Universidad Rafael Landívar

Ingeniería en informática y sistemas

Estructuras de datos I

Ing. Pablo Godoy

**PROYECTO**

**“COVID-19”**

|  |  |
| --- | --- |
| José Vinicio De León Jiménez | 1072619 |
| Javier Andrés Morales | 1210219 |

Guatemala, 9 de mayo de 2020

**INTRODUCCIÓN**

El proyecto final de la clase de Estructura de Datos I consistió en hacer una implementación eficiente de la administración de un sistema de registro, contabilización y manejo de pacientes ante la crisis actual generada por la propagación del COVID-19 en Guatemala.

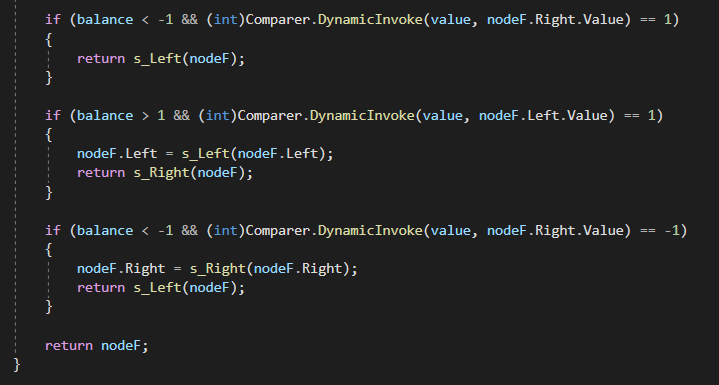
El proyecto consta de distintas estructuras de datos lineales y no lineales que trabajan en conjunto para dar acceso y administrar los datos de una forma robusta y eficiente ante cada modificación. Las principales estructuras utilizadas fueron un árbol AVL para una búsqueda de pacientes eficiente, un montículo o Heap para la atención de personas sospechosas o confirmadas de COVID-19 y que el criterio de orden de atención dependiera de la prioridad y fecha de ingreso del paciente, por otra parte, se implementó el uso de un Dictionary artesanal para el almacenamiento de las personas en una posición enumerable como abstracción de su posición en la cama del hospital específico.

El uso de estas estructuras y sus características especiales al igual que los delegados permitieron manejar los datos de manera genérica y orientada a objetos permitiendo su escalabilidad a un ingreso de datos mayor.

**MÉTODOS MÁS IMPORTANTES**

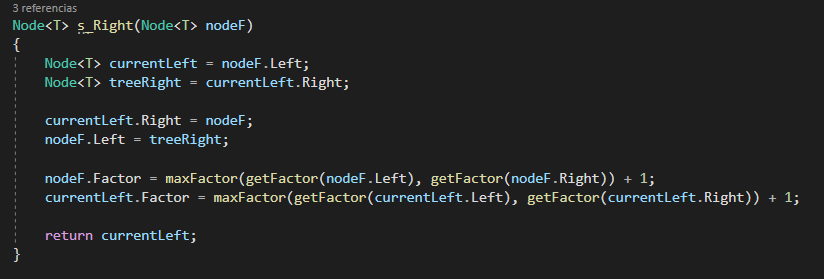
* **Inserción en el árbol**

En este método recursivo recibe la raíz y el valor a insertar en el árbol hace las comparaciones de la llave para determinar la posición a almacenar en el árbol si inserta a la izquierda o derecha y actualiza la altura del nodo cada movimiento, al existir mayor peso en uno de los subárboles realiza una rotación a la derecha o a la izquierda de cada nodo o subárbol o incluso ambas dependiendo de los factores de equilibrio, luego va subiendo de nivel saliéndose de la recursividad y finalmente devuelve la raíz con todas sus ramas balanceadas.



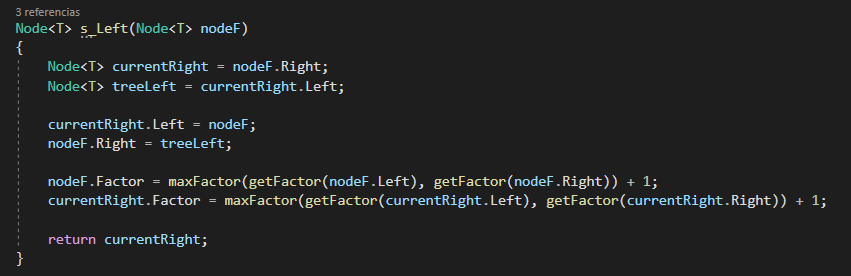
* **Rotación simple derecha**

Esta rotación implica un mayor peso o mayor cantidad de nodos insertados en el subárbol izquierdo, por lo tanto, el padre pasa a ser el hijo derecho de su primer hijo izquierdo, y el anterior hijo derecho de este hijo izquierdo pasa a ser el nuevo hijo izquierdo del anterior nodo padre es decir el nodo que se desplazo a la derecha. Todo esto sin afectar la referencia al nodo izquierdo del nuevo nodo padre, es decir el anterior hijo izquierdo, al hacer la nueva asignación de nodos actualiza el peso de dicha rama y finalmente devuelve el nuevo nodo padre.



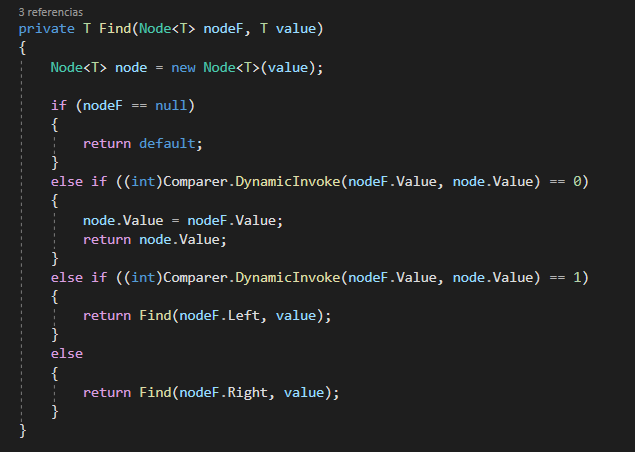
* **Rotación simple izquierda**

Esta rotación implica un mayor peso o mayor cantidad de nodos insertados en el subárbol derecho, por lo tanto, el padre pasa a ser el hijo izquierdo de su primer hijo derecho, y el anterior hijo izquierdo de este hijo derecho pasa a ser el nuevo hijo derecho del anterior nodo padre es decir el nodo que se desplazó a la izquierda. Todo esto sin afectar la referencia al nodo derecho del nuevo nodo padre, es decir el anterior hijo derecho, al hacer la nueva asignación de nodos actualiza el peso de dicha rama y finalmente devuelve el nuevo nodo padre.



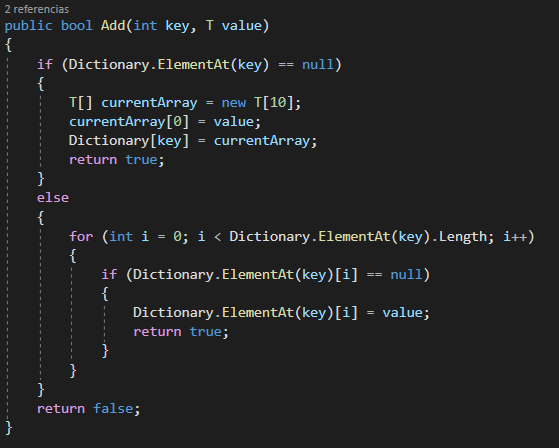
* **Búsqueda en el árbol**

Nuevamente se hace uso de la recursividad para moverse entre los nodos del árbol y se mueve hacia la izquierda o derecha dependiendo de la comparación de la llave, al encontrar que el valor de la llave es igual devuelve el nodo, de encontrar no encontrar un valor devolverá un nodo null.



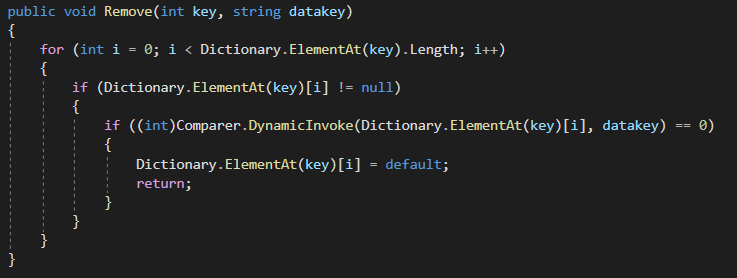
* **Inserción en el Dictionary artesanal**

Este método consiste en acceder a la posición especificada por la llave previamente convertida y realizar un recorrido lineal a través de las posiciones del arreglo interno, al encontrar una posición vacía se almacena el dato y se sale del método devolviendo un bool true.



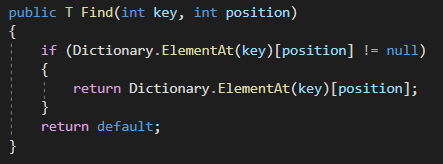
**Eliminación en el Dictionary artesanal**

Este método consiste en acceder a la posición especificada por la llave previamente convertida y realizar un recorrido lineal a través de las posiciones del arreglo interno, al encontrar al encontrar la posición con la llave única del dato especificado se cambia su valor en dicha posición por null.



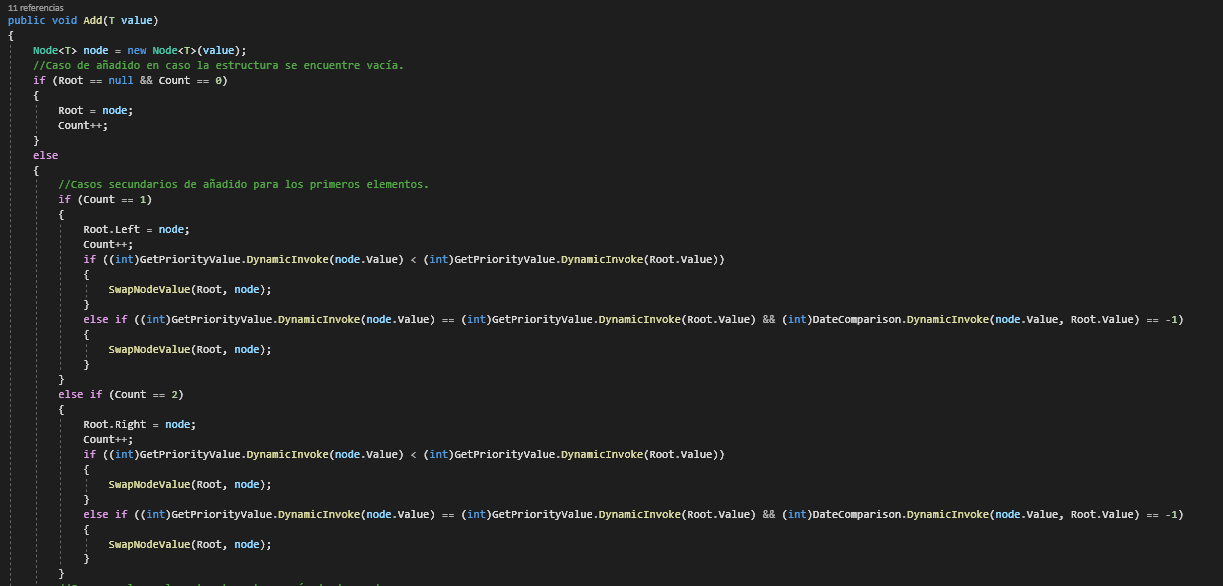
* **Búsqueda en el Dictionary artesanal**

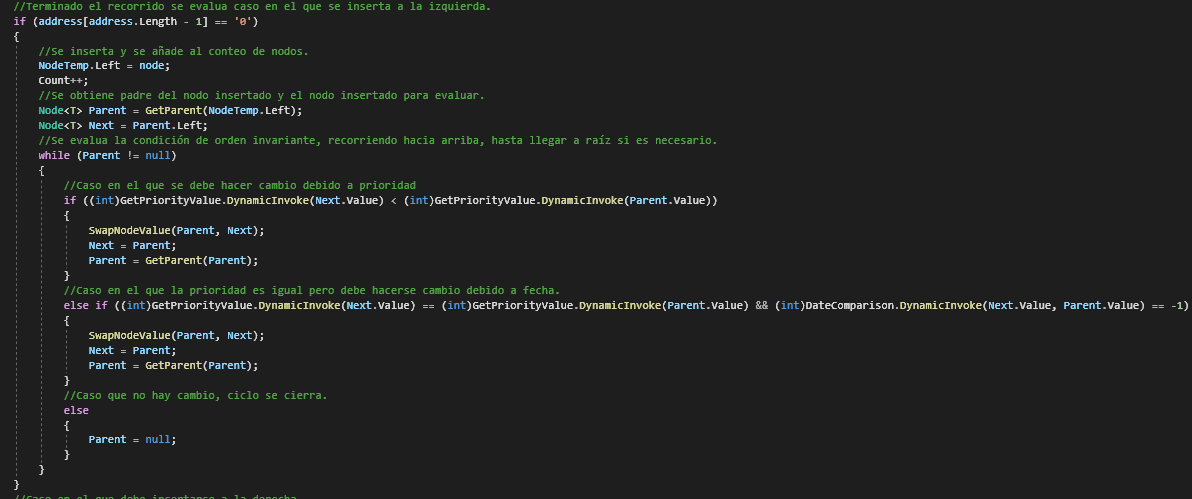
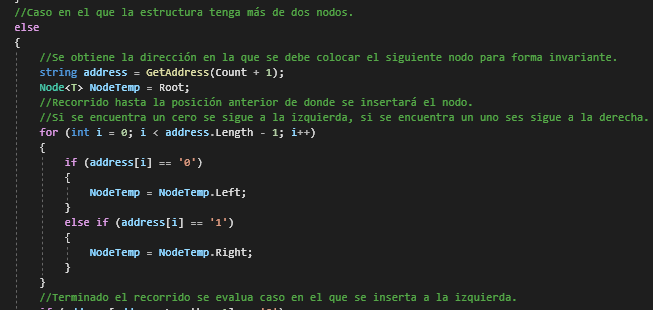
Devuelve el dato contenido en la posición especificada por la llave del dato en la lista como su posición en el arreglo, esto sirve como la abstracción de una posición definida en una fila específica, en este caso cada fila representa un hospital y cada casilla una cama.



* **Inserción en el Heap**

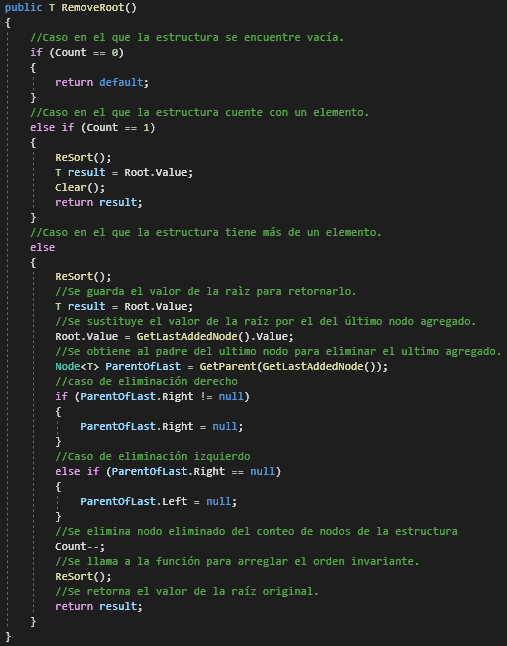
Este método obtiene la “dirección” en la que se insertará el valor que la función recibe como parámetro, la cual es la cantidad de nodos +1. Se recorre el string donde está almacenada la dirección, si el carácter es 0 se continúa a la izquierda y si este es 1 a la derecha, al llegar al espacio designado según la forma invariante se inserta y se procede a realizar comprobaciones para cumplir con el orden invariante con el padre del nodo insertado, si el nodo tiene mayor prioridad que su padre se intercambian los valores y se continua verificando hacia arriba, si el nodo tiene igual prioridad se comparan las fechas de ingreso del paciente, si la fecha del hijo es previa a la del padre se intercambian los valores y se continua verificando hacia arriba, en caso contrario se deja de comprobar y los valores se quedan en su posición.





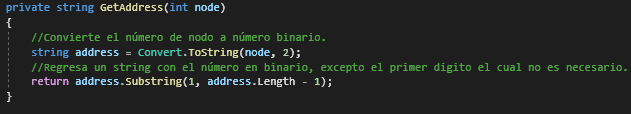
* **Eliminación en el Heap**

Método que consiste en extraer el valor de la raíz debido a que este es el elemento más prioritario, se recorre la estructura hasta llegar al ultimo nodo agregado, se intercambia el mismo con la raíz y se elimina el nodo original, finalmente se llama al método ReSort para conservar el orden invariante, este método evalua nodo por nodo desde la raíz hasta abajo, intercambiando cuando la prioridad de la raíz es menos prioritaria que la de un subárbol. Se retorna el valor de original de la raíz.



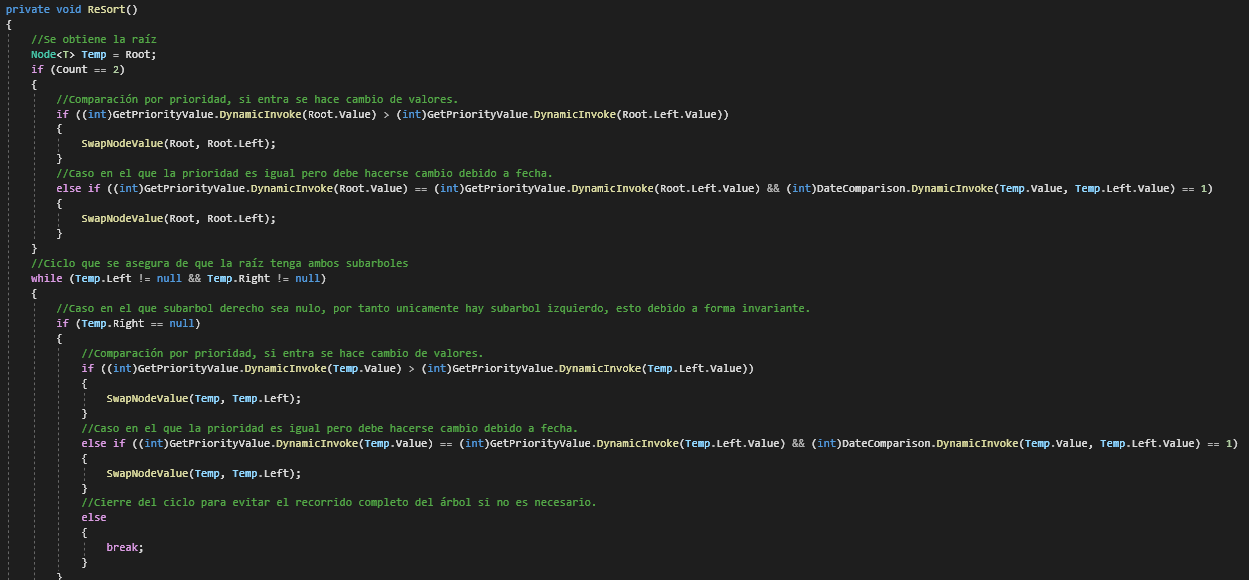
* **GetAddress en el Heap.**

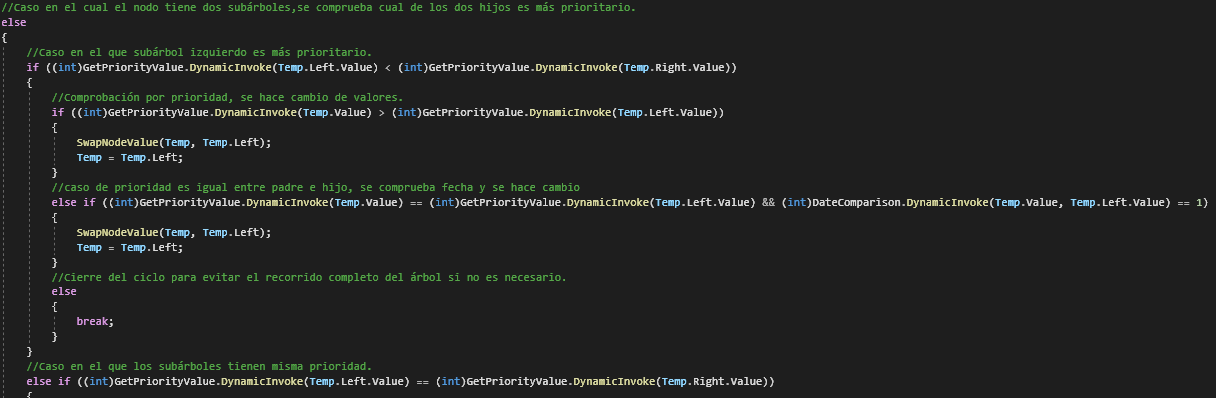
Método simple que recibe como parámetro un entero, este es convertido a número binario, del cual se retorna un substring del mismo, ignorando el primer carácter de la cadena de texto. Las indicaciones que da la cadena indican que si se encuentra un cero se debe continuar a la izquierda, en el caso de un uno, a la derecha.4



* **ReSort en el Heap.**

Método que inicia en la raíz de la estructura y la recorre hacia abajo para mantener el orden invariante, inicia comprobando si la raíz tiene ambos subárboles, de solo tener uno se continua hacia la izquierda, comprobando si la prioridad del padre es menor a la del subárbol, en ese caso se intercambian los nodos y se continua con este proceso, en el caso de tener prioridad igual, se verifica si la fecha de ingreso del paciente de la raíz es mayor a la del subarbol, en dicho caso se intercambian, en el caso de que existan ambos subárboles se comprueba cual tiene mayor prioridad y se continua por ese, se realiza el proceso anterior hasta que la condición se cumpla o al llegar a una hoja.

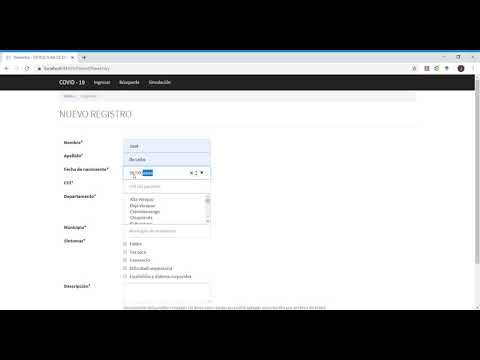




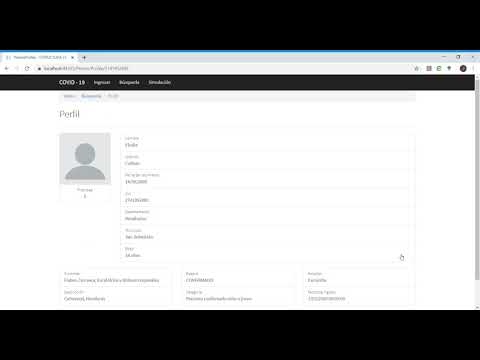
**COVID-19**

**MANUAL DE USUARIO**

**INGRESO**

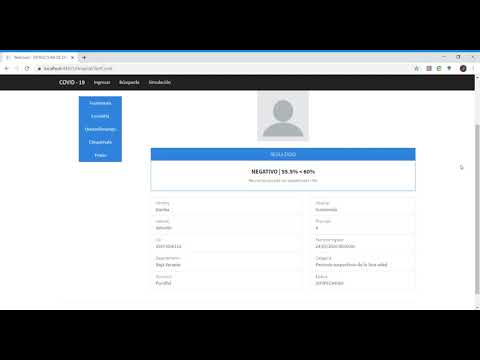
[](https://www.youtube.com/embed/WMU3Ddwnzg0?feature=oembed)

**BUSQUEDA**

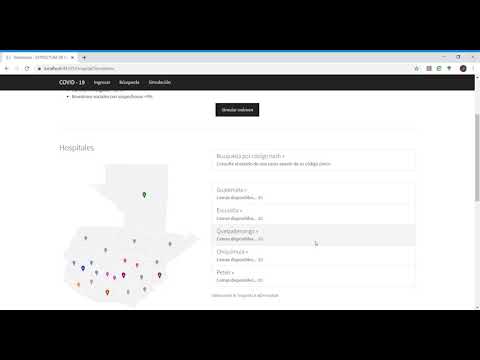
[](https://www.youtube.com/embed/T13lI9X3VIo?feature=oembed)

**SIMULACIÓN**

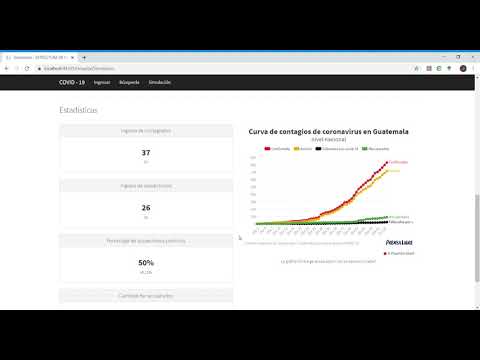
* **PRUEBA DE COVID-19**

[](https://www.youtube.com/embed/95cDNVTjVGQ?feature=oembed)

* **ADMINISTRACIÓN DE HOSPITALES**

[](https://www.youtube.com/embed/-HVhrse2Ekk?feature=oembed)

* **ESTADÍSTICAS**

[](https://www.youtube.com/embed/IEsyXsAP960?feature=oembed)