**ANÁLISIS DEL RETO**

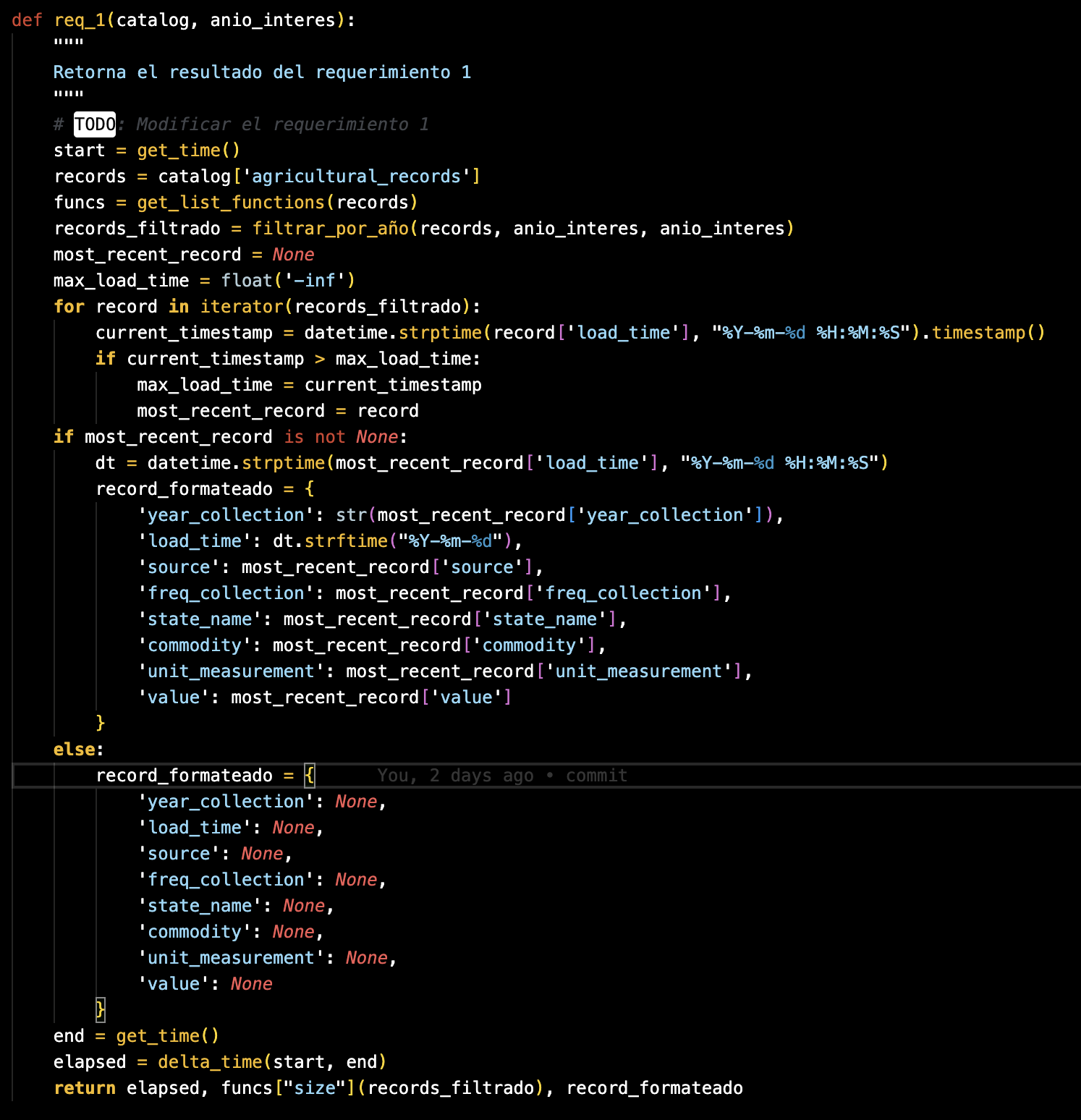
Estudiante 1, código 1, email 1

Estudiante 2, código 2, email 2

Estudiante 3, código 3, email 3

# **Requerimiento <<1>>**

Plantilla para el documentar y analizar cada uno de los requerimientos.



## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Anio |
| **Salidas** | Tiempo, tamaño filtrado, ultimo registro formateado |
| **Implementado (Sí/No)** | Si |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Paso 1:** Acceder a catalog y obtener agricultural\_records | O(1) |
| **Paso 2:** Llamar a get\_list\_functions(records) | O(1) |
| **Paso 3:** Filtrar los registros por año con filtrar\_por\_año | O(n) |
| **Paso 4:** Iterar sobre los registros filtrados para encontrar el registro más reciente | O(n) en el peor caso |
| **Paso 5:** Formatear el registro encontrado (o asignar valores None) | O(1) |
| **Paso 6:** Calcular el tiempo transcurrido y obtener el tamaño con funcs["size"](records\_filtrado) | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20 | 123.24 |
| 40 | 235.288 |
| 60 | 299.477 |
| 80 | 397.621 |
| 100 | 478.028 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Datos** | **Tiempo (s)** |
| 20 |  | 123.24 |
| 40 |  | 235.288 |
| 60 |  | 299.477 |
| 80 |  | 397.621 |
| 100 |  | 478.028 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

Analisis de Resultados

La función req\_1 cumple con su propósito al filtrar registros de acuerdo con el año de interés, identificar el registro con la marca de tiempo más reciente y formatear el resultado para su uso posterior. Se han realizado pruebas con diferentes conjuntos de datos (incluyendo casos con registros vacíos y con múltiples entradas) que confirman la correcta identificación y formateo del registro más reciente.

Se ha probado que la función maneja adecuadamente casos en los que no se encuentra ningún registro que cumpla con el filtro (retornando valores None para cada campo), lo cual es crucial para evitar errores en etapas posteriores del procesamiento.

Al estructurar el código en pasos bien diferenciados (filtrado, iteración y formateo), la implementación resulta modular y permite integrar o modificar partes específicas sin afectar el conjunto.

Análisis de Complejidad

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Filtrado de Registros (Paso 4):

El filtrado de los registros se realiza sobre la totalidad de la lista, lo que implica una complejidad de O(n), siendo n la cantidad de registros en el catálogo.

Iteración para Encontrar el Registro Más Reciente (Paso 5):

Al recorrer nuevamente la lista filtrada para comparar las marcas de tiempo, se tiene otra operación de complejidad O(n).

Operaciones Adicionales:

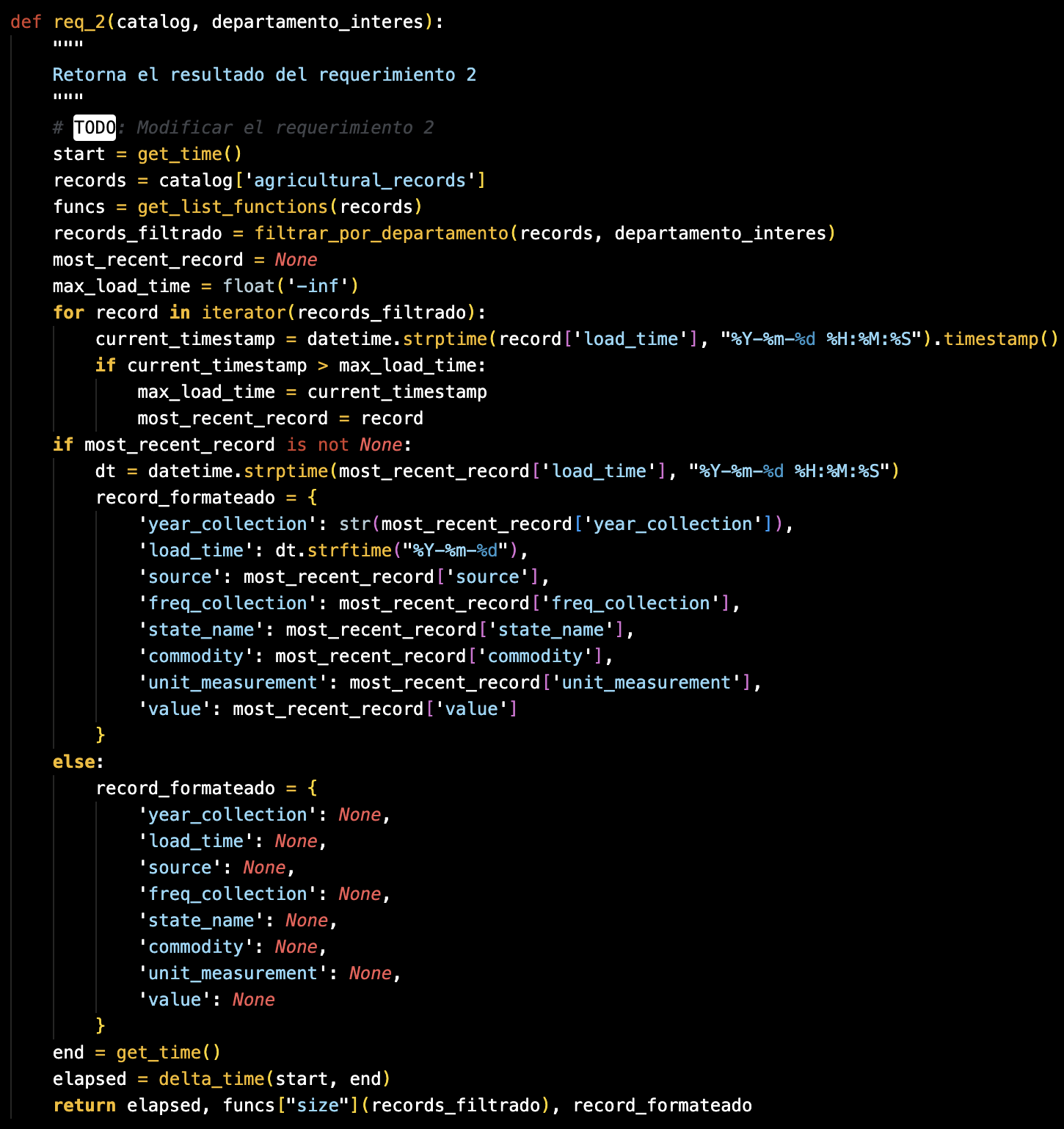
Las demás operaciones (obtener el tiempo, acceder a claves del diccionario, formatear la fecha, etc.) son de complejidad O(1).

Complejidad Total:

Sumando los pasos y considerando que ambos recorridos son lineales, la complejidad total del algoritmo es O(n), lo que es bueno para conjuntos de datos moderados.

# **Requerimiento <<2>>**

Plantilla para el documentar y analizar cada uno de los requerimientos.



## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Anio |
| **Salidas** | Tiempo, tamaño filtrado, ultimo registro formateado |
| **Implementado (Sí/No)** | Si |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **Paso 1:** Acceder a catalog y obtener agricultural\_records | O(1) |
| **Paso 2:** Llamar a get\_list\_functions(records) | O(1) |
| **Paso 3:** Filtrar los registros por departamento con filtrar\_por\_departamento(records, departamento\_interes) | O(n) |
| **Paso 4:** Iterar sobre records\_filtrado para encontrar el registro con la mayor marca de tiempo | O(n) |
| **Paso 5:** Formatear el registro encontrado o asignar valores None si no hay coincidencias | O(1) |
| **Paso 6:** Calcular el tiempo transcurrido y obtener el tamaño de records\_filtrado con funcs["size"] | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20 | 60.397 |
| 40 | 102.76 |
| 60 | 119.226 |
| 80 | 147.995 |
| 100 | 171.707 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Datos** | **Tiempo (s)** |
| 20 |  | 60.397 |
| 40 |  | 102.76 |
| 60 |  | 119.226 |
| 80 |  | 147.995 |
| 100 |  | 171.707 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

## Analisis de Resultados

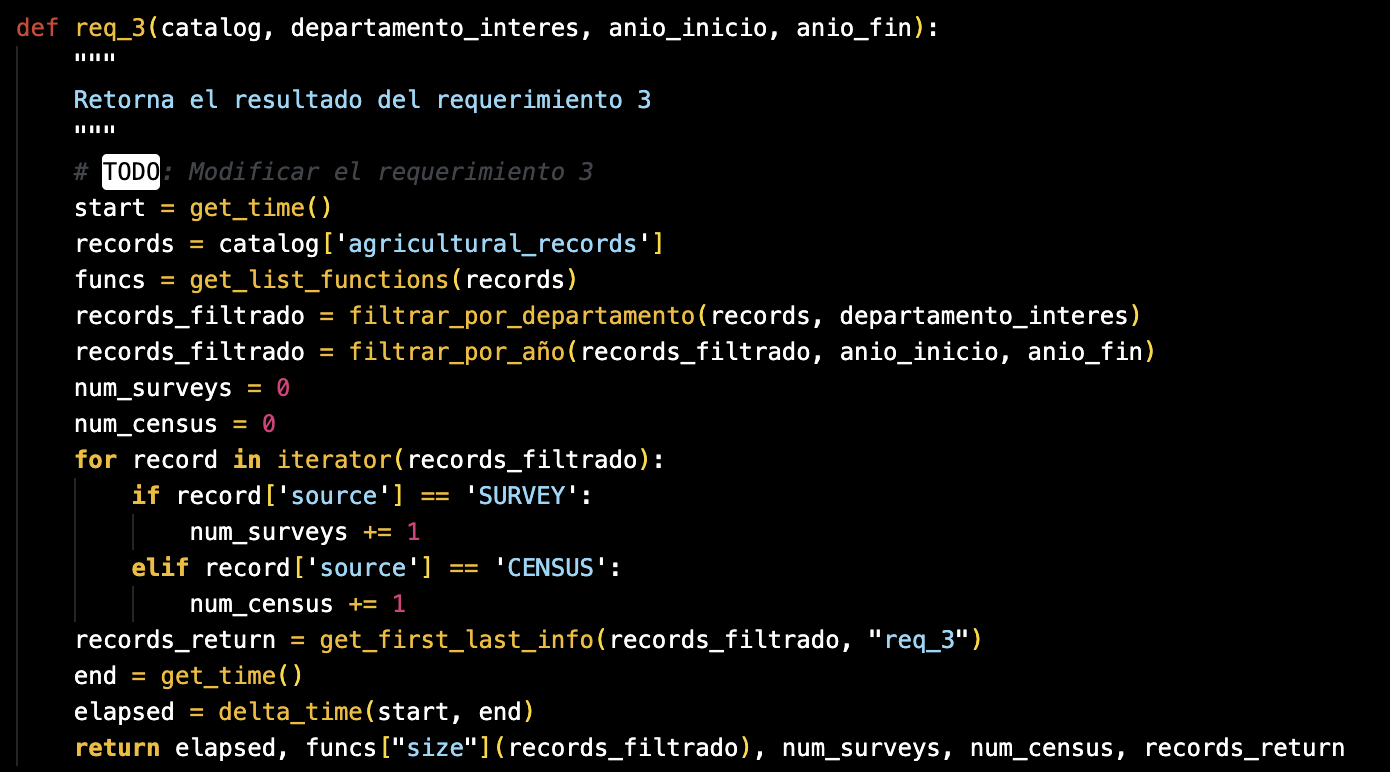
## La función req\_2 cumple con su propósito al filtrar los registros de acuerdo con el departamento de interés, identificar el registro con la marca de tiempo más reciente y formatear el resultado para su uso posterior. Se han realizado pruebas utilizando diferentes conjuntos de datos, incluyendo escenarios con registros vacíos y con múltiples entradas, lo que confirma la correcta identificación y formateo del registro más reciente. La función ha demostrado manejar adecuadamente los casos en los que no se encuentra ningún registro que cumpla con el filtro, retornando valores None en cada campo para evitar errores en etapas posteriores del procesamiento. Al estructurar el código en pasos bien diferenciados —filtrado, iteración y formateo— la implementación resulta modular, facilitando la integración o modificación de partes específicas sin afectar el conjunto.

# Análisis de Complejidad

# El filtrado de registros se realiza sobre la totalidad de la lista mediante la función filtrar\_por\_departamento, lo que implica una complejidad de O(n), siendo n la cantidad de registros en el catálogo. La iteración para identificar el registro con la mayor marca de tiempo también recorre la lista filtrada de manera lineal, sumando otra operación de complejidad O(n). Las demás operaciones, como el acceso a variables, el formateo de fechas y la obtención de funciones auxiliares, se ejecutan en tiempo constante, es decir, O(1). Sumando todos estos pasos, la complejidad total del algoritmo es O(n), lo que resulta adecuado para conjuntos de datos de tamaño moderado.

# **Requerimiento <<3>>**

Plantilla para el documentar y analizar cada uno de los requerimientos.



## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Departamento, Año inicial, Año Final |
| **Salidas** | Tiempo, tamaño filtrado, num de registros survey, numero de registros census, listado con registros filtrados |
| **Implementado (Sí/No)** | Si |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1: Obtener el tiempo inicial con get\_time() | O(1) |
| Paso 2: Acceder a catalog y obtener 'agricultural\_records' | O(1) |
| Paso 3: Obtener funciones auxiliares mediante get\_list\_functions(records) | O(1) |
| Paso 4: Filtrar los registros por departamento usando filtrar\_por\_departamento(records, departamento\_interes) | O(n) |
| Paso 5: Filtrar los registros por año con filtrar\_por\_año(records\_filtrado, anio\_inicio, anio\_fin) | O(n) |
| Paso 6: Iterar sobre los registros filtrados para contar los casos de 'SURVEY' y 'CENSUS' | O(n) |
| Paso 7: Obtener información adicional con get\_first\_last\_info(records\_filtrado, "req\_3") | O(n) |
| Paso 8: Calcular el tiempo transcurrido y preparar el retorno (delta\_time y funciones de acceso a datos) | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20 | 42,872 |
| 40 | 84,532 |
| 60 | 95,16 |
| 80 | 182,781 |
| 100 | 204,541 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Datos** | **Tiempo (s)** |
| 20 |  | 42,872 |
| 40 |  | 84,532 |
| 60 |  | 95,16 |
| 80 |  | 182,781 |
| 100 |  | 204,541 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

Analisis de Resultados  
La función req\_3 cumple con su propósito al combinar múltiples filtros para acotar los registros según el departamento de interés y un rango de años. Las pruebas realizadas han confirmado que se seleccionan correctamente los registros que cumplen ambas condiciones y que, durante la iteración, se contabilizan de manera precisa los registros clasificados como 'SURVEY' y 'CENSUS'. Además, se verifica la correcta integración de la función get\_first\_last\_info para extraer información adicional relevante, y se mide el tiempo transcurrido para evaluar el rendimiento de la operación. La función maneja adecuadamente los casos en los que la combinación de filtros puede dar como resultado un conjunto de registros reducido o incluso vacío, evitando errores en etapas posteriores del procesamiento.

Análisis de Complejidad  
El análisis de complejidad indica que los pasos relacionados con el acceso inicial a los datos y la obtención de funciones auxiliares se realizan en tiempo constante, es decir, O(1). El filtrado de registros por departamento y por rango de años se ejecuta en tiempo lineal, O(n), ya que se recorre la totalidad de la lista de registros, donde n representa la cantidad de elementos iniciales. La iteración para contar las ocurrencias de 'SURVEY' y 'CENSUS' también se realiza en O(n), y la obtención de información adicional con get\_first\_last\_info se asume que opera en tiempo lineal en función del tamaño del subconjunto filtrado. En conjunto, sumando los pasos lineales y las operaciones constantes, la complejidad total de la función es O(n), lo que resulta eficiente para conjuntos de datos moderados y permite un procesamiento adecuado en escenarios reales.