## línea horizontal



Memoria Asistente de Orientación a la Universidad

20/06/2025

**─**

Adrián Olmos Amador

# Visión general

El objetivo principal de este proyecto es desarrollar un **asistente virtual de orientación universitaria**, capaz de guiar a estudiantes en el proceso de toma de decisiones académicas hacia su futuro universitario.

Este asistente, llamado **Edu**, está diseñado para:

* 👤 **Brindar una experiencia personalizada**, iniciando con una serie de preguntas para conocer los intereses, preferencias y situación académica del usuario.
* 🎓 **Ayudar a descubrir el grado universitario más adecuado** según los intereses del estudiante.
* 🧭 **Informar sobre las distintas vías de acceso a la universidad**, en función del perfil del usuario.
* ❓ **Resolver dudas frecuentes** sobre requisitos de acceso, notas de corte, procesos de admisión y plazos clave.
* 🏛️ **Ofrecer información detallada** sobre titulaciones, universidades disponibles y trámites administrativos.

El asistente busca ser una herramienta clara, amigable y útil para estudiantes que se enfrentan al complejo proceso de elección académica, proporcionando orientación inmediata y confiable.

# Objetivos Primera Fase

1. Recopilar información sobre el estudiante
2. Guardar datos del estudiante en Base de Datos

# Objetivos Segunda Fase

1. Scrapping de Web para optimización de sugerencias educativas
2. Consulta de memoria en chat

# Tecnologías utilizadas

El proyecto emplea **n8n** como plataforma principal para la **automatización de flujos de trabajo** y la **orquestación lógica del asistente virtual**. Esta herramienta low-code y de código abierto permite conectar múltiples servicios y definir comportamientos dinámicos de forma visual y escalable.

##### **Componentes y nodos destacados utilizados en n8n:**

* **Nodo de Telegram**: Telegram ha sido elegido como **canal de comunicación principal** del asistente virtual. A través del nodo de Telegram en n8n, se gestionan la recepción y el envío de mensajes entre el usuario final y el asistente, ofreciendo una experiencia de uso directa, accesible y sin necesidad de descargar aplicaciones adicionales.
* **Nodo de Agente de IA**: Este nodo permite el **procesamiento inteligente de los mensajes enviados por el usuario**. Funciona como el motor conversacional, interpretando las preguntas, proporcionando respuestas personalizadas y guiando al usuario en función de sus respuestas previas.
* **Nodo de Supabase**: Supabase se utiliza como **sistema de base de datos** para almacenar información estructurada del usuario (respuestas, contexto de conversación, preferencias, etc.). **n8n** interactúa con Supabase mediante sus nodos para realizar operaciones CRUD (crear, leer, actualizar, eliminar), manteniendo la persistencia del estado y la trazabilidad de las interacciones.
* **Nodo de JavaScript**: Se han incorporado nodos de código JavaScript personalizados para tareas como:  
  + **Formateo de entradas** antes de ser procesadas por el asistente.
  + **Validación de campos** introducidos por el usuario (por ejemplo, comprobaciones de fechas, estructura de texto, opciones seleccionadas, etc.).
  + Transformaciones dinámicas necesarias entre nodos, para adaptar las salidas de un servicio como entrada válida para otro.

# Evolución del Proyecto

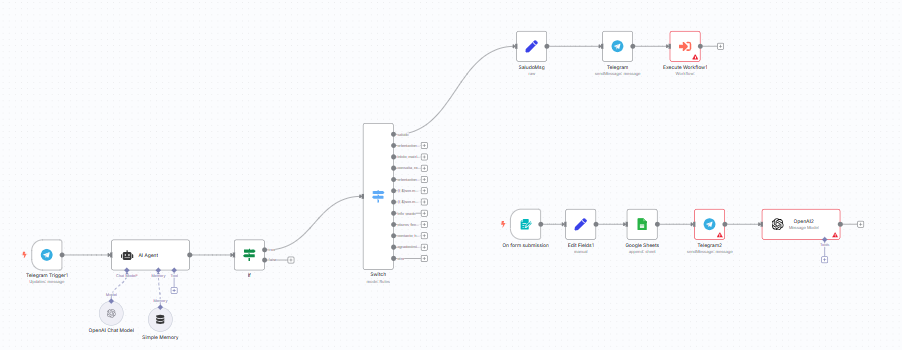
#### **Fases de desarrollo**

* **10 de junio de 2025 – Inicio del proyecto** El proyecto comenzó con una fase intensiva de **investigación y aprendizaje sobre n8n**, ya que el enfoque basado en flujos de trabajo representaba un cambio importante respecto al modelo de programación modular al que el desarrollador estaba habituado. Esta etapa fue clave para adquirir las bases conceptuales necesarias para estructurar flujos funcionales y sostenibles.
* **Exploración del ecosistema y configuración de servicios** En una segunda etapa se abordó la **obtención y configuración de múltiples API Keys** necesarias para los servicios potencialmente integrables en el proyecto.  
   Se registraron y probaron conexiones con los siguientes servicios:  
  + **OpenAI**: para el procesamiento de lenguaje natural mediante un Agente de IA.
  + **Supabase**: como sistema de base de datos en la nube, para persistencia de datos del usuario.
  + **OracleMongoDB** y **Gemini (de Google)**: también fueron considerados para el almacenamiento o procesamiento inteligente, aunque finalmente no se incorporaron al flujo productivo del proyecto.
* **Diseño del asistente virtual y estructura del flujo conversacional** Una vez familiarizado con n8n y con los servicios preparados, se inició la fase de **planteamiento del chatbot**.  
   Se definieron aspectos clave como:  
  + **Qué datos se iban a recopilar del usuario**, como intereses académicos, nivel educativo actual, preferencias de estudio, etc.
  + **Cómo sería el flujo de preguntas y decisiones**, asegurando que cada etapa condujera lógicamente a la siguiente, permitiendo adaptar la conversación en función de las respuestas.
* **Prototipado y pruebas de diseño** Posteriormente, se inició la fase de **primeros diseños a modo de aprendizaje**.  
   Durante esta etapa se:  
  + **Probaron diferentes diseños y herramientas** para la construcción del flujo y la interacción.
  + Se fue **perfilando el diseño final** del asistente, tomando en cuenta la experiencia de usuario y la efectividad de la comunicación.
  + Se realizaron pruebas iterativas para ajustar los flujos y detectar posibles mejoras.
* **Consolidación del diseño final y herramientas** Una vez asentadas las bases y con mayor claridad sobre el funcionamiento de n8n y las necesidades del asistente, se procedió a **diseñar la versión definitiva del flujo conversacional**.  
   En esta fase:  
  + Se **consolidaron las herramientas elegidas**: Telegram como canal de comunicación, Supabase como base de datos, OpenAI como motor de IA y JavaScript para lógica auxiliar.
  + Se estructuró el flujo completo de conversación, teniendo en cuenta ramificaciones, condiciones y puntos de almacenamiento de datos.
  + Se establecieron los bloques finales del asistente, con diseño modular, limpio y enfocado en la orientación universitaria.

# Evolución del Diseño del Flujo Conversacional

Durante el desarrollo del asistente, el diseño del flujo conversacional atravesó varias iteraciones. Cada versión fue refinando tanto la lógica como la experiencia de usuario. A continuación se describen algunos de los principales desafíos enfrentados, junto con las soluciones aplicadas en cada caso.

## Laboratorio 1



En la fase inicial del desarrollo del workflow en n8n, esbocé diversas ideas sobre nodos que podrían resultar útiles en futuras iteraciones. Durante este proceso, redescubrí el potencial del nodo *switch* y reforcé mi manejo de estructuras JSON, elementos clave para dirigir el flujo de datos. Sin embargo, uno de los primeros retos fue adaptar mi mentalidad tradicional, basada en programación modular, al paradigma orientado a flujos que propone n8n. Este cambio conceptual no fue inmediato ni sencillo.

Intenté replicar el comportamiento de funciones mediante sub-workflows, buscando modularizar tareas dentro del flujo principal. Pero pronto me encontré con un problema: los sub-workflows generaban bucles infinitos debido a un planteamiento inicial inadecuado en la gestión del flujo de datos. Esta experiencia me llevó a reflexionar sobre la necesidad de un control más estricto y lineal en la recopilación de información.

Para superar esta dificultad, probé integrar un formulario que facilitara la captura ordenada de datos. Sin embargo, el formulario resultó poco compatible con la dinámica conversacional que buscaba para la interacción vía Telegram al mandar un link que te sacaba del chat.Por este motivo, opté por descartarlo y buscar una solución más orgánica.

Simultáneamente, exploré la incorporación de una inteligencia artificial básica mediante un prompt sencillo. Esta IA tenía el propósito de extraer el contexto del mensaje del usuario y clasificar automáticamente la información en categorías como nombre, edad, ciudad, entre otras. Aunque esta idea aportaba una interacción más flexible e inteligente, pronto me enfrenté a varios problemas para lograr una clasificación precisa y consistente del texto con el agente de IA. Estas dificultades evidenciaron la complejidad de interpretar mensajes abiertos y variados, lo que me impulsó a buscar soluciones más robustas y específicas para asegurar la calidad y fiabilidad en la recopilación de datos.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

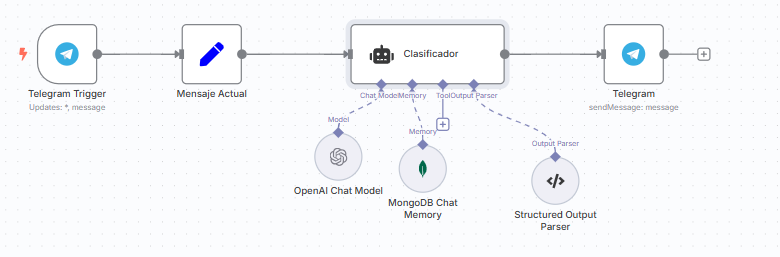
## 

## 

## 

## 

## Laboratorio 2



Ante las dificultades persistentes en la clasificación precisa de la información proporcionada por los usuarios mediante el agente de IA inicial, decidí profundizar en nuevas herramientas que pudieran mejorar el rendimiento del sistema. Fue en este proceso de investigación donde descubrí **MongoDB** y, en particular, su herramienta *ChatMemory*. Esta funcionalidad ofrecía la posibilidad de conservar el contexto conversacional de manera estructurada y persistente, lo cual resultaba especialmente útil para corregir los errores de clasificación previamente detectados.

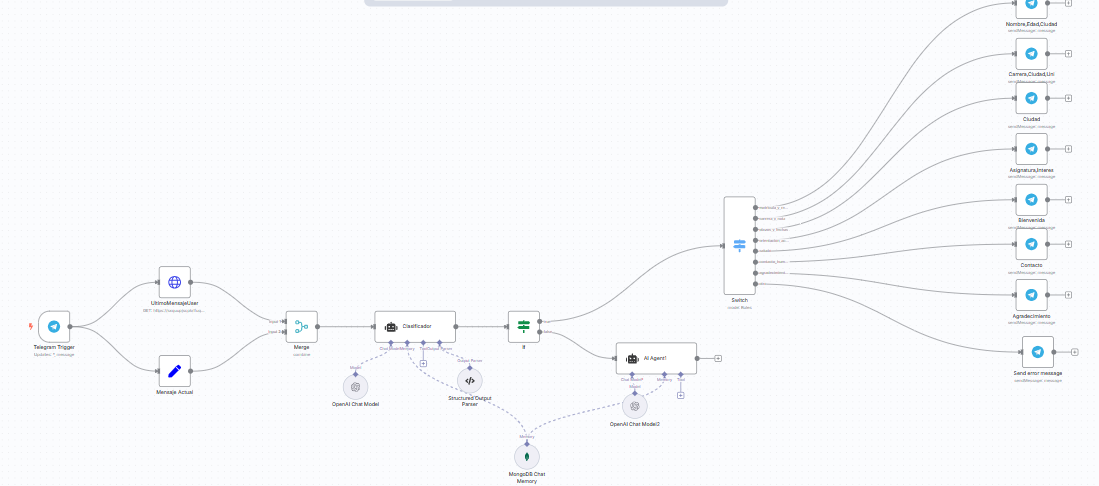
A partir de este hallazgo, diseñé un workflow de prueba en el que desarrollé un nuevo **agente de IA conectado directamente a una base de datos MongoDB**. Este agente recibía los mensajes del estudiante y, en función del contexto acumulado, era capaz de responder con afirmaciones directas cuando se disponía de la información suficiente, o formular preguntas específicas cuando detectaba lagunas en los datos recopilados. En otras palabras, comenzaba a comportarse como un verdadero **orientador educativo**, capaz de conducir de forma inteligente y empática el proceso de entrevista.

Este experimento representó un punto de inflexión en el desarrollo del proyecto. Por primera vez, observé cómo la inteligencia artificial, asistida por un sistema de memoria contextual, era capaz de mantener un hilo conversacional coherente, personalizado y orientado a un objetivo claro: conocer al estudiante y guiarlo de manera adecuada. El resultado fue una interacción significativamente más fluida, natural y funcional, que abría la puerta a implementar asistentes virtuales con un alto nivel de adaptabilidad y sensibilidad frente a distintos perfiles de usuario.

## 

## 

## Laboratorio 3



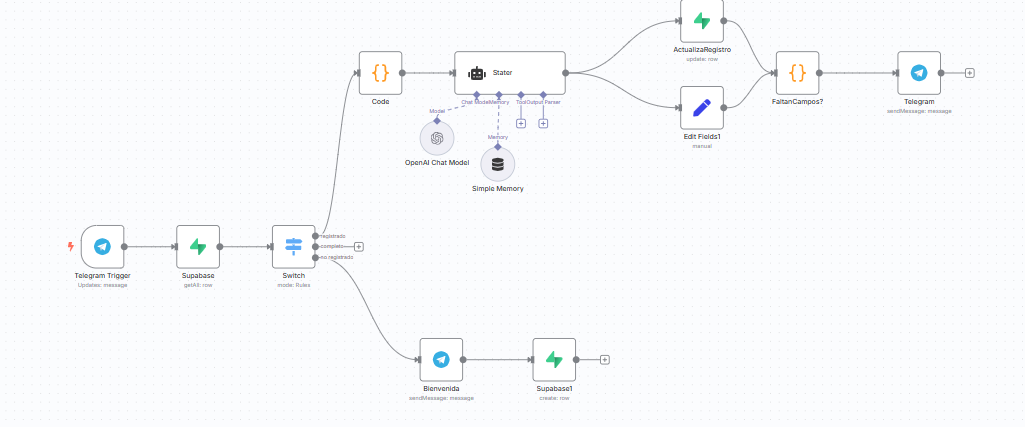
Con el desarrollo del agente de IA respaldado por memoria contextual, el siguiente paso natural fue avanzar hacia una **lógica de estados** que permitiera estructurar y controlar con mayor precisión el flujo conversacional. En este nuevo enfoque, cada mensaje recibido era evaluado y clasificado dentro de uno de dos estados principales: **"Inicial"** o **"Respuesta"**. Esta clasificación permitía establecer una bifurcación clara en el comportamiento del sistema: si el mensaje se encontraba en estado "Inicial", el chatbot respondía con un mensaje por defecto diseñado para iniciar la conversación; en cambio, si se trataba de una "Respuesta", se activaba un segundo agente de IA, más especializado, que aprovechaba el contexto almacenado y el tema identificado para ofrecer una contestación más adecuada y personalizada.

A su vez, cada mensaje era etiquetado según la categoría de información a la que hacía referencia —por ejemplo, “Nombre”, “Ciudad”, “Edad”, etc.—, lo que proporcionaba al agente una guía adicional para generar respuestas más precisas y orientadas a recopilar datos específicos. Esta combinación de **clasificación por estado** y **etiquetado temático** representó un avance sustancial en la calidad de la interacción, permitiendo un mayor grado de adaptabilidad en las respuestas del asistente.

Sin embargo, este enfoque trajo consigo nuevos desafíos. La implementación de múltiples agentes en paralelo, especialmente en procesos con lógica condicional y uso intensivo de contexto, provocó un **consumo elevado de recursos**, tanto en términos de tiempo de ejecución como de **tokens utilizados en la API de la IA**. Esto impactaba directamente en la eficiencia del sistema y su escalabilidad. Además, el nivel de personalización alcanzado en las respuestas generadas, si bien aportaba naturalidad a la conversación, introducía una dificultad inesperada: **la conversación se desviaba con facilidad**, impidiendo que se completara la recopilación ordenada de los campos requeridos.

A pesar de estas complicaciones, la introducción de una lógica de estados se reveló como un acierto clave en la estructuración del flujo conversacional. A partir de esta base, se hizo evidente la necesidad de diseñar una solución más **estable, eficiente y enfocada** a la recolección de datos esenciales, sin sacrificar la calidad de la experiencia conversacional.

## Laboratorio 4



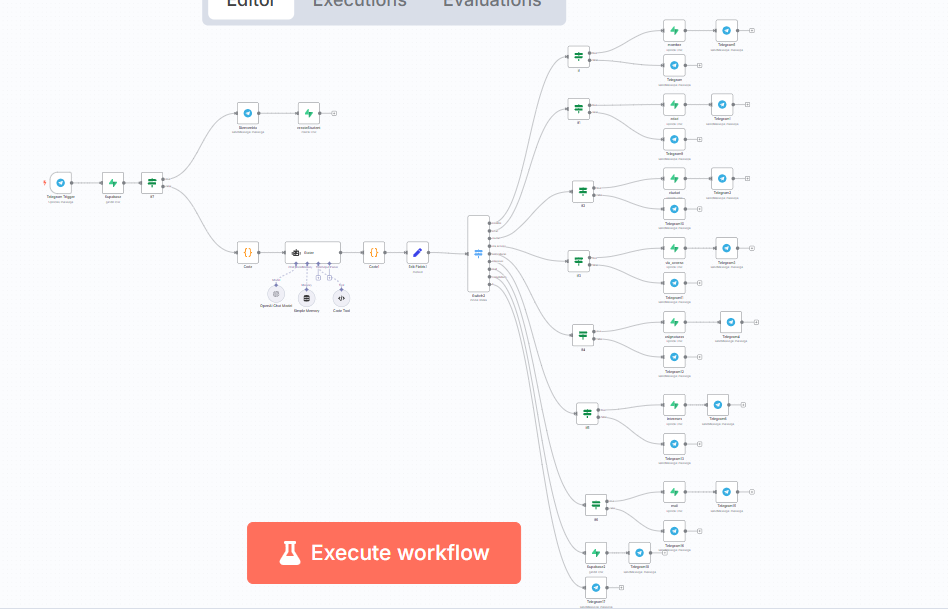
Tras los problemas detectados en el denominado *Laboratorio 5*, donde la conversación tendía a desviarse y no se lograba recopilar toda la información requerida, decidí replantear por completo la estructura del flujo. En lugar de seguir basándome en una conversación abierta y dependiente de memoria contextual, opté por diseñar un enfoque mucho más **lineal y controlado**, orientado específicamente a asegurar la correcta recolección de datos clave.

Este nuevo diseño prescindió de la memoria conversacional y, en su lugar, se implementó una **base de datos Supabase**  como núcleo de persistencia, encargada de almacenar de manera progresiva la información recogida de cada estudiante. Desde el inicio del chat —en el que se daba la bienvenida al usuario— se generaba un nuevo registro vinculado a su ID de Telegram. A partir de allí, cada mensaje recibido era analizado por un agente de IA que clasificaba el contenido según un conjunto de **estados predefinidos**: "nombre", "edad", "ciudad", "via\_acceso", "intereses", "asignaturas" y "mail".

La lógica era sencilla pero efectiva: el agente recibía dos insumos clave —el mensaje del usuario y el estado actual de la conversación— y, en función de ello, evaluaba si el contenido del mensaje contenía información relevante para el campo correspondiente. Si era así, el sistema procedía a **actualizar el registro del estudiante en la base de datos**, consolidando progresivamente su perfil académico. Este cambio de enfoque permitió una recopilación mucho más fiable y estructurada, reduciendo la ambigüedad en la interpretación de los mensajes y eliminando la dependencia de agentes complejos o memoria prolongada.

Este rediseño marcó un avance importante hacia la estabilidad y funcionalidad del asistente, al equilibrar de forma efectiva la intervención de la IA con una lógica conversacional secuencial y orientada a objetivos concretos.

## Diseño Final



El denominado *Laboratorio 4* representó el primer boceto funcional del diseño final del workflow, integrando todos los aprendizajes acumulados hasta ese punto. Se consolidó la **lógica de estados**, se mantuvo la **persistencia de datos mediante una base de datos estable (Supabase)**, y se desarrolló un **agente inteligente** que, guiado por el estado actual, analizaba el contenido del mensaje recibido en busca de la información correspondiente.

En esta arquitectura, los estados avanzaban de manera **lineal y secuencial**, siguiendo el siguiente orden predefinido: "nombre", "edad", "ciudad", "via\_acceso", "intereses", "asignaturas" y finalmente "mail". Esta secuencia estructuraba el flujo conversacional y aseguraba que el chatbot guiara al estudiante de forma ordenada a través de todos los campos necesarios.

Durante la ejecución del flujo —como se puede observar en las últimas ramas del diseño—, cada mensaje recibido era procesado por el agente en función del estado actual. Por ejemplo, si el agente recibía el mensaje "Mi nombre es Juan" y el estado era "nombre", se esperaba que extrajera ese dato específico. Una vez identificado, el valor se actualizaba directamente en la **columna de la base de datos que coincidía con el nombre del estado**, consolidando así el perfil del estudiante.

La lógica de ejecución general del flujo se definió de la siguiente manera:

1. **Recepción del mensaje del usuario.**
2. **¿El usuario ya está registrado?**
   * **No:** Se crea una nueva entrada en la base de datos Supabase, se asigna el estado inicial "nombre", y se envía un mensaje de bienvenida.
   * **Sí:** Se envían el mensaje y el estado actuales al agente de IA. Este analiza el contenido, intenta extraer el dato correspondiente al estado, y devuelve una respuesta.
3. **Evaluación del resultado:**
   * Si la salida del agente incluye información válida, se actualiza el campo correspondiente en el registro del estudiante.
   * Si el agente devuelve una marca de "NoInfo", indicando que no ha podido extraer correctamente el dato, el flujo solicita al estudiante que repita su respuesta, evitando así que se avance hasta que se haya capturado correctamente la información.

Las columnas configuradas en la base de datos de Supabase para este propósito fueron: "nombre", "edad", "ciudad", "via\_acceso", "asignaturas", "intereses" y "estado". Esta última columna (estado) cumplía una función crítica: actuar como un puntero que guiaba al sistema sobre qué dato debía solicitarse o verificarse en cada paso de la conversación.

Este diseño permitió alcanzar un **flujo robusto, controlado y progresivo**, en el que cada etapa estaba bien definida y validada, reduciendo al mínimo los errores de clasificación y permitiendo una recopilación de datos eficiente, incluso ante respuestas ambiguas por parte del usuario. Supuso una evolución clara respecto a versiones anteriores y sentó las bases de un sistema confiable, escalable y alineado con el objetivo final: ofrecer una experiencia orientativa inteligente, pero también estructurada.

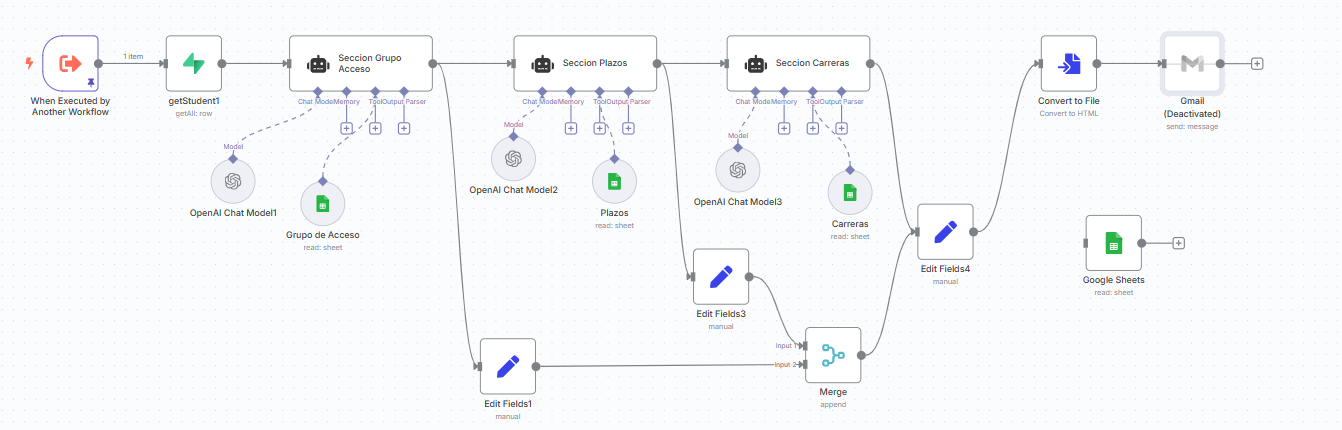
## Evolutivo - Diseño

Como evolución natural del flujo desarrollado, se planteó la incorporación de un **subworkflow** orientado a mejorar la experiencia post-interacción del estudiante. Este subflujo tiene como objetivo **generar un correo electrónico personalizado** que consolide la información recopilada durante la conversación y ofrezca una guía clara y útil al usuario, adaptada a su perfil académico y contexto geográfico.

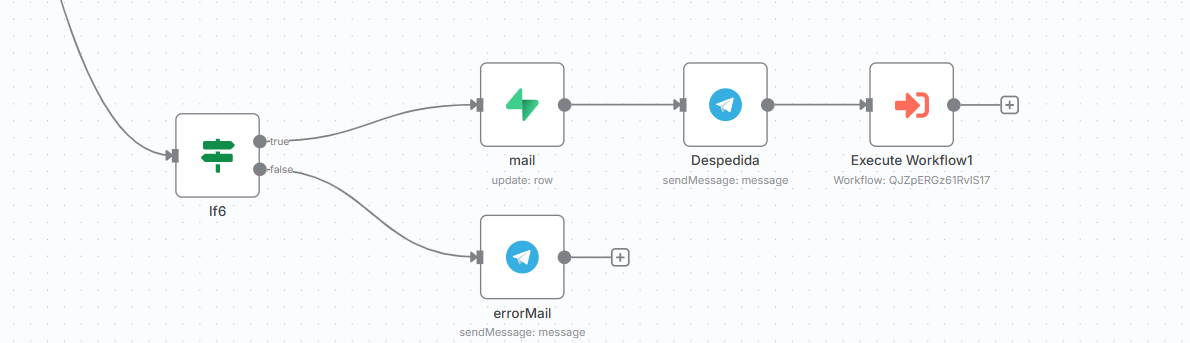
Este correo incluirá tres bloques principales:

1. **Información sobre el grupo de acceso a la universidad** al que pertenece el estudiante, según su edad y vía de acceso.
2. **Plazos clave de matriculación e inscripción** en función de la ciudad de residencia.
3. **Sugerencias de posibles carreras** a partir de sus intereses y asignaturas favoritas.

La ejecución de este subworkflow se activa **automáticamente al finalizar el flujo principal**, una vez recopilados todos los datos requeridos y enviado el mensaje de despedida.



Su diseño modular se apoya en tres agentes de IA especializados, que trabajan de forma secuencial para construir las diferentes secciones del correo:

1. **Primer agente:** Este componente recoge toda la información personal del estudiante directamente desde la base de datos (almacenada en Google Sheets). A continuación, accede a la **tabla de Grupos de Acceso**, que contiene campos como: IdGrupo, Grupo, Definición, Edad, Experiencia, y URL de información. En base a la edad y vía de acceso del estudiante, el agente genera un primer párrafo personalizado que explica a qué grupo pertenece y le ofrece un resumen adaptado de su situación académica, incluyendo enlaces relevantes.
2. **Segundo agente:** A partir de la ciudad de residencia del estudiante, este agente consulta una tabla adicional llamada **Plazos**, cuyas columnas incluyen: Comunidad Autónoma, Publicación de listas, Reclamaciones, y Matrícula / Reserva. Con esta información, se genera el segundo párrafo del correo, enfocado en **los plazos y fechas importantes** que aplican al estudiante en función de su ubicación geográfica.
3. **Tercer agente:** Finalmente, el tercer agente analiza los datos relacionados con los **intereses y asignaturas favoritas** del estudiante. Con ello, consulta una **tabla de Carreras** que contiene los campos Carrera, Intereses (Recomendado para...), y Asignaturas típicas. Este agente genera un párrafo final con **sugerencias personalizadas de carreras universitarias**, justificadas según el perfil del estudiante, en un tono orientador e inspirador.  
   

Una vez que los tres bloques han sido generados, el sistema procede a **unir las secciones en orden**, dar formato al mensaje completo (título, saludo, cuerpo, cierre) y, finalmente, **enviar el correo electrónico** al estudiante utilizando el email previamente recopilado en el flujo principal.

Este subworkflow marca un salto cualitativo en la funcionalidad del sistema, al transformar la simple recolección de datos en una **experiencia de orientación completa**, personalizada y automatizada. Además, refuerza la escalabilidad del proyecto mediante el uso de subprocesos autónomos, manteniendo la limpieza y modularidad del diseño general.

# Conclusión

El desarrollo de este workflow en la plataforma **n8n** ha representado mucho más que una simple automatización de tareas: ha sido un proceso formativo intensivo y acelerado, en el que he tenido la oportunidad de **evolucionar técnica y conceptualmente como desarrollador**. En muy poco tiempo, he pasado de un enfoque tradicional basado en la programación modular, a comprender y dominar la lógica **orientada a flujos** que propone n8n, una transición que no estuvo exenta de retos, pero que ha sido profundamente enriquecedora.

Desde los primeros esquemas —marcados por intentos fallidos de replicar funciones mediante sub-workflows y estructuras que desembocaban en bucles infinitos— hasta la implementación de un sistema estable basado en **lógica de estados**, agentes de IA y una base de datos en **Supabase**, el proyecto ha ido madurando progresivamente. Cada obstáculo se convirtió en una oportunidad para profundizar: las dificultades para clasificar texto mediante IA condujeron a explorar y aplicar la **ChatMemory de MongoDB**, y más adelante, a estructurar el flujo con mayor control para evitar conversaciones erráticas.

He aprendido a **orquestar procesos complejos mediante la integración de múltiples tecnologías**, como:

* **n8n**, como motor principal de automatización y flujo.
* **Supabase**, para la persistencia de datos en tiempo real.
* **Google Sheets**, como almacén consultable para tablas de referencia externas.
* **Modelos LLM**, utilizados como agentes inteligentes con prompts adaptados.
* **Telegram**, como canal de entrada para la interacción con el usuario.

Además, el diseño y posterior evolución hacia un **subworkflow final**, capaz de generar correos electrónicos personalizados según múltiples fuentes de datos y características del estudiante, ha sido una demostración concreta de cómo los agentes de IA pueden operar de forma **modular, contextual y útil** dentro de flujos diseñados para la orientación educativa.

Las dificultades encontradas —desde la gestión de estados, el agotamiento de tokens, el desvío de conversaciones o la necesidad de validar correctamente la información antes de avanzar— han sido abordadas con un enfoque iterativo, creativo y orientado a soluciones. Este proceso ha reforzado no solo mis habilidades técnicas, sino también mi capacidad de **análisis, diseño estructurado y toma de decisiones bajo incertidumbre**.

Hoy, no solo comprendo mejor las posibilidades de n8n, sino que vislumbro en esta plataforma un **enorme potencial para construir sistemas inteligentes, escalables y altamente personalizados**, especialmente cuando se integran con bases de datos y servicios externos. La versatilidad de n8n como entorno de automatización, sumada al poder de los LLM, abre un horizonte de posibilidades para crear herramientas útiles y humanas.

En definitiva, este proyecto ha sido una oportunidad para crecer como desarrollador y como diseñador de experiencias orientadas al usuario. La evolución no solo está en el código o en los workflows, sino también en la forma en que ahora concibo y resuelvo los problemas. Y eso, sin duda, es uno de los aprendizajes más valiosos que me llevo.