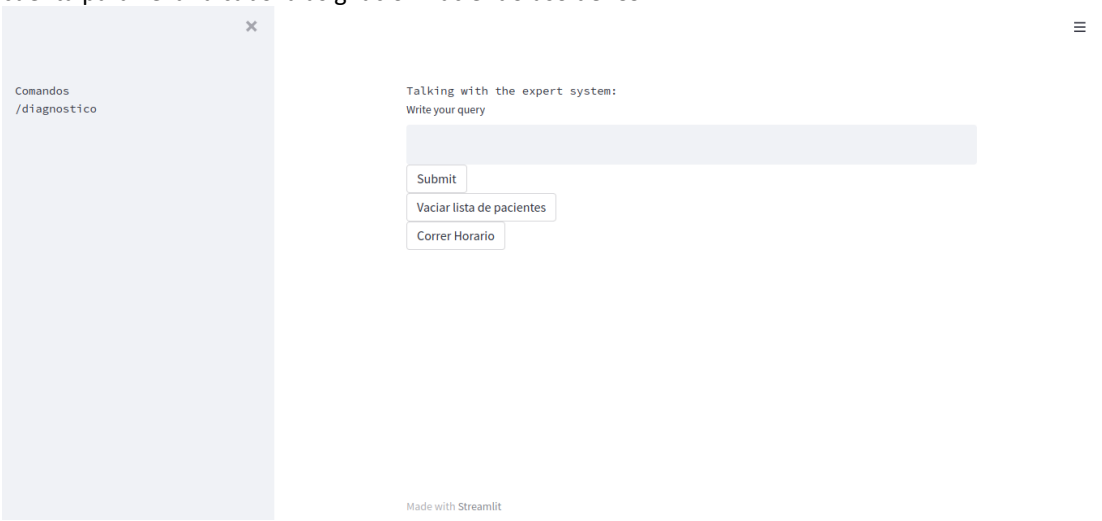
Definición del problema
Nuestro problema consiste en facilitar la interacción con los usuarios en un hospital, así como la planificación del horario del hospital para garantizar una buena atención a todos los clientes y que los mismos estén satisfechos.

Funcionamiento general de la aplicación
Para la realización de nuestro proyecto se llevó a cabo la implementación de un sistema experto basado en reglas que responde a las interrogantes en lenguaje natural y un algoritmo de satisfacción de restricciones CSP para simular la asignación de cada paciente a su respectiva consulta o terapia, así como la planificación que se llevaría a cabo en el hospital para atender a los pacientes.
Todo el código fue escrito en Python.

Para la realización del sistema experto se diseñó una interfaz gráfica mediante una aplicación web utilizando la biblioteca de streamlit. En esta se crea un campo para la entrada de texto, en el que se puede introducir alguna consulta o interrogante para que el sistema experto de respuesta y también el comando que permite la asignación de horarios a los pacientes y doctores y aparatos correspondientes a la consulta o terapia que se esté llevando a cabo. En la parte izquierda de la aplicación se muestra información sobre los comandos que pueden ser introducidos. También cuenta con 3 botones, uno para enviar el mensaje al sistema experto, otro para llevar a cabo la asignación y otro para vaciar la lista de los pacientes que se han introducido hasta el momento y que se tendrán en cuenta para llevar a cabo la asignación haciendo uso del CSP.



Para dar respuesta a las interrogantes introducidas en lenguaje natural, se pasa por 2 fases:
- Fase de Procesamiento del Lenguaje Natural a la consulta.
- Inferir respuesta utilizando PyKE

Fase de Procesamiento del Lenguaje Natural a la consulta.
Para llevar a cabo esta fase se realizaron las siguientes transformaciones al texto:
-Segmentación de oraciones. Se espera que el usuario pueda introducir un párrafo con varias ideas, se separan cada oración o ideas mediante el uso de la biblioteca nltk.
-Tokenización y Lemmatización de cada palabra en cada oración y asignación de etiquetas, para definir su función en la oración: POSTag. Se realiza con spacy.
-Se filtran las palabras que nos interesan mediante reglas teniendo en cuenta la función que cumplen en la oración.
-Se separan las ideas tomando ciertos bloques de cada oración compuesto por cierto número de palabras, en su mayoría 1,2 o 3 palabras. Llamaremos a estos bloques chunks. Esta separación de las ideas nos servirá para buscar relaciones entre 2 o más ideas en el proceso de inferencia con PyKE.

El código correspondiente a esta parte del procesamiento de la consulta se encuentra en la clase PreprocessText del archivo preprocess.py.

```python
import nltk, spacy

import examples.system.driver as driver
nlp = spacy.load('es_core_news_md')

@staticmethod
def sentence_segmentation(document: str, language="spanish")->list:
    """Dado un texto con cierta cantidad de oraciones, devuelve una lista d
    return nltk.sent_tokenize(document, language="spanish")

@staticmethod
def tokenize_and_tag_sent(sent:str):
    """Dada una oracion (sent:str) Devuelve una lista de tuplas de la forma
    document = nlp(sent)

    return [(token.text, token.lemma_, token.pos_) for token in document]
```

Fase de Inferir respuesta utilizando PyKE

En lo adelante nos referiremos a Procesamiento del Lenguaje Natural como NLP.

Para facilitarnos el trabajo en cuanto a las relaciones existentes entre las palabras o ideas, se crearon categorías, estas, a su vez están relacionadas y existe un verbo que describe dicha relación. También se definieron los términos, los cuales pertenecen a una categoría y se pueden relacionar con otros términos que pertenezcan a categorías que se encuentren relacionada con la suya.

Se debe de tener en cuenta que cada palabra de la oración que se modifica a través de NLP representa una categoría, una relación entre 2 categorías o un término.

La biblioteca de Python llamada PyKE, nos suministra una gran herramienta para la realización de un sistema experto basado en reglas. Para su confección fue necesaria la creación de:

-Palabras claves a tener en cuenta que representan los nombres de las categorías: Se crea una relación llamada system.category_name(category, alias)

-Términos a los que pertenece cada categoría: Se crea una relación llamada

-Relaciones o funciones entre las entidades que se definan: Tienen la sintaxis de una función de python y se le pasan cierta cantidad de parámetros, que serán las palabras que se relacionen, Ej system.relation_cat_cat(category1, category2, relationship).

-Hechos: Dígase por hechos, relaciones que sean verdaderas para nuestra base de conocimientos. Se encuentran en el archivo system/pyke_utils/examples/system/system.kfb

-Reglas: Son ciertas restricciones aplicadas al camino que tomara el grafo de inferencia. Así, dadas ciertas informaciones sobre las palabras de entrada y los hechos que contiene nuestra base de conocimientos, se van aplicando ciertas restricciones en funcion de lo que se quiera buscar. Las reglas se encuentran en el archivo system/pyke_utils/examples/system/bc_system.krb.

Flujo del proceso para interrogantes introducidas en lenguaje natural.

Se introduce la consulta deseada por el usuario, siendo recibida en el archivo visual.py y se llama al método ask() perteneciente al archivo main.py. Dentro del método ask() se lleva a cabo el procesamiento de lenguaje natural de la consulta y cada oración del resultado de separar las oraciones o ideas del párrafo que se espera se pasan una a una al método get_answer del archivo driver.py y con la salida del misma se va conformando la respuesta que se le devolverá al usuario en la parte visual de la aplicación. Una vez dentro del método get_answer, se busca para palabra que conforma la oración, si es un nombre de categoría, una instancia de categoría, una relación o un término. En caso de ser un término, se verifica que este se encuentre en nuestros hechos de PyKE creados en el archivo system.kfb. Luego pasamos a relacionar cierta cantidad de palabras para crear nuestros llamados chunks. Disponemos de varios tipos de chunks:

Chunks de términos: Compuesto por 1 termino

Chunks de Categorías con Categorías: Compuesto por 2 nombres de categorías

Chunks de Categorías con Términos: Compuesto por 1 nombre de categoría y 1 término

Chunks de Categorías con Términos y Relaciones: Compuesto por 1 nombre de categoría, 1 término y 1 relación.

Se crearon clases para representar y contener información de los mismos.

```
class Chunk:
    def __init__(self) -> None:
        self.relationships = []
        self.chunk = ""
        self.text_chunk = ""
        self.chunk_type = ""
class CatChunk(Chunk):
    def __init__(self, chunk, cat_instances=None, cat_relationships=None) -> None:
        self.chunk = chunk
        self.chunk_type = "cat"
        self.instances = cat_instances
        self.relationships = self.instances
        self.text_chunk = chunk
```

Ejemplo de implementación de clase de chunk de categoría.

Luego, tenemos una lista con todos los chunks ordenados siguiendo el orden que tenían las palabras en la oración que se está analizando actualmente. Vamos iterando por cada Chunk que se ha ido conformando buscando si este se puede relacionar con con el actual y el siguiente, es decir, el chunk en la posición n se puede relacionar con los de la posición n-1 y n+1. Para estos buscamos los valores válidos obtenidos de los hechos a través de las reglas de PyKE. En caso de existir relación entre 2 chunks puede que estos se unan formando un chunk compuesto por ambas ideas y las relaciones obtenidas de los hechos definidos en PyKE.
Luego de iterar por cada Chunk y obtener en cada uno de estos la información que nos interesa buscada en los hechos guiándonos por las reglas, procedemos a devolver los hechos que fueron válidos al usuario de forma más estructurada, para que sea legible, esta será la respuesta que daremos a su interrogante. En caso de no encontrar cierta cantidad de términos en nuestra base de conocimiento, buscaremos la consulta hecha por el usuario en Wikipedia.

```
def ask(query:str, rules:dict=rules) -> str:
    """Dada una pregunta se devuelve la respuesta a la misma luego de ser procesada por el sistema experto"""
    try:
        sentences = PreprocessText.sentence_segmentation(query)

        chunks_list = []
        for sentence in sentences:
            chunks_list.append(PreprocessText.filter_by_rules(sentence, rules=rules))

        # for chunks in chunks_list:
        #     words = [l for (w,l,p) in chunks]
        output = ""

        print(chunks_list)
        for index, chunks in enumerate(chunks_list):
            output += driver.get_answer(sentences[index], PreprocessText.normalize(chunks))

        return output#driver.get_answer(query, chunks)
    except Exception as e:
        print(e)
        return "Error! :("
```

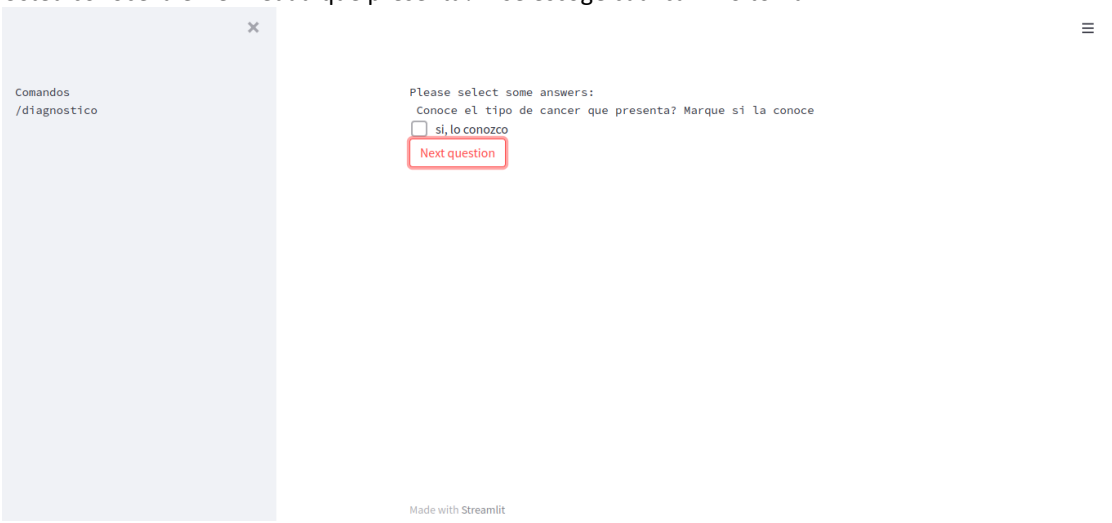
Método ask(), query es la consulta q nos introduce el usuario, rules se define antes y se le asigna una regla para escoger los sustantivos y verbos y tener en cuenta cierta cantidad de palabras consecutivas para buscar como términos en el sistema experto.

Flujo del proceso para cuestionario de asignación de horario a los pacientes
Se supone que se hable sobre las clases Paciente, Doctor y demás creadas, así como del CSP.

La biblioteca de streamlit nos brinda gran facilidad para obtener y guardar en cache cada una de las respuestas seleccionadas por el usuario a través del cuestionario. Haremos uso de estas para ir confeccionando una instancia de la clase Pacient(que representa a los pacientes) y al final se guardan los mismos en la lista st.session_state.patients_list. La cual será utilizada por el CSP.

Se debe de introducir el comando /diagnostico en la entrada de texto de la aplicación visual.

Luego se procede a completar el cuestionario siguiendo los pasos que nos brinda el menú interactivo. A lo largo de este cuestionario, se pueden tomar 2 caminos, uno para los usuarios que ya conocen la enfermedad que presentan y otro para los que no la conocen. ¿Al estar situados en la pregunta de Usted conoce la enfermedad que presenta? se escoge cual camino tomar:



CSP

El csp se utiliza en nuestro Proyecto con el objetivo de generar un horario para un hospital que recibe personas con diagnóstico de algún posible cáncer o a realizar una terapia teniendo en cuenta una cantidad de doctores y aparatos o salas fijas.

Los pacientes tienen varias propiedades, en el caso de los pacientes con tumor diagnosticado, tienen una propiedad que es el tipo de tumor que presentan y el tipo de terapia que reciben, además del nombre y en el caso de los pacientes con posibles tumores que pudieran tener, que necesitan el diagnóstico de un doctor, presentan una lista de los tumores teniendo en cuenta los síntomas que seleccionaron en el cuestionario.

En el primer caso conociendo las terapias: radioterapia, quimioterapia, cirugía, etc. el paciente se la aplica con un aparato que puede ser específicamente para el tipo de tumor que presenta o puede ser un aparato general como es los Rayos – X a través de la radioterapia.

Existen 3 tipos de aparatos, los aparatos normales que en algunos casos también se consideran aparatos para terapias, los aparatos de terapia y el aparato cirugía, que este último aparato cuenta con la cantidad de horas que se demora una cirugía según el tipo de tumor que presenta el paciente que va a ser operado.

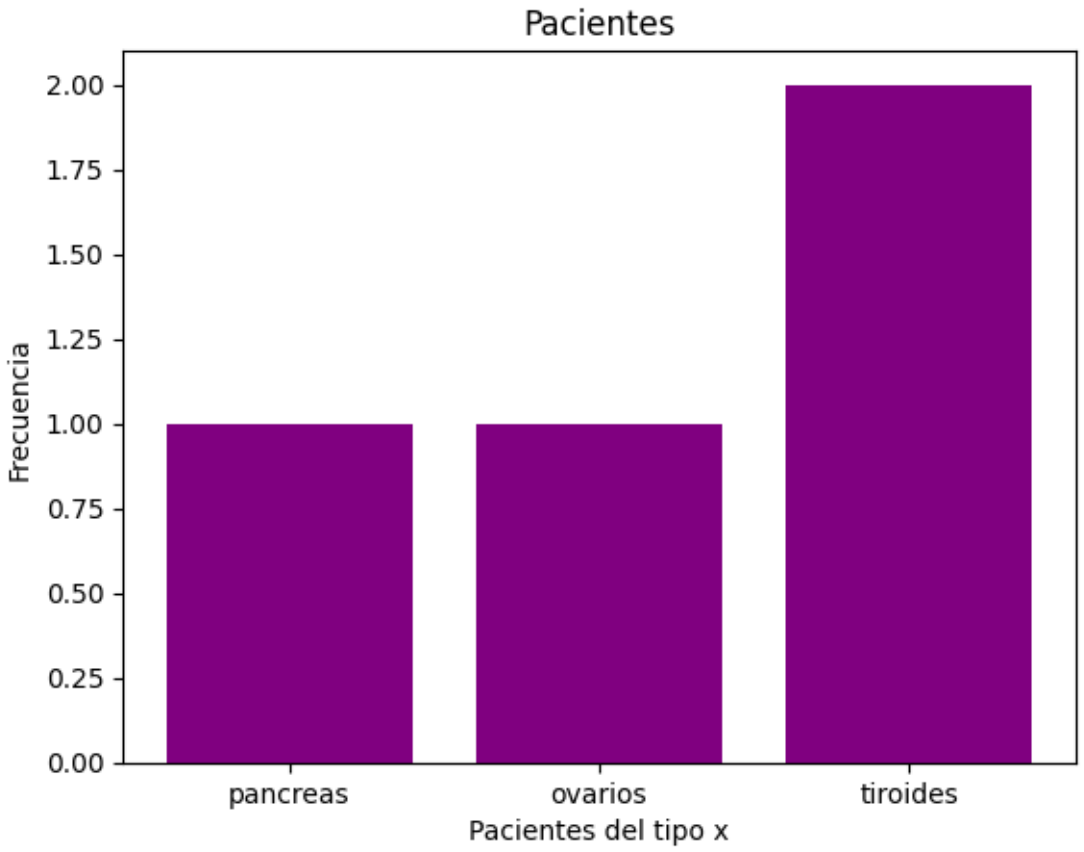
Los doctores presentan diversas características una de ellas es la especialidad que tiene, es decir los doctores pueden ser especialista en una región o órgano del cuerpo o pueden ser de especialidad general, los cuales puede atender a los pacientes que reciben terapias y que no tienen disponible un doctor especialista en el tipo de cáncer que presenta. Los doctores poseen un límite de horas de trabajo, al llegar a ese límite, ese doctor no puede atender a más ningún paciente en ese día. Presenta otra propiedad que es si se encuentra disponible en cierto horario, por ejemplo, si un doctor se encuentra en una cirugía o simplemente no vino al hospital, ese doctor tiene esa propiedad en falso.

En el método hospital_simulation es donde se simulan los turnos del día. Tiene como parámetros de entrada la lista de los pacientes, la lista de los doctores del hospital y la de los aparatos. Cada consulta se demora 2 horas, donde la cantidad de pacientes que se atienden es igual a la cantidad de doctores que se encuentran disponibles en el hospital en ese momento.

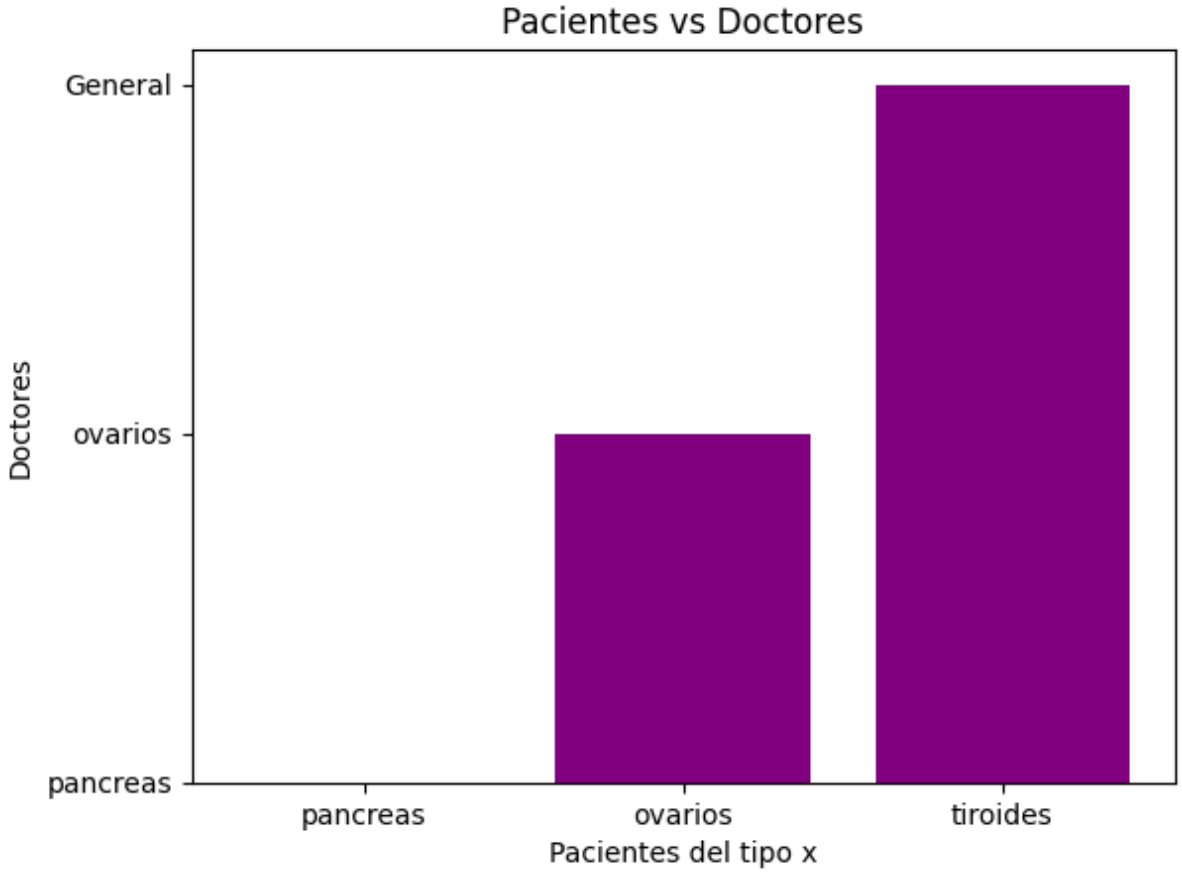
Primero se selecciona a los pacientes que les toca atenderse en la consulta actual, se crea un diccionario en donde se le asigna a esos pacientes los doctores y aparatos que son compatibles con el tipo de cáncer y terapia que presentan o de los posibles cánceres a diagnosticar. Luego se crean los grafos, es decir el grafo que muestra las relaciones entre los pacientes con respecto a los doctores que tienen en común y el grafo que muestra las relaciones entre los pacientes con respecto a los aparatos en común. Teniendo toda esa información se procede a realizar el CSP que es el llamado al método backtraking_search donde en los parámetros de entradas en vez de pasarle los grafos

creados anteriormente, se pasa los grafos ordenados guiándose por las aristas que tienen nodos con mayor dominio que las demás. Una vez se termina el CSP, se guarda esa solución en la lista de csp del día y se actualiza la cantidad de horas que ha trabajado cada doctor y en los casos que exista una cirugía se actualiza la cantidad de horas que pasaron, una vez terminada la cirugía, ese doctor vuelve a estar disponible.
El método ac_3 en donde se realiza el algoritmo del csp, recibe como parámetros de entrada el diccionario descrito anteriormente, las 2 colas a utilizar, la cola de los doctores y la cola de los aparatos que no es más que los grafos ordenados, una lista donde se encuentran las aristas ya visitadas de la cola, las listas con los doctores y los aparatos.
El algoritmo es el siguiente, a la hora de seleccionar los doctores o los aparatos para un par de nodos de una arista se toma del diccionario de pacientes los doctores o aparatos que son compatible con cada nodo(paciente) y se llama al método búsqueda que lo que hace es seleccionar un doctor(aparato) a cada nodo y verifica si existe inconsistencia con el resto de los nodos, si no ocurrió ninguna inconsistencia por el momento ese sería el valor que tendría los nodos en cuanto al doctor y al aparato. Seguido de la verificación de consistencia se agregan las nuevas aristas a los respectivos grafos teniendo en cuenta los vecinos de los nodos actuales. Al terminar ese proceso se comprueba si ya se tiene una solución válida, una solución válida es cuando cada paciente tiene asignado como máximo 1 doctor y un aparato y que no ocurra que un doctor o aparato estén repetidos en más de 1 paciente.
En el método de búsqueda se utiliza un método que se llama actualiza_dominio, este método recibe un paciente, el valor del doctor(aparato) que se le asignó y el diccionario de pacientes y lo que hace es que en el paciente que se le quiere asignar el valor se elimina todos los demás doctores(aparatos) que tiene en el diccionario dejando solamente el del valor que se le quiere asignar y en el resto de los pacientes se le quita ese valor de su dominio porque por el momento ya está tomado.

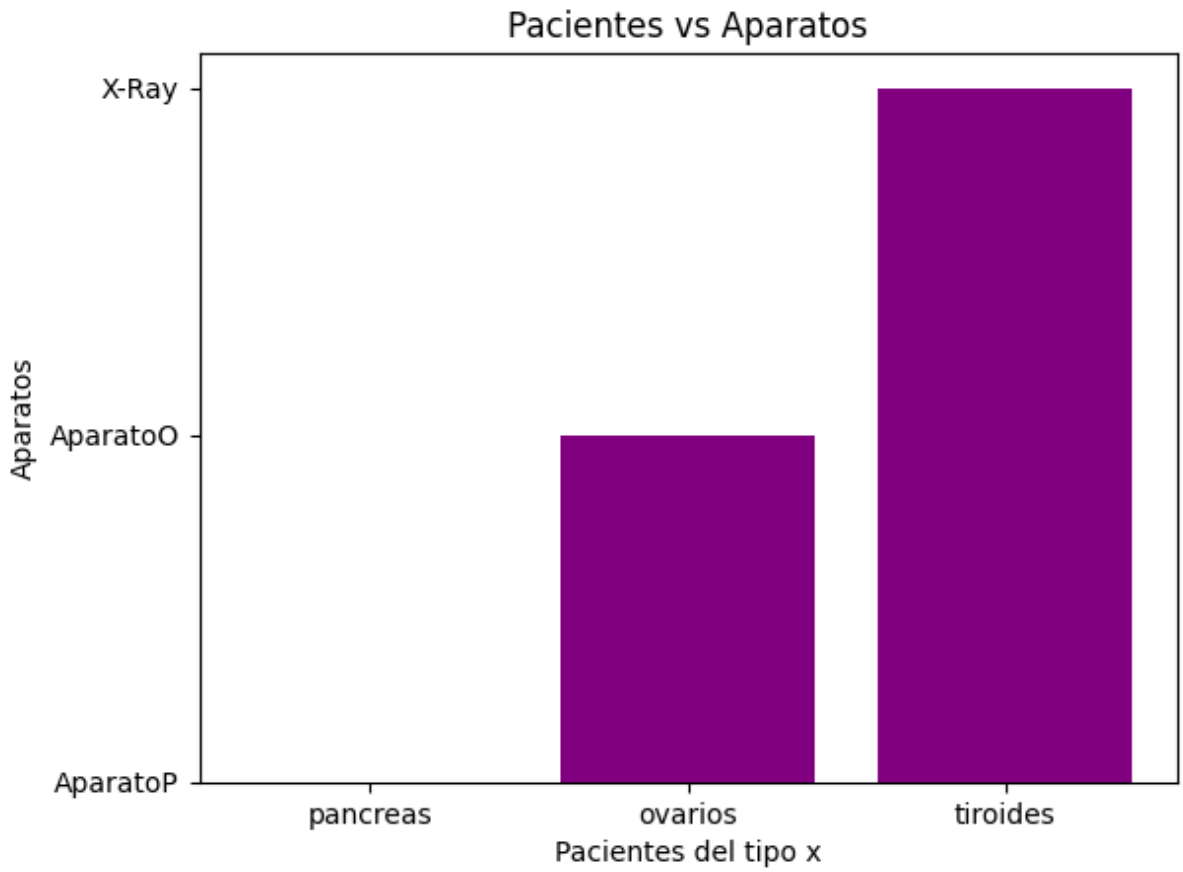
Para mostrar los resultados se simula el algoritmo varias veces a través del método descrito anteriormente hospital_simulation donde en cada llamado se entran pacientes distintos y los resultados que se obtuvieron fueron:



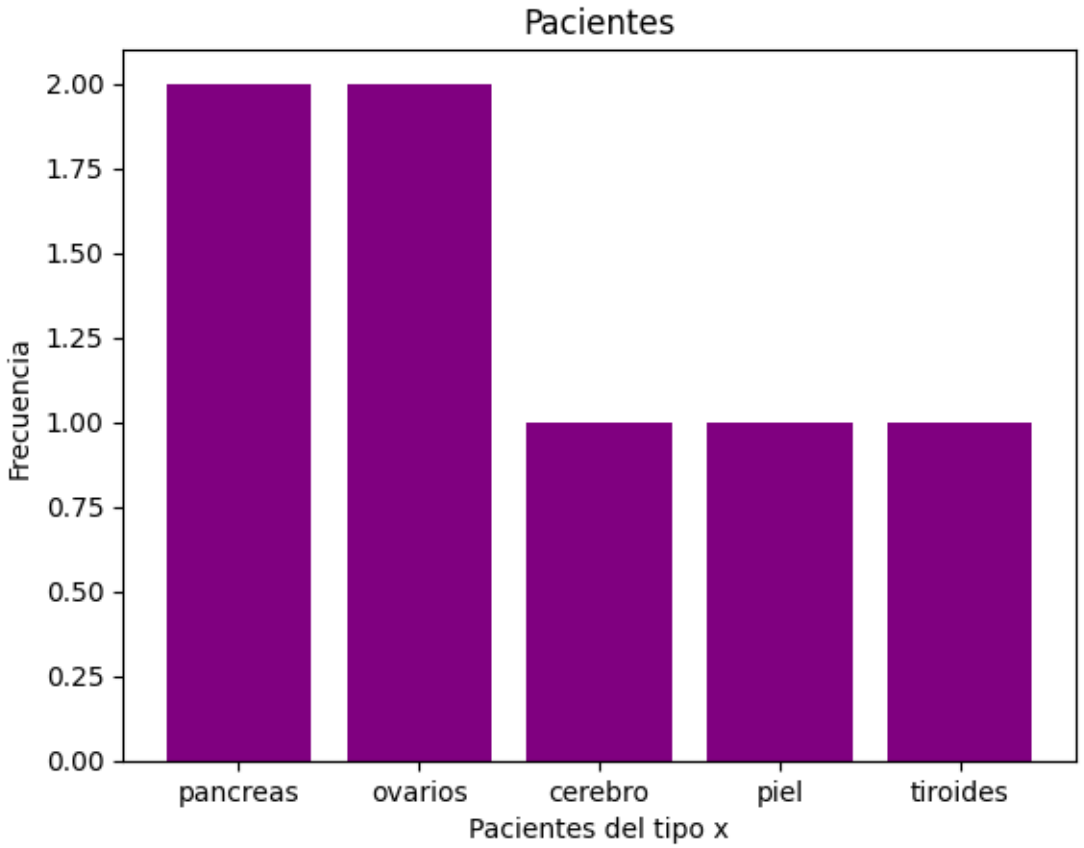
En esta gráfica se muestra la frecuencia de las enfermedades de los pacientes que visitaron el hospital el día de la simulación, como se puede apreciar había más pacientes con cáncer de tiroides o posible diagnóstico de tiroides con respecto a el cáncer de páncreas y ovarios.



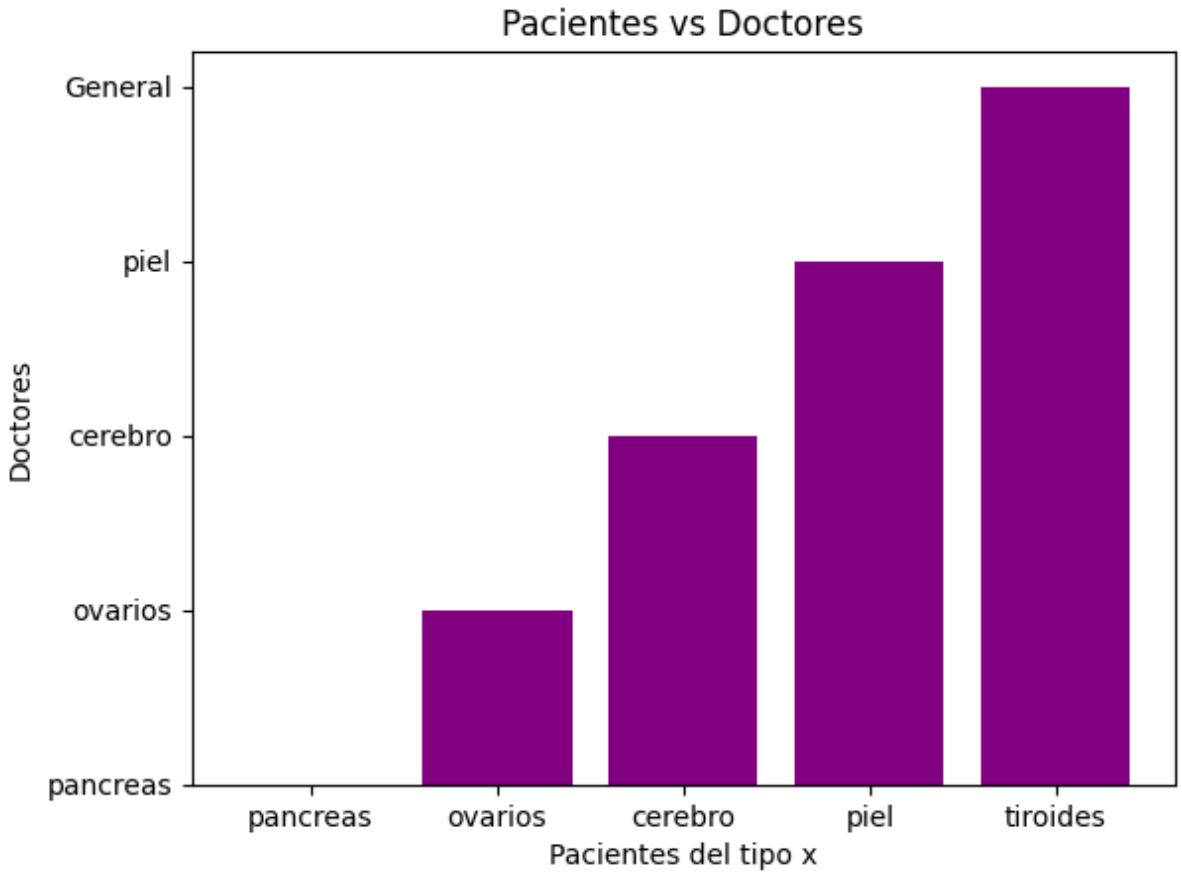
Aquí se muestra la comparación con respecto a los pacientes y los doctores que los atendieron, en el caso del cáncer de tiroides la mayoría de los casos fueron pacientes con terapias, los cuales fueron atendidos por doctores de especialidad General, por otra parte, se encuentran los pacientes con cáncer de ovarios que la mayoría fueron atendidos por especialistas en ese órgano.



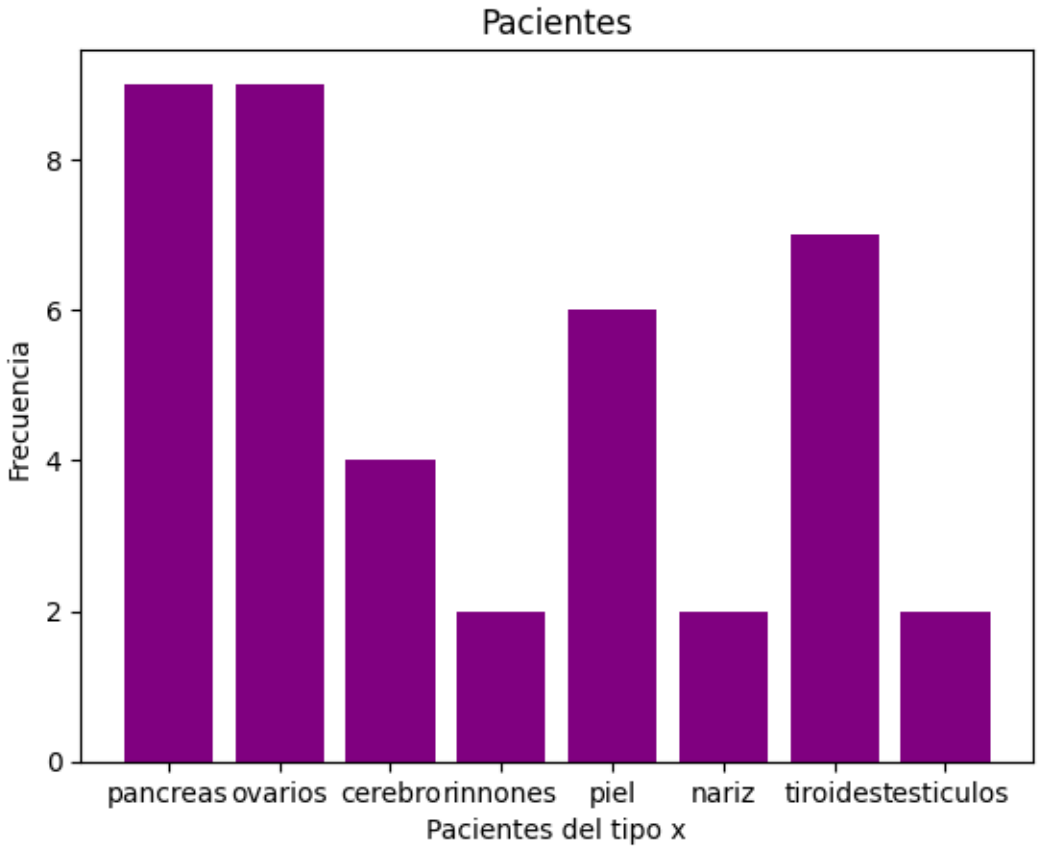
En la gráfica se muestra la relación de los tumores de los pacientes con respecto a los aparatos utilizados, los tumores en los ovarios tienen un aparato que es AparatoO aunque igual se puede usar los rayos X como se describió al inicio, pero en este caso la mayoría de los pacientes que usaron terapias o diagnóstico con rayos X fueron los que presentaban cáncer de tiroides.



Sin embargo, el segundo día hubo más variedad con respecto al tipo de cáncer o posible pronóstico, a diferencia del día 1 existieron más personas con cánceres en el páncreas y los ovarios en comparación con los demás tumores.



Gráfica con respecto a la relación pacientes y doctores en el día 2.



s.

En esta gráfica se muestra el promedio de todos los días como se puede ver la mayor frecuencia lo tuvieron los cánceres de páncreas y ovarios seguido se tiene el de tiroides