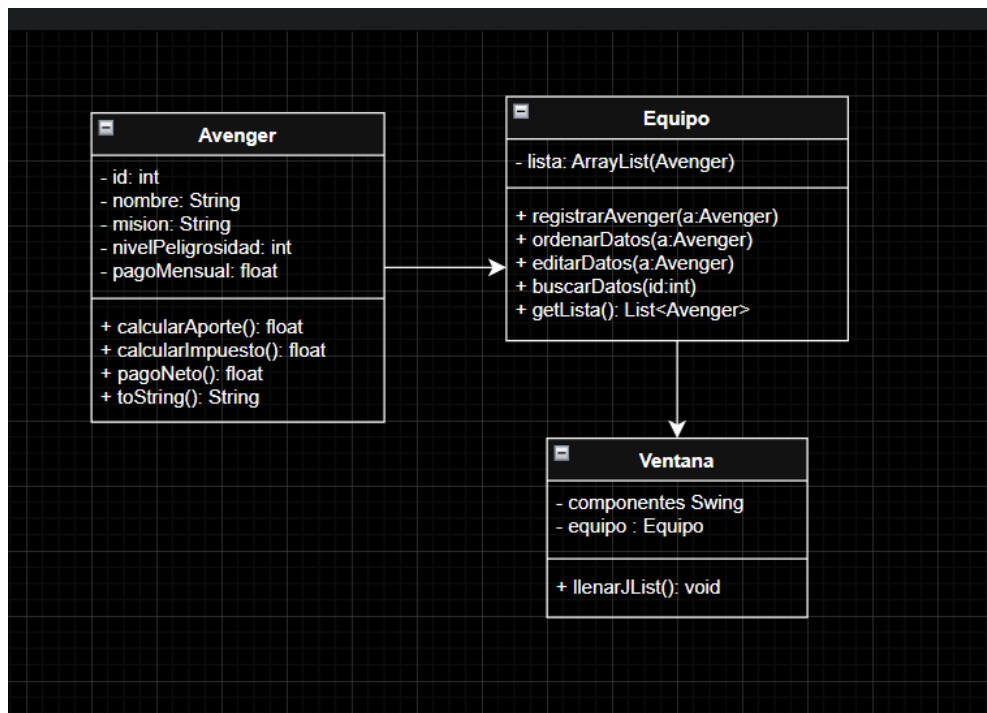
Diagrama de clases:



## Solucion 2:

La segunda alternativa consiste en mantener un ArrayList ordenado y aplicar búsqueda binaria para optimizar el tiempo de consulta, reduciendo la complejidad a  $O(\log n)$ . Este método divide repetidamente la lista hasta localizar el elemento buscado, aumentando la eficiencia en comparación con la búsqueda lineal. Sin embargo, tal como explican Sedgewick y Wayne (2011), la búsqueda binaria requiere que la colección esté estrictamente ordenada y que los elementos puedan compararse de manera consistente. Esto introduce ciertas limitaciones: los datos deben ingresarse en orden creciente o la lista deberá reorganizarse cada vez que se añade un nuevo Avenger. Aunque esta propuesta es más eficiente desde el punto de vista algorítmico, también demanda un mayor control en la entrada de información y una estructura más rígida, al enfatizar la importancia de mantener la coherencia en colecciones ordenadas para evitar errores lógicos.

Diagrama de clases:



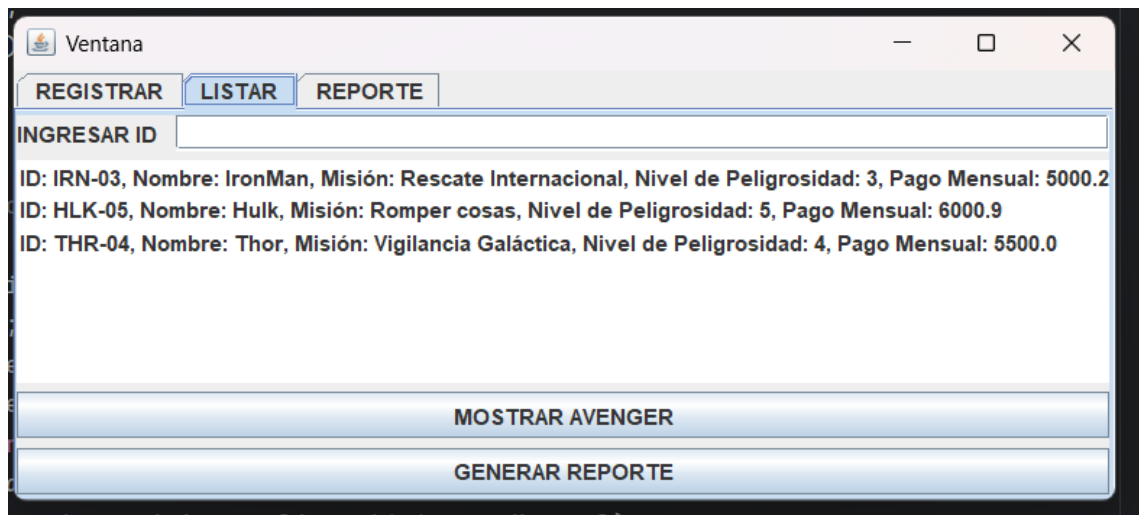
## Selección de la solución:

Tras analizar ambas propuestas, la alternativa seleccionada es la primera: el uso de un ArrayList con búsqueda mediante Naive String Matching. Esto se debe a que el sistema de gestión de los Avengers no maneja grandes volúmenes de datos y, por tanto, no requiere algoritmos de búsqueda altamente optimizados como la búsqueda binaria. Además, la flexibilidad para permitir IDs alfanuméricos y realizar búsquedas parciales mejora significativamente la experiencia del usuario, eliminando la necesidad de imponer restricciones estrictas sobre el formato y el orden de los datos. Knuth (1999) también resalta la importancia de elegir algoritmos

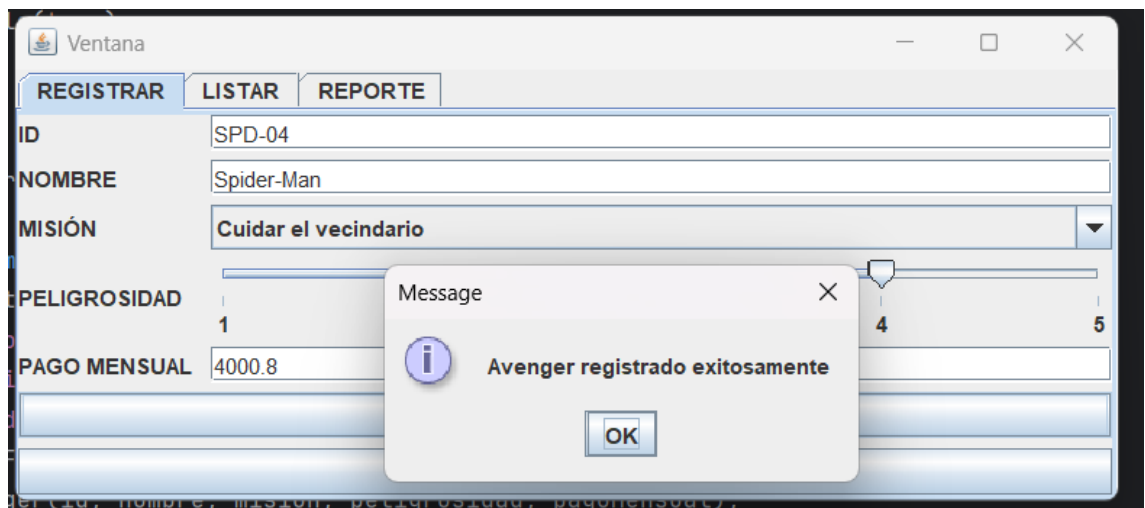
adecuados al tamaño y características de los datos, evitando optimizaciones innecesarias que complican la implementación. En pruebas realizadas durante el desarrollo del proyecto, esta alternativa demostró ser suficiente, intuitiva y de rápida implementación, facilitando la consulta de los Avengers de manera eficiente.

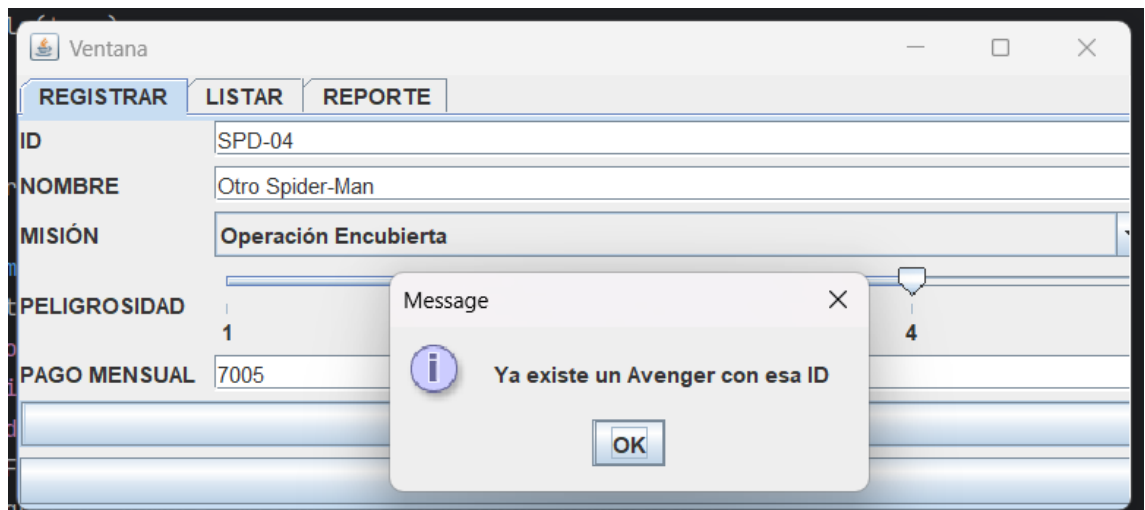
#### Análisis de resultados y casos de prueba:

- La lista de Avengers predefinida se despliega correctamente en la interfaz gráfica al iniciar la aplicación.

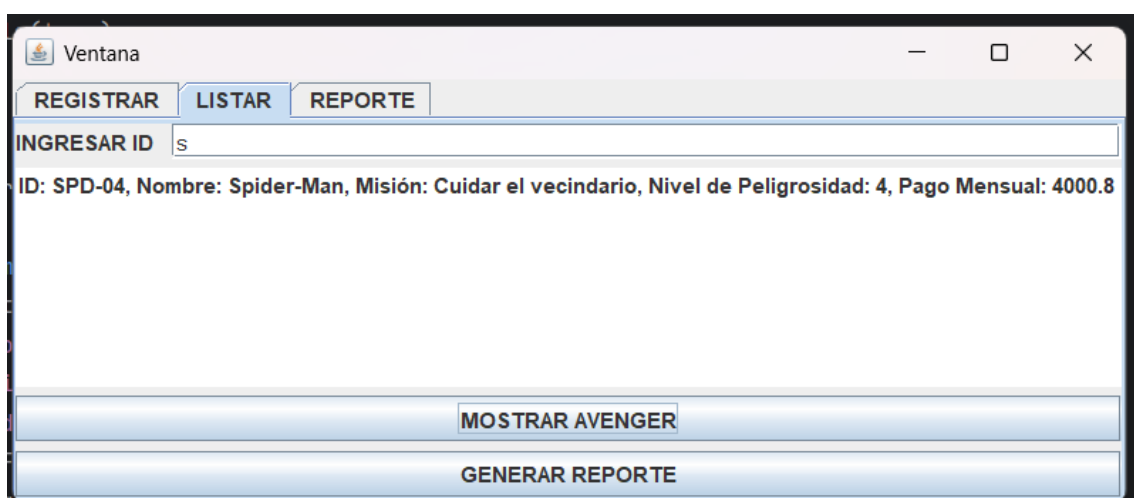


- El registro de nuevos Avengers es correcto y se evitan los duplicados, al intentar agregar un Avenger con ID existente, se muestra un mensaje de alerta.

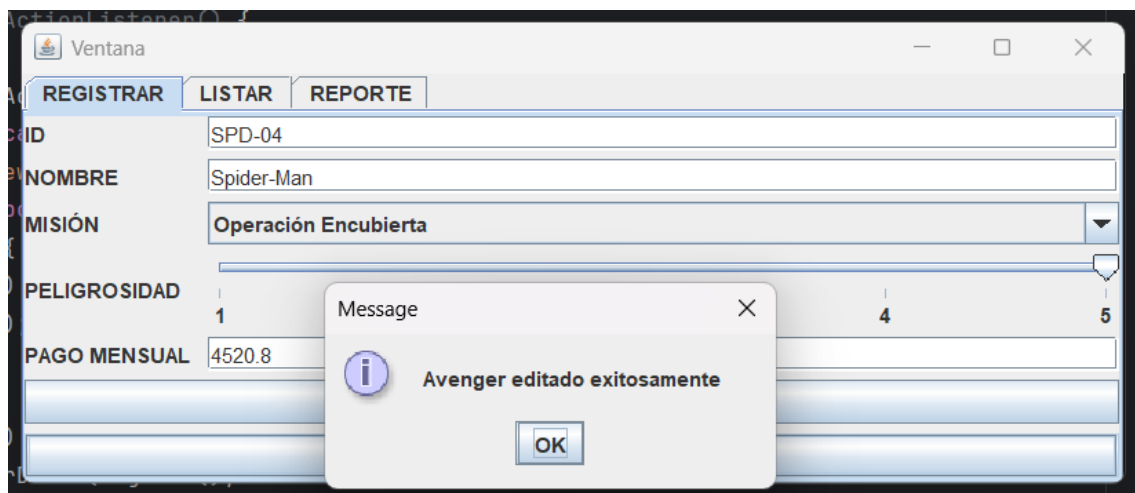
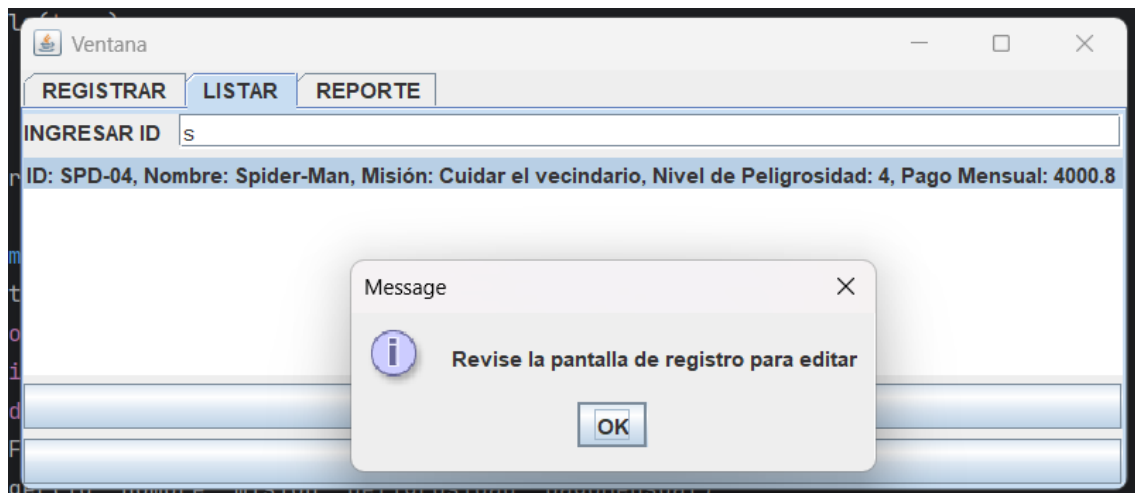




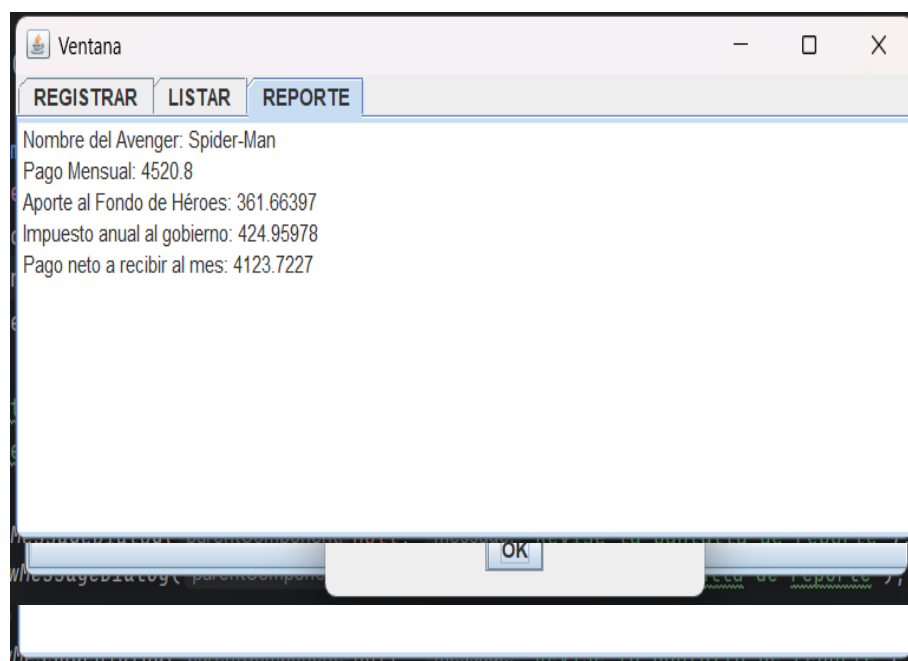
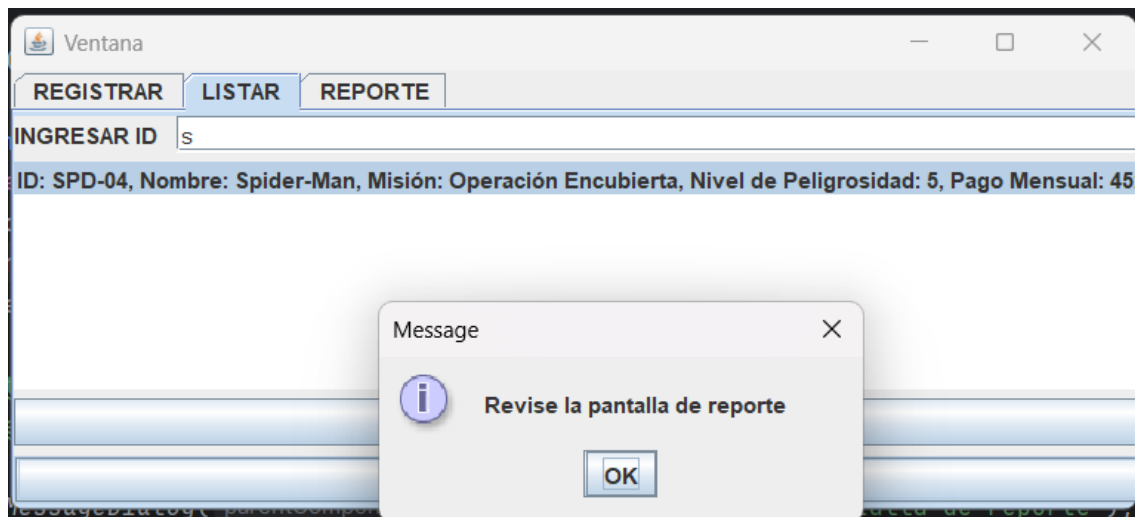
- La búsqueda parcial por ID funciona con el algoritmo Naive, no es necesario ingresar el ID completo para encontrar un héroe



- La edición de datos de un Avenger actualiza correctamente nombre, misión, nivel de peligrosidad y pago mensual.



- La generación de reportes muestra todos los datos relevantes de cada Avenger, incluyendo pago mensual, aporte, impuesto y pago neto.



## Referencias:

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

[https://books.google.com.ec/books?hl=en&lr=&id=RSMuEAAQBAJ&oi=fnd&pg=PR13&dq=Introduction+to+Algorithms.+cormen&ots=a3m3-Z3HRL&sig=BvSNrJCDJI5gVJZM7DT5AmKFpr4&redir\\_esc=y#v=onepage&q=Introduction%20to%20Algorithms.%20cormen&f=false](https://books.google.com.ec/books?hl=en&lr=&id=RSMuEAAQBAJ&oi=fnd&pg=PR13&dq=Introduction+to+Algorithms.+cormen&ots=a3m3-Z3HRL&sig=BvSNrJCDJI5gVJZM7DT5AmKFpr4&redir_esc=y#v=onepage&q=Introduction%20to%20Algorithms.%20cormen&f=false)

Sedgewick, R., & Wayne, K. (2011). *Algorithms*. Addison-wesley professional.

[https://books.google.com.ec/books?hl=en&lr=&id=MTpsAQAAQBAJ&oi=fnd&pg=PR6&dq=Algorithms.+Princeton+University&ots=QhpDFPHahZ&sig=kHLZt0C47gmrMtAiZ2rCKBbKPzk&redir\\_esc=y#v=onepage&q=Algorithms.%20Princeton%20University&f=false](https://books.google.com.ec/books?hl=en&lr=&id=MTpsAQAAQBAJ&oi=fnd&pg=PR6&dq=Algorithms.+Princeton+University&ots=QhpDFPHahZ&sig=kHLZt0C47gmrMtAiZ2rCKBbKPzk&redir_esc=y#v=onepage&q=Algorithms.%20Princeton%20University&f=false)

Knuth, D. (1999). The art of computer programming. *Sorting and searching*, 3(426-458), 4. <https://www.ioccc.org/2019/ciura/fasc5c.pdf>