Software de Apoyo para Diagnóstico Diferencial

David Pedraza, Sergio Pardo, Aldo Bornacelly, Nicolás Valderrama

No. de Equipo Trabajo: 4

I. INTRODUCCIÓN

Aquí una introducción somera a nuestro proyecto: una herramienta de diagnóstico diferencial para usos médicos y académicos, mejorando así la salubridad y apoyo médico en áreas rurales Colombianas.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

En el ejercicio de la medicina existe una gran variedad de diagnósticos posibles para una colección relativamente pequeña de síntomas. En general, el entrenamiento médico y la experiencia sugieren la asociación de síntomas comunes con enfermedades comunes. En ocasiones, el no tener en cuenta condiciones mucho más exóticas puede tener consecuencias terribles e incluso mortales. Nuestro objetivo es desarrollar una herramienta de apoyo que sugiera, no sólo lo que el hábito propone al diagnosticar a un paciente, sino las posibilidades menos obvias.

III. USUARIOS DEL PRODUCTO DE SOFTWARE.

Hay conflictos de carácter ético y práctico a la hora de dejar estas utilidades a disposición del público. Por ese motivo, nuestro enfoque está dirigido a profesionales de la salud, tanto en el ámbito profesional como académico. Como se ha descrito ya; este software es una herramienta de apoyo, no un reemplazo para los profesionales médicos.

Por otro lado, es importante para el uso correcto de la herramienta conocer distinciones sutiles, pero importantes, entre conceptos como “síntoma” y “signo”. Esta razón, junto con la anterior, confinan nuestros esfuerzos hacia el marco del ejercicio médico profesional.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

Para el funcionamiento del programa se necesita el ingresar síntomas y signos presentados por el paciente. Pueden ser ingresados directamente en la ventanilla principal separados por comas. Luego de buscar por estos síntomas y signos se realiza una búsqueda en la base de datos con el propósito de encontrar posibles condiciones que se asemejan a los datos ingresados.
Los datos de la base están sorteados en orden lexicográfico con el propósito de facilitar la búsqueda. Sólo los desarrolladores pueden ingresar nuevos datos a la base de datos, al menos por el momento.

- ***Búsqueda de Síntomas y Signos***
  - Esta funcionalidad permite buscar por síntomas y signos dentro de la aplicación.
- ***Acciones iniciadoras y comportamiento esperado***
  - ❖ Luego de ingresar los síntomas y signos que se desea cotejar, el programa despliega un catálogo de posibles dolencias.
  - ❖ Se espera una lista de condiciones que intercepten sus síntomas y signos característicos con los ingresados por el usuario.

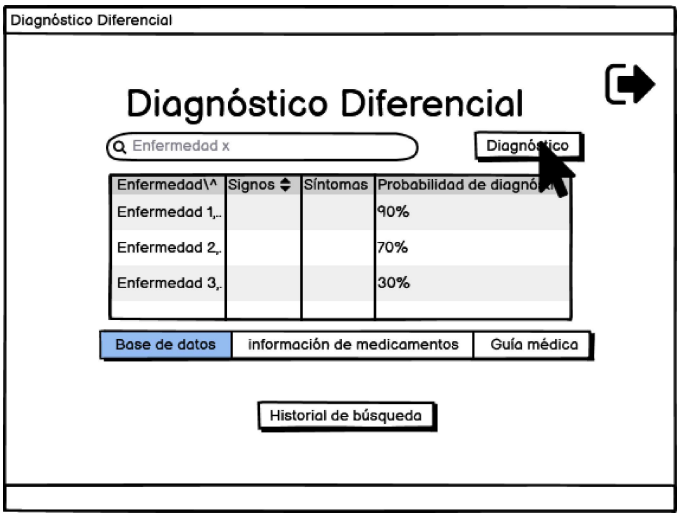
Diagnosticar enfermedad	
Precondición	El usuario ha ingresado la información adecuada para realizar el proceso de cotejo.
Datos esperados	Se realiza el anàlisis y la comparaciòn a partir de los datos ingresados.
Despliegue de posibilidades	La interfaz mostrará las enfermedades junto con las posibilidades de ser dependiendo de los síntomas ingresados.
Diagnosticar	A partir de nuestra herramienta, el personal de salud encargado tomará su proxima desiciòn a partir del apoyo e informaciòn dada por la aplicaciòn.

V. DESCRIPCIÓN DE LA INTERFAZ DE USUARIO PRELIMINAR

La interfaz gráfica de usuario cumplirá con el objetivo de proporcionar un entorno visual sencillo, que permita la más eficiente comunicación entre el personal de la salud y las principales funcionalidades que provee el programa.

La ventana principal contará con una caja de búsqueda, que le permitirá al usuario ingresar la información específica de los signos y síntomas que aquejan al paciente. En futuras entregas se planea añadir botones que permitan obtener información de medicamentos y guías médicas, tal y como se muestra a continuación.

Al pulsar el botón de “Diagnóstico” se dará inicio a la búsqueda de las tres enfermedades que más se ajusten al conjunto de información ingresada. Cada vez que se ejecute una búsqueda se desplegará una tabla con las enfermedades con mayores probabilidades de diagnóstico.



Otra de las funcionalidades que se planea introducir en un futuro es la pestaña de historial, esta herramienta permitirá recuperar la información sobre las búsquedas que se hayan realizado.

VI. ENTORNOS DE DESARROLLO Y OPERACIÒN

Teniendo en cuenta que el desarrollo del proyecto no solo es realizado por una persona, se presentan distintos entornos de desarrollo los cuales son: Visual Studio Code, IntelliJ y Netbeans para la creación de la interfaz. Este desarrollo es multiplataforma, ya que ha sido desarrollado en el lenguaje java, está planeado para ser ejecutado en dispositivos Windows, Linux y MacOs, posteriormente está planeado para estar disponible en dispositivos android. Sin embargo también se hizo presente un bajo pero importante porcentaje del uso de Python.

VII. PROTOTIPO DE SOFTWARE INICIAL

Para esta entrega tenemos una interfaz funcional, respecto al diagnóstico diferencial mostrando un prototipo con una base de datos, información de medicamentos, una guía médica y por ultima la oportunidad de realizar un diagnóstico. Dando de esta manera un abrebocas al proyecto final dejando bien planteado en el prototipo cual es nuestro objetivo con lo mostrado.

La realización y unificación del prototipo del proyecto fue desarrollado en el repositorio de software de Github llamado Diagnóstico Diferencial el cual fue compartido no solo con los participantes y desarrolladores del proyecto, sino que también con el profesor de la materia el cual está al tanto del avance grupal del trabajo.

La versión gráfica del trabajo en el repositorio está dada por:

```
.
├── diagnosticoDiferencial.iml
├── enfermedadPaciente.txt
├── generador_datos.py
├── muestra2.txt
├── muestra.txt
├── src
│   ├── diagnosisTools
│   │   ├── diagnosticoArrays.java
│   │   └── enfermedad.java
│   └── main
│       ├── dataReader.java
│       ├── history.java
│       ├── main.java
│       └── Window.java
├── structures
│   ├── DNode.java
│   ├── doubleLinkedList.java
│   ├── DoubleList.java
│   ├── doubleNode.java
│   ├── dynamicArray.java
│   ├── queue.java
│   ├── simpleLinkedList.java
│   ├── simpleNode.java
│   ├── Stack.java
│   ├── stringDoubleLinkedList.java
│   └── stringDoubleNode.java
└── tree.txt
```

VIII. IMPLEMENTACIÓN Y APLICACIÓN DE LAS ESTRUCTURAS DE DATOS

El programa se apoya especialmente en las cualidades que provee la implementación de una lista doblemente encadenada. Esta estructura es responsable de contener, de manera temporal, el conjunto de información de cada enfermedad almacenada en la base de datos. La estructura cuenta con referencias tanto del primer como del último nodo y posee las operaciones básicas de cualquier lista de datos. El rol de este tipo de listas es fundamental para la funcionalidad del diagnóstico, ya que con sus operaciones, y con la ayuda de un algoritmo de ordenamiento, se pueden comparar los datos ingresados con los almacenados, con el fin de encontrar similitudes.

En cuanto al uso de estructuras unidimensionales se refiere, implementamos un array dinámico para introducir la funcionalidad del " historial de búsqueda". Esta estructura nos permite almacenar la información relacionada con las búsquedas de diagnóstico realizadas. Debido a que esta es una funcionalidad secundaria, no se realizaron pruebas de tiempo.

IX. PRUEBAS DEL PROTOTIPO Y ANÁLISIS COMPARATIVO

A Continuación mostraremos la comparativa de algunas pruebas del prototipo para mostrar la funcionalidad del mismo:
La primera funcionalidad es la carga de datos a partir de un archivo de texto. Estos datos pueden ser engañosos, ya que cada datos representa un objeto compuesto por 3 cadenas adicionales, una en la que se almacena el nombre de la enfermedad, otra para signos y la última para los síntomas. Haciendo que la lectura (Y cada operación que se haga posteriormente) de cada dato varíe tanto en tamaño como en tiempo. Esta operación consta de un ciclo que se ejecutará n veces, procesando las 3 líneas adicionales de cada dato. Estas líneas tienen tamaños despreciables (de 20 a 50 palabras), por lo que no han sido tenidas en cuenta en nuestro análisis asintótico.

CARGA DE DATOS O(n)		
n	Tiempo en ns	Tiempo en ms
1000	51718419	51
10000	245627926	245
100000	1183864110	1183

La siguiente operación es la ordenación de una lista enlazada, proceso que se hace frecuentemente y es indispensable para el funcionamiento del programa. El ordenamiento de la lista se consiguió mediante la implementación del algoritmo merge sort, u ordenamiento por mezcla, cuya complejidad es n Log n.

Ordenar Lista Enlazada (de objetos) O(nLogn)		
n	Tiempo en ns	Tiempo en ms
1000	976205	0.9
10000	9318237	9
100000	31555991	31

Posteriormente se realiza el diagnóstico, operación que se hace al encontrar la intersección entre dos listas enlazadas, esta operación es confusa a primera vista, ya que se realizará n veces, y encontrar la intersección de las listas, funciona también en complejidad lineal, pero esta segunda complejidad dependerá de la cantidad de signos y síntomas de las enfermedades y este parámetro está aleatorizado. Además, se hacen comparaciones lexicográficas usando la librería estándar de java. Es por esto que medir esta complejidad mediante la notación O grande, podría servir como guía, pero para mayor precisión podría hacerse uso de otras herramientas como el análisis amortizado.

Diagnóstico O(n)		
n	Tiempo en ns	Tiempo en ms
1000	19935151	19
10000	673754423	673
100000	oo	oo

X. ROLES Y ACTIVIDADES.

A continuación una breve descripción de los toles que, durante esta entrega, han estado presentes en nuestro grupo de trabajo.

Durante el curso de las siguientes entregas esperamos la diversificación de los roles con el propósito de cubrir a cabalidad el crecimiento integral de todos los involucrados en este proyecto.

INTEGRANTE	ROL(ES)	ACTIVIDADES REALIZADAS (Listado)
Nicolás Ricardo Valderrama	Líder	Ha sido instrumental en esculpir la visión del proyecto.
		Su trabajo en la plataforma Github ha sido diligente y constante.
	Técnico	
	Investigador	Organizó una reunión con estudiantes de medicina con el propósito de informar la envergadura del problema.
Aldo Bornacelly Navarro	Coordinador	Sus constantes actualizaciones con respecto al progreso del software han sido indispensables para la finalización de esta entrega.
	Técnico	
	Experto	Ha concentrado sus esfuerzos en la implementación de estructuras de datos necesarias para el funcionamiento correcto del producto.
Sergio David Lopez Pardo	Observador	Se ha encargado de describir los aportes del proyecto de forma sencilla con el propósito de finalizar el informe.
David Ricardo Pedraza Silva	Animador	Enfatiza un ambiente de camaradería y respeto por medio de una comunicación constante.

XI. DIFICULTADES Y LECCIONES APRENDIDAS

Han existido dificultades en sincronizar las necesidades de tiempo de cada uno a la hoa de congregar nuestros esfuerzos con el propósito llevar el proyecto a plenitud.

Entendemos, sin embargo, que la clave está en una comunicación constante y honesta, para garantizar que las responsabilidades de todos y cada uno de los miembros de este equipo sean satisfechas.

La principal lección ha sido establecer la prioridad necesaria para nuestra tarea actual.

XII. REFERENCIAS BIBLIOGRÁFICAS