# Posibles Flujos en el sistema: [ACOMODAR COMO ÍNDICE DE CONTENIDO ]

1. Flujo de recolección, filtrado, procesamiento y persistencia de mensajes intercambiados en el canal de consultas de Discord de la asignatura Diseño de Sistemas (Formación de la base de conocimiento).
2. Flujo de captura de preguntas y respuestas en tiempo real en el canal de consultas de Discord de la asignatura Diseño de Sistemas (con el chatbot en producción).
3. Flujo de una consulta en el chatbot.

# Flujo de recolección, filtrado, procesamiento y persistencia de mensajes intercambiados en el canal de consultas de Discord de la asignatura Diseño de Sistemas (Formación de la base de conocimiento).

Se extraen los mensajes intercambiados desde el 2024 hasta marzo del 2025 (incluido) del canal de consultas de Discord de la asignatura Diseño de Sistemas. Los mismos van a estar en archivos con formato JSON, para poder filtrarlos (eliminar contenido irrelevante: mensajes que son vacíos, que son solo stickers, solo emoticones, solo gifs, solo símbolos), procesarlos (clasificar cada mensaje como una pregunta o una respuesta y poder asociarlos cuando corresponda) y persistirlos (una vez clasificados, se almacenan en una estructura intermedia para persistirlos en una base de datos relacional, en este caso, en postgreSQL). De esta manera queda construida la base de conocimiento.

Esta base de conocimiento será una base de datos relacionas constituida por las tablas con sus respectivos atributos:

* Mensajes
  + id\_mensaje: clave primaria (PK).
  + id\_mensaje\_discord: identificador del mensaje de Discord que está en el archivo JSON.
  + autor\_id : clave foránea (FK).
  + fecha\_mensaje: fecha que está en el archivo JSON.
  + Contenido: mensaje que está en el archivo JSON.
  + es\_pregunta: booleano, que tendrá valor FALSE por defecto. Pasará a TRUE si el mensaje es considerado pregunta.
  + origen: indica la ruta (completa) de la que vino el mensaje
* Preguntas
  + id\_pregunta: clave primaria (PK).
  + mensaje\_id : clave foránea (FK).
  + texto: mensaje que está en el archivo JSON.
  + esta\_cerrada: boolean, que tendrá FALSE por defecto. Se considera cerrada cuando un alumno o docente envían mensaje de cierre, o por pasar cierto periodo de tiempo o por cierta cantidad de respuestas. Siempre debe tener al menos una respuesta de un docente (que es una respuesta validada). Los alumnos son los únicos que hacen preguntas. Todo mensaje de docente es considerado una respuesta siempre.
  + sin\_contexto: boolean, que tendrá FALSE por defecto. Se refiere a preguntas cortas con 5 o menos palabras que no pueden identificarse claramente un tema concreto.
  + es\_administrativa: boolean, que tendrá FALSE por defecto. Se refiere a las preguntas vinculadas a temas administrativos como fecha de parciales, armado de grupo de trabajo prácticos, por ejemplo.
* Respuestas
  + id\_respuesta: clave primaria (PK).
  + mensaje\_id: clave foránea (FK).
  + pregunta\_id: clave foránea (FK).
  + Texto: mensaje que está en el archivo JSON.
  + Orden: necesario para recuperar cada una de las respuestas que tuvo el intercambio para cierta pregunta y poder pasarla a Mistral (modelo de lenguaje largo) para que pueda reconstruir una repuesta a partir de ellas.
  + es\_validada: boolean, por defecto es false. Para que pase a true debe ocurrir que sea docente el autor del mensaje.
  + es\_corta: boolean, se refiere a las respuestas cuya cantidad de caracteres es menor o igual a 14.
* Autores
  + id\_autor: clave primaria (PK).
  + nombre\_autor: usuario de discord que aparece en el archivo JSON.
  + es\_docente: boolean, es false por defecto. Esto se marca manualmente a partir de la observación de mensajes en la que se detecto quién era docente.
* Adjuntos
  + id\_adjunto: clave primaria (PK).
  + mensaje\_id: clave foránea (FK).
  + url: es nombre.tipo
  + tipo
* fragmentos\_preguntas
  + id\_fragmento: clave primaria (PK).
  + pregunta\_id: clave foránea (FK).
  + texto\_fragmento: parte del mensaje del JSON identificado como pregunta que entra en un único embedding
  + orden para poder reconstruir la pregunta si es necesario
* Embeddings
  + id\_embedding: clave primaria (PK).
  + fragmento\_id: clave foránea (FK).
  + id\_chroma\_db: identiifcador que se obtiene de la base de datos vectorial.

1. A partir de un conjunto de archivos en formato JSON, donde se encuentran los mensajes intercambiados entre alumnos y docentes, se utiliza una estructura llamada DataFrame que es propia de la librería Pandas de Python. La misma permite guardar los datos en forma de tabla (filas y columnas), lo que facilita su procesamiento.
2. Se van a tener varios filtros para aplicar al conjunto de mensajes, para quitar los irrelevantes. Para lograr trazabilidad, se van a tener archivos logs y csv para poder realizar un control sobre el filtrado y procesamiento de mensajes.
3. Luego de que los filtros se apliquen se procede a clasificar cada mensaje, es decir, determinar cuál es pregunta, cuál es respuesta y cómo se asigna una respuesta a una pregunta. Y asea por cierre de un alumno con un mensaje de agradecimiento, o cierre de un docente afirmando, o por cierta cantidad de mensajes intercambiados o el tiempo transcurrido, las preguntas se van a cerrar: se van a considerar como respondidas.
4. Una vez que se tienen todas las preguntas identificadas con sus respuestas, se van a persisitir en la base de datos relacional.

## Flujo de captura de preguntas y respuestas en tiempo real en Discord (Con el chatbot en producción)

1. Alguien escribe en el canal de “consultas” en Discord
   * Se captura el mensaje usando la API de Discord o discord.py.
2. Se pasa el mensaje a Mistral (o algún modelo de lenguaje largo, similar a chat gpt, mediante una api que sirve para enviar el texto o mediante un modelo corriendo local, lo más común es envió por API) para clasificarlo:
   * Si es una pregunta: Se almacena en la base de datos relacional (se agrega a tabla mensajes y a tabla preguntas, con estado esta\_cerrada=false, también se evalua si el autor es nuevo o no, si existe ese autor, sedebe agregar a la tabla autores, También se hacen los inserts si corresponde en caso de haber archivos adjuntos).
   * Si es una respuesta: Hay que encontrar a qué pregunta pertenece.

* Se toman las preguntas abiertas (si son muchas se toman las últimas 5, no más), se las pasa a Mistral con un prompt adecuado para establecer a cual corresponde dicha respuesta. Si es considerada respuesta para más de una ´prehgunta, se asigna a todas a las que pueda corresponder. Tambien se puede determinar si es un mensaje de cierre o no. Si es de cierre de alumno hay que verificar que la pregunta tenga al menos una respuesta de un docente (una respuesta validada). la pregunta debe cerrarse. Entonces, si el mensaje es de cierre de alumno, no se guarda, pero si es de cierre de docente si se almacena y si no es de cierre, es una repuesta más, se guarda en la base de datos. Cuando se guarda el mensaje, el miso debe persistirse en la tabla mesnajes y en la tabla respuestas y se hace la asociación adecuada entre ´pregunta y respuesta (la calve foránea de la pregunta en la respuesta). También se hacen los inserts si corresponde en caso de haber archivos adjuntos o si el autor es nuevo.
* Si la pegunta se cierra, entonces hay que generar el o los embedding o vectores, de manera de que ya se puedan incluir en la base de datos vectorial y así se pueda realizar la búsqueda semnatica e incorporar esta pregunta y sus respuestas para ser accedidas por el chatbot, por los alumnos al realizar una consulat.

## Flujo de una consulta en el chatbot

1. Se captura la pregunta del usuario en Discord.
2. Se genera un embedding de la pregunta y se busca en ChromaDB (comparando con embeddings de preguntas previas).
3. Se obtiene la preguntas más similar y se recuperan sus respuestas desde la base relacional.

* Si hay muchas respuestas:
* Mistral es quien combina respuestas usando todas las respuestas existentes.
* Si no hay respuestas, se indica que debe realizarse la pregunta en Discord porque nunca antes se hizo.

1. Se muestra en el chatbot la respuesta correspondiente.