AstraTask Engine

Sistema de IA que centraliza tareas académicas dispersas en un solo flujo en Telegram

RESUMEN EJECUTIVO:

Karol, estudiante de primer cuatrimestre de TI, recibe tareas dispersas en Classroom, Bécalos, Coursera y avisos verbales, olvidando actividades clave y perdiendo porcentaje de calificación. La solución es un bot de Telegram conectado a Make, Google Gemini y Google Sheets que registra automáticamente cada tarea enviada en lenguaje natural, calcula prioridad y fecha límite. Con esto reduce el tiempo de organización diaria de **30–40 minutos a menos de 5 minutos** y pasa de **3–5 tareas olvidadas al mes a 0**, disminuyendo el riesgo de reprobar por no entregar proyectos de 30–50% de la calificación. En un piloto de 2 semanas, el sistema registró 11 entregas con 0 tareas olvidadas y mantuvo Classroom sin pendientes vencidos.

Nombre completo: Karol Uriel Morales Ramírez
Correo electrónico: moralesuriel058@gmail.com
Rol profesional actual: Estudiante de Tecnologías de la Información e Innovación Digital
Empresa/organización: Sin empresa — proyecto académico individual (Diplomado IA)
Años de experiencia en tu área: En formación, con primeros proyectos prácticos usando IA y automatización

PERFIL PROFESIONAL:

Soy estudiante de Tecnologías de la Información e Innovación Digital, actualmente en formación y desarrollando mis primeros proyectos con herramientas de inteligencia artificial. Me interesa especialmente la automatización de tareas repetitivas, la organización de información y la integración de IA en el trabajo diario. Este proyecto representa mi primer caso formal de diseño, implementación y validación de una solución de IA para un usuario real.

DECLARACIÓN DE AUTORIDAD:

Tengo acceso directo al usuario real y a su proceso actual, además de las herramientas de IA del diplomado, lo que me permite diseñar, probar e implementar una solución funcional, medible y basada en datos reales.

MOTIVACIÓN DEL PROYECTO:

Este proyecto es el puente entre lo que estudio y los problemas reales que enfrentan las personas en su trabajo diario. Me permite construir portafolio con resultados medibles y demostrar que puedo aplicar IA más allá de la teoría. También sienta las bases de los servicios y soluciones que quiero ofrecer profesionalmente en el futuro.

SECCIÓN 2 – HISTORIA DE TRABAJO

2.1 Contexto del usuario y del problema

Recién egresado del bachillerato, el estudiante ingresa a la universidad en una ingeniería, donde la carga académica se vuelve aproximadamente el doble de lo que estaba acostumbrado a manejar. Antes no tenía el hábito de anotar todas sus tareas, y esa falta de sistema se vuelve crítica en el contexto universitario: algunos profesores publican actividades en Google Classroom, otros solo dan las instrucciones de forma verbal en clase y el estudiante debe apuntarlas por su cuenta.

Actualmente cursa 7 materias en la universidad y, además, se inscribe a cursos externos como Bécalos y BeChallenge English, lo que prácticamente duplica su carga de trabajo semanal. Esta combinación de múltiples fuentes de tareas, falta de centralización y ausencia de un método sólido de registro provoca que algunas actividades simplemente no se registren ni se recuerden a tiempo. Un ejemplo concreto es la no entrega de una actividad con un porcentaje considerable de la calificación, evidenciada en las capturas de pantalla donde aparece como “no entregada” pese a su peso en la evaluación de la materia.

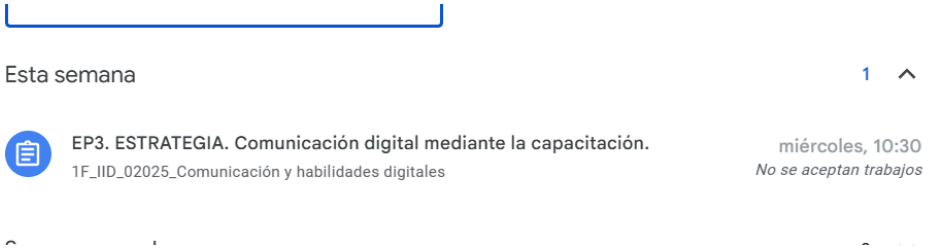


Figura 1 - Evidencia de tarea no entregada con peso del 50% en calificación en la materia

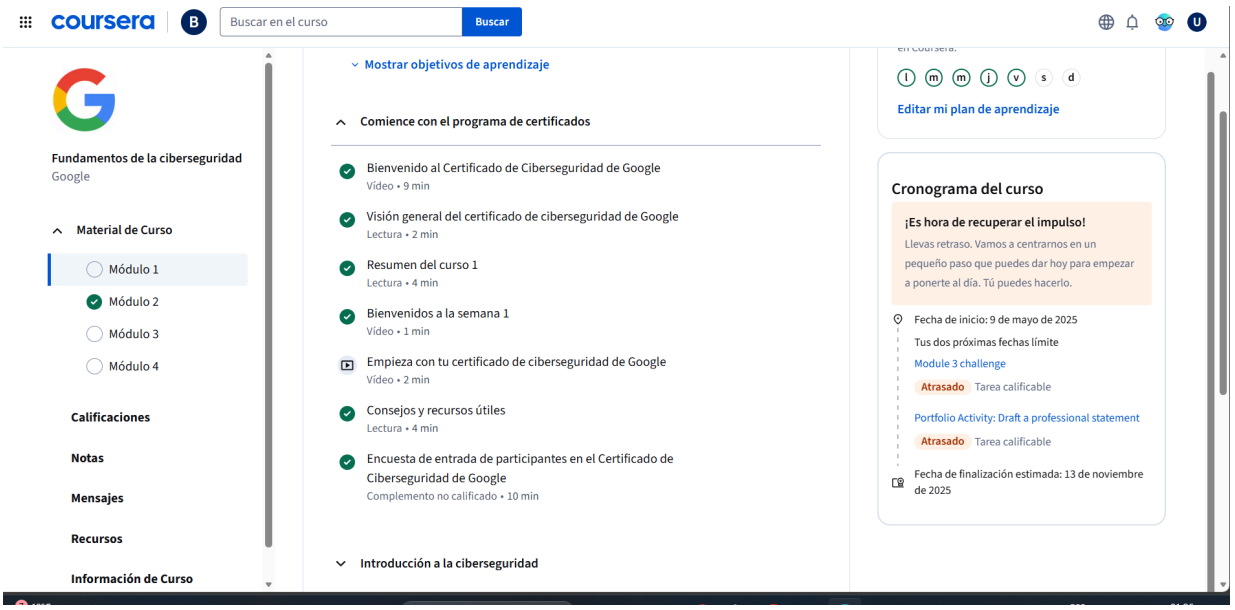


Figura 2 - Evidencia de plataforma coursera de programa bécalos, con actividades con retraso.

2.2 JOB STORY:

Cuando termino mis clases del día o me siento a revisar qué pendientes tengo en Google Classroom, Bécalos, Coursera y las tareas que los profesores dieron solo de forma verbal, quiero registrar cada nueva entrega (tareas, proyectos y evidencias de conocimiento) enviando un solo mensaje en lenguaje natural a un bot de IA en Telegram que la interprete, estructure (materia, tipo, fecha límite, esfuerzo) y la guarde automáticamente en una lista centralizada, para que deje de olvidar entre 3 y 5 entregas al mes, reduzca mi tiempo diario de organización de 30–40 minutos a menos de 5-6 minutos y tenga visibilidad completa de todas mis entregas

2.3 JUSTIFICACIÓN IA:

Ventaja vs Excel/Google Sheets:
Excel/Sheets solo almacenan datos; tendría que capturar manualmente cada campo de cada tarea (materia, tipo, título, fecha, esfuerzo). Con IA, basta un mensaje en lenguaje natural y el modelo extrae y limpia la información, rellenando automáticamente las columnas correctas. Esto reduce fricción, errores por cansancio y el tiempo invertido en “dar formato” a cada registro.

Ventaja vs proceso manual optimizado (libreta/notas):
Aunque usara libreta o notas bien organizadas, sigo dependiendo de mi memoria y disciplina para anotar todo en el momento exacto. Cuando el profesor habla rápido, cambio de clase o reviso varias plataformas, es fácil que algo no se anote o quede

incompleto. La IA permite que una única acción (mandar un mensaje al bot) dispare todo el flujo: interpretación, registro estructurado y actualización automática, evitando que tareas importantes se pierdan por descuido o saturación.

Ventaja vs herramientas existentes sin IA (apps de tareas genéricas):

Apps como To Do, Notion o Trello requieren adaptarme a su interfaz (crear tarea, elegir lista, fecha, etiquetas) y no están integradas nativamente con mi realidad académica (Classroom, Bécalos, Coursera, instrucciones verbales). Mi solución usa IA para entender cómo realmente escribo mis pendientes, funciona directamente en Telegram (app que ya uso diario) y se conecta con Make + Google Sheets para automatizar el guardado y la priorización, sin obligarme a cambiar de herramientas ni de hábitos.

2.4 CONTEXTO DE ADOPCIÓN:

Frecuencia:
El problema ocurre prácticamente todos los días. El estudiante recibe entre 2 y 4 nuevas tareas diarias, sumando aproximadamente 10–20 actividades por semana entre las 7 materias universitarias, Bécalos/BeChallenge English y cursos como Coursera (tareas, proyectos, quizzes y entregas semanales).

Herramientas actuales:

- Google Classroom (tareas y avisos de varias materias).
- Plataformas de Bécalos / BeChallenge (retos y entregas).
- Coursera u otras plataformas de certificación (módulos con deadlines).
- Instrucciones verbales de profesores en clase.
- WhatsApp u otros chats (recordatorios informales).
- Notas del celular / memoria (registro parcial y disperso).

Integración con el workflow diario:

Cada vez que se asigna una tarea nueva (en clase, en Classroom, en Bécalos o en Coursera), el estudiante puede tomar 10–30 segundos para enviar un mensaje al bot de Telegram con la descripción de la actividad. La IA interpreta ese mensaje, estructura los datos y los guarda en una hoja de

cálculo centralizada, donde luego se consultan mediante comandos como `/resumen` y se actualizan con `/hecho [id]`. Así, la solución se incrusta en la rutina existente sin añadir nuevas plataformas complejas y convierte un flujo caótico de múltiples fuentes en un sistema único y controlable.

Este patrón de tareas olvidadas y entregas tardías es lo que la solución de IA busca eliminar.

2.5 Proceso Design Sprint aplicado

- **Understand:** Durante el mes previo a AstraTask Engine revisé mi historial en Classroom, Bécalos y Coursera y detecté entre **3 y 5 tareas no entregadas o entregadas tarde**, incluyendo un proyecto con peso de **30–50%** de la calificación. Además, durante **5 días** cronometré el tiempo diario de organización de tareas, obteniendo un rango de **30–40 minutos** al día.

- **Diverge:** Consideré alternativas como usar solo Google Calendar, una libreta organizada, plantillas en Excel y apps de tareas genéricas (To Do, Notion, Trello).

- **Converge:** Elegí un **bot de Telegram + IA (Google Gemini) + Make + Google Sheets**, porque se integra con mis hábitos actuales, requiere solo un mensaje en lenguaje natural por tarea y automatiza el registro estructurado.

- **Prototype:** Construí un primer escenario en Make que conecta Telegram, Google Gemini y Google Sheets para probar los comandos **/registrar, /resumen y /hecho** con tareas reales de mis materias y cursos externos.

- **Test (piloto inicial + plan a cuatrimestre):** Se realizó un piloto de 2 semanas y se planea medir un cuatrimestre completo AstraTask Engine se comparara tareas olvidadas, tiempo de organización y puntos de calificación perdidos antes vs después, usando el mismo método de medición.

SECCIÓN 3 – USUARIO OBJETIVO ESPECÍFICO

3.1 PERFIL USUARIO:

- **Nombre:** Karol Uriel Morales Ramírez
- **Rol:** Estudiante de Ingeniería en Tecnologías de la Información e Innovación Digital (1.er año)
- **Empresa:** Universidad Politécnica del Estado de Morelos (UPEMOR) – institución pública de educación superior
- **Equipo:** Trabajo principalmente individual; colabora en equipos de 3–5 compañeros según la materia, sin personas a su cargo
- **Responsabilidades:**
    - Asistir a clases y cumplir con tareas, proyectos y exámenes de 7 materias
    - Gestionar actividades y entregas de programas externos (Bécalos, BeChallenge English, Coursera)
    - Organizar su calendario académico y priorizar entregas según peso en la calificación
    - Mantener un registro funcional de tareas para evitar reprobar o bajar promedio por no entregar

3.2 COMPETENCIAS TÉCNICAS:

- **Herramientas diarias:** Google Classroom, Google Drive/Docs/Sheets, Coursera, plataforma Bécalos/BeChallenge, Telegram, WhatsApp, navegador web, Make.com, ChatGPT/Gemini para apoyo puntual, notas del celular.
- **Nivel IA/tech:** 3/5 – Usuario intermedio. Maneja con soltura plataformas digitales, ya ha construido escenarios básicos en Make y probado modelos de IA generativa, pero sigue en etapa de aprendizaje y experimentación.
- **Experiencia automatización:** Ha creado y depurado un flujo en Make que conecta Telegram, Google Sheets y modelos de IA para registrar tareas y devolver resúmenes. Ha hecho pruebas reales con sus propias actividades

académicas y entiende el concepto de triggers, rutas y módulos.

3.3 CONTEXTO DEL PROBLEMA:

- **Frecuencia problema:** Todos los días lectivos. Cada día recibe entre 2 y 4 nuevas actividades (tareas, prácticas, proyectos, quizzes), repartidas entre universidad y cursos externos.

- **Tiempo semanal invertido:** Aproximadamente 30–40 minutos diarios solo en revisar plataformas y organizar pendientes, lo que suma unas **3–4 horas semanales** dedicadas exclusivamente a entender qué tiene que entregar y cuándo.

- **Herramientas que debe integrar la solución:**

    - Entrada principal: Telegram (bot)
    - Almacenamiento y vista estructurada: Google Sheets
    - Automatización: Make.com
    - Fuentes de información: Google Classroom, Bécalos/BeChallenge,
    - Coursera, instrucciones verbales en clase
    -
- **Restricciones:**

    - Presupuesto casi nulo: debe trabajar con planes gratuitos o de muy bajo costo.
    - Tiempo limitado por carga académica y trabajo adicional.
    - Sin permisos administrativos sobre las plataformas de la universidad (no puede modificar configuraciones de Classroom, solo consumir la información como estudiante).

3.4 VALIDACIÓN USUARIO:

- **¿Problema frecuente?** Sí. Se repite a diario: la combinación de múltiples plataformas y falta de sistema ha generado 3–5 tareas olvidadas al mes, incluyendo actividades con peso importante en la calificación.

- **¿Autoridad adopción?** Sí. Es su propio flujo de estudio; puede decidir qué herramientas usar para organizarse sin necesitar aprobación institucional.

- **¿Está dispuesto a invertir tiempo en aprender?** Sí. Ya invirtió tiempo en el Diplomado de IA, en configurar Make y en diseñar un bot de Telegram; está

motivado por evitar más pérdidas de calificación.

- **¿Puede medir impacto?** Sí. Puede medir:
    - Número de tareas olvidadas antes vs después
    - Tiempo diario dedicado a organización (cronómetro)
    - Porcentaje de tareas entregadas a tiempo por periodo
      Todo esto es registrable en Google Sheets y comprobable con capturas de Classroom y plataformas externas.

SECCIÓN 4 – ESTADO ACTUAL DOCUMENTADO

4.1 Proceso actual paso a paso

PROCESO ACTUAL PASO A PASO:

1. Revisa curso por curso en Google Classroom para ver si hay nuevas tareas
   – Tiempo: 10 min
   – Herramienta: Navegador + Google Classroom
   – Frustración: 3/5
   – Valor agregado: Medio (necesario, pero repetitivo)
   – Error estimado: 10% (algún curso se revisa por encima o se salta una publicación).

2. Abre la plataforma de Bécalos / BeChallenge para revisar retos y entregas
   – Tiempo: 5 min
   – Herramienta: Plataforma Bécalos / BeChallenge
   – Frustración: 3/5
   – Valor agregado: Medio
   – Error estimado: 10% (algún reto pasa desapercibido si la plataforma carga lento o se distrae).

3. Revisa Coursera para ver módulos, quizzes y actividades con fecha límite
   – Tiempo: 5 min
   – Herramienta: Coursera
   – Frustración: 2/5
   – Valor agregado: Medio
   – Error estimado: 5% (algún deadline se ve pero no se registra).

4. Intenta recordar tareas que los profesores dieron solo de forma verbal y las anota en libreta o notas del celular
   – Tiempo: 8 min
   – Herramienta: Libreta física / app de notas
   – Frustración: 4/5
   – Valor agregado: Bajo (mucho esfuerzo para un registro poco confiable)
   – Error estimado: 30% (se le olvidan detalles o tareas completas).

5. Reescribe o acomoda las tareas anotadas en una lista general (libreta, notas o pseudo-calendario)
   – Tiempo: 7 min
   – Herramienta: Libreta / notas / calendario del celular
   – Frustración: 4/5
   – Valor agregado: Bajo–Medio (organiza un poco, pero sigue desordenado)
   – Error estimado: 20% (fechas mal copiadas, tareas repetidas o sin prioridad clara).

6. Al final del día o al inicio del siguiente, vuelve a revisar la lista para decidir qué hacer primero
   – Tiempo: 5 min
   – Herramienta: Libreta / notas / memoria
   – Frustración: 3/5
   – Valor agregado: Medio (da una idea general, pero no prioriza bien)
   – Error estimado: 15% (subestima duración o importancia de ciertas tareas).

Tiempo total diario del proceso actual: ~35 minutos dedicados solo a revisar plataformas, recordar tareas y organizarlas de forma manual y fragmentada.

4.2 Métricas baseline

- **Tiempo total del proceso (por día):**
  ~35 minutos/día para revisar plataformas y organizar pendientes.

- **Frecuencia:**
  Proceso repetido **6 días a la semana** (lunes a sábado).

- **Tiempo semanal total:**
  35 min/día × 6 días ≈ **210 minutos/semana** (3.5 horas) solo en “descubrir + organizar” qué hay que hacer.

- **Costo de oportunidad:**
  Esas 3.5 horas semanales podrían usarse para avanzar módulos de Bécalos/Coursera, preparar exámenes o proyectos, o descansar. En la práctica, es

tiempo invertido en logística en lugar de trabajo académico profundo.

- **Tasa de errores actual e impacto académico:**
  Entre **3 y 5 entregas olvidadas o mal gestionadas al mes**.
  Si se consideran ~40–50 entregas/mes (entre tareas, proyectos y evidencias), la tasa de error está entre **10% y 12.5%**.

  No todas las entregas tienen el mismo peso:

  - **Tareas y actividades cortas:** suelen valer entre **5% y 15%**.
  - **Proyectos parciales o finales:** pueden valer hasta **40–50%** de la materia.
  - **Evidencias de conocimiento / productos integradores:** según la materia, representan entre **30% y 40%** del total.

- Además, en la universidad del usuario se aplican las siguientes reglas:

  - Si una entrega se realiza **fuera de tiempo**, normalmente solo se califica hasta sobre el **80% del valor original** (por ejemplo, una actividad que valía 10 puntos pasa a tener un máximo de 8 aunque el contenido esté completo).
  - La **calificación mínima para aprobar** una materia es **7.0**; cualquier calificación menor se considera **reprobada**.
  - Cuando el promedio final queda por debajo de 7.0, el estudiante entra en una cadena de recuperación: primero **“recuperación”**, si falla pasa a **“extraordinario”**, y si vuelve a reprobar llega a **“recursamiento”** de la materia.

- En la práctica, esto significa que **una sola evidencia o proyecto de alto porcentaje no entregado o entregado tarde** puede reducir tanto el promedio que obligue al estudiante a entrar en recuperación, con el riesgo real de terminar recursando la materia si el patrón de desorganización se mantiene.

- **Satisfacción/frustración general:**
  2/5. La sensación predominante es de descontrol y saturación: aunque dedica tiempo a “organizarse”, sigue perdiendo entregas importantes (incluyendo proyectos y evidencias de alto porcentaje), ve afectado directamente su promedio y se expone al riesgo de entrar en recuperación, extraordinario o recursamiento por problemas de organización más que por falta de capacidad académica.

  Para obtener el baseline, revisé el **mes anterior sin AstraTask Engine** y conté entre **3 y 5 tareas no entregadas o entregadas tarde**, incluyendo al menos un proyecto con peso de **30–50%** de la calificación. Además, durante **5 días consecutivos** cronometré el tiempo dedicado cada día a revisar Classroom, Bécalos, Coursera e instrucciones verbales, obteniendo un rango de **30–40 minutos**

diarios. Estas mediciones sirven como punto de partida para comparar el desempeño de la solución.

4.3 Análisis de oportunidades

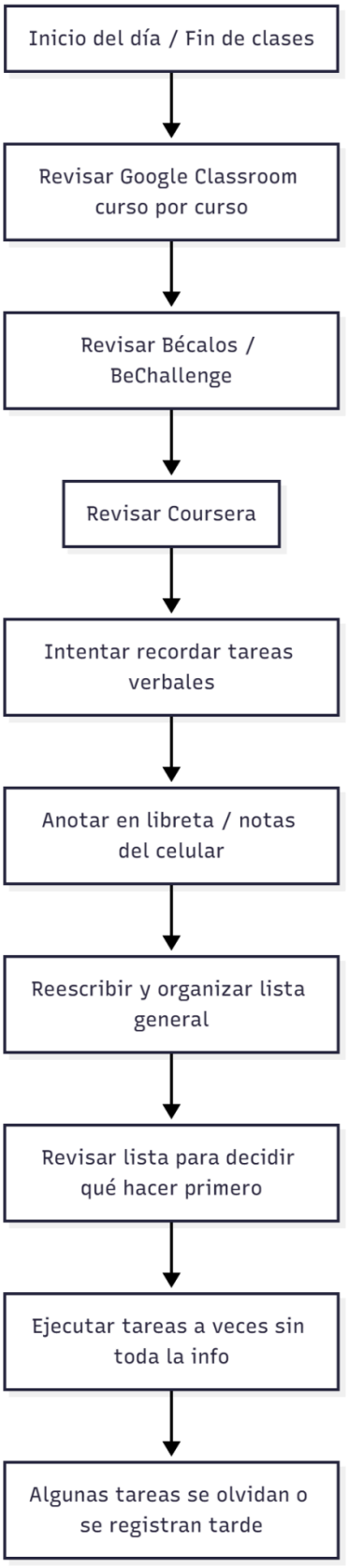
ANÁLISIS OPORTUNIDADES:

- Alta repetitividad + baja creatividad:
    - Revisar Classroom curso por curso.
    - Abrir Bécalos y Coursera solo para buscar si hay algo nuevo.
    - Reescribir listas de tareas en libreta/notas.
      Estos pasos son mecánicos y perfectos para la automatización.
- Decisiones automatizables (criterios claros):
    - Priorizar tareas por fecha de entrega y peso en la calificación.
    - Separar tipos de tarea (tarea, proyecto, examen, quiz) según palabras clave.
    - Asignar un nivel de esfuerzo estimado (1–10) a partir de la descripción.
- Búsqueda / análisis de información:
    - Localizar rápidamente todas las tareas por fecha.
    - Detectar cuáles vencen en la próxima semana.
    - Resumir en un solo mensaje lo más urgente del día.
- Validaciones con patrón:
    - Comprobar si ya se registró una tarea antes de volver a escribirla.
    - Verificar que cada tarea tenga materia, fecha y tipo antes de considerarla “lista”.
    - Marcar como “completada” cuando ya se entregó en Classroom/Bécalos/Coursera y sacarla del resumen diario.

Estas oportunidades muestran que el proceso actual tiene mucho trabajo mecánico, repetitivo y con alto riesgo de olvido, ideal para delegar a una combinación de IA + automatización.

4.4 Diagrama del proceso actual

Figura 3 - Proceso que sigue el usuario actualmente



SECCIÓN 5 – SOLUCIÓN PROPUESTA

SOLUCIÓN PROPUESTA :

Mi solución es un bot de Telegram conectado a Make y Google Gemini que transforma mensajes en lenguaje natural sobre tareas académicas diarias en registros estructurados dentro de Google Sheets. Make orquesta el flujo entre Telegram, Gemini y las hojas de cálculo, calculando prioridades y fechas límite. El estudiante consulta sus pendientes con comandos como /resumen y marca tareas completadas con /hecho, obteniendo una vista centralizada confiable de todas sus entregas universitarias y cursos externos.

ARQUITECTURA TÉCNICA:

- **IA Principal:** Google Gemini, usada como parser estricto para convertir mensajes en lenguaje natural en campos estructurados (materia, tipo, título, fecha, hora, esfuerzo).

- **Automatización:** Make, con un escenario que recibe los webhooks de Telegram, llama a Gemini, procesa la respuesta y escribe/lee en Google Sheets.

- **Interfaz usuario:** Bot de Telegram con comandos como `Registrar`, `/resumen` y `/hecho [id]`.

- **Integraciones:** API de Telegram Bot, módulo de Google Sheets en Make, con conexión a Google Workspace y a la API de Gemini.

- **Almacenamiento:** Google Sheets como base de datos principal de tareas, con columnas para id, timestamp, materia, tipo, título, fecha de entrega, esfuerzo, prioridad calculada y estado (pendiente/completado).

FLUJO DE DATOS:

1. **Input:** El estudiante envía desde Telegram un mensaje del tipo: `Registrar: Materia X, tipo tarea, título Y, fecha DD/MM, hora HH:MM, esfuerzo 7`.

2. **Procesamiento IA:** Make recibe el webhook, envía el texto a Google Gemini con un prompt de "parser estricto"; Gemini devuelve un objeto estructurado con los campos requeridos.

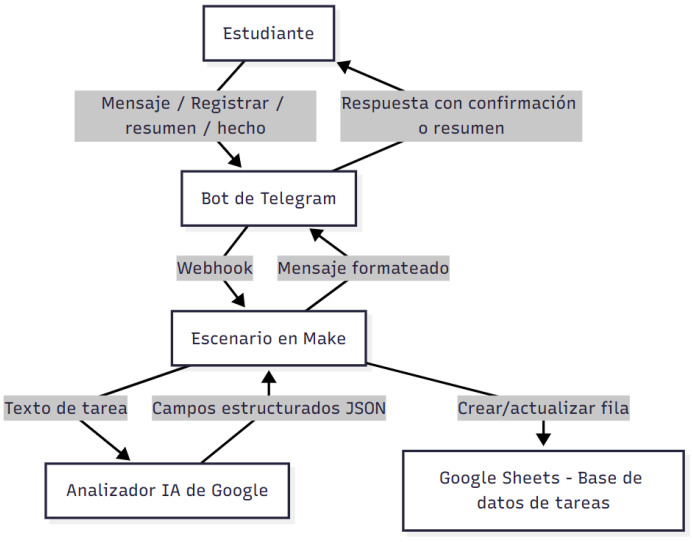
3. **Automatización:** Make valida y normaliza los datos (por ejemplo, convierte la fecha y hora a formato ISO), asigna un id único y crea o actualiza la fila correspondiente en Google Sheets.

4. **Output:** Para registros, el bot responde en Telegram con un mensaje de confirmación; para `/resumen`, Make lee las filas pendientes en Sheets, construye un listado legible y lo devuelve al usuario; para `/hecho [id]`, actualiza el estado de la tarea y confirma el cambio.

5. **Almacenamiento:** Google Sheets guarda el histórico de todas las tareas, permitiendo posteriormente calcular métricas como número de tareas olvidadas, porcentaje de entregas a tiempo y carga semanal.

DIAGRAMA ARQUITECTURA

Figura 4. Arquitectura técnica de la solución (Telegram + Make + Gemini + Google Sheets).



SECCIÓN 6 – IMPACTO ESPERADO CUANTIFICADO

MÉTRICAS DE EFICIENCIA

Tiempo de proceso
De **35 minutos diarios** dedicados a revisar Classroom/Bécalos/Coursera, buscar instrucciones verbales y organizar entregas, a **≈6 minutos diarios** usando el bot de IA (3.5 min para registrar tareas + 1–2 min para `/resumen` y 1 min para marcar completadas).
→ Reducción aproximada del **83%** del tiempo invertido en organización.

Volumen procesado (entregas organizadas por hora)
De **~5 entregas/hora** (≈3 entregas en 35 minutos, registro manual) a **~17 entregas/hora** (≈3 entregas en 10–11 minutos, 3.5 min c/u con el bot).
→ La capacidad potencial de organización aumenta aproximadamente **3.3x** (**≈230%** más volumen por hora).

Tasa de errores (entregas olvidadas o mal registradas)
De **10–12.5%** de entregas al mes (3–5 de 40–50) a un objetivo de **≤2%** (0–1 entrega mal gestionada al mes).
→ Mejora de **8–10 puntos porcentuales** en la tasa de error.

Satisfacción del usuario con su organización académica (escala 1–5)
De **2/5** (sensación de descontrol y saturación) a **4/5** (sensación de control y claridad sobre todas las entregas).
→ Mejora de **2 puntos** en percepción subjetiva.

IMPACTO ECONÓMICO

Supuesto: el valor de una hora del estudiante se aproxima a su ingreso como mesero de fin de semana (**~$25 MXN/h**).

Tiempo liberado
De **~3.5 h/semana** en organización manual (≈35 min/día × 6 días) a **~0.6 h/semana** con el bot (≈6 min/día × 6 días).
→ Se liberan aproximadamente **3 horas por semana** que pueden usarse para estudiar, avanzar cursos externos o descansar.

Valor del tiempo liberado
3 h/semana × $25 MXN/h ≈ **$75 MXN/semana**
≈ $300 MXN/mes
≈ $1,200 MXN por cuatrimestre (4 meses, ~16 semanas).

ROI estimado (ejemplo con costo simbólico de herramientas)
Si se considera un costo de **$100 MXN/mes** en herramientas (IA + automatización), en un cuatrimestre serían **$400 MXN**.
Beneficio estimado en tiempo: **$1,200 MXN**
→ ROI aproximado de **200%** en un cuatrimestre.

Payback period
Con un beneficio de **~$75 MXN/semana** y un costo acumulado de **$400 MXN**, el punto de equilibrio se alcanza alrededor de las **5–6 semanas** de uso continuo.

MÉTRICAS DE ADOPCIÓN

Tiempo de implementación
Desarrollo y configuración inicial del flujo Telegram + Make + Gemini + Google Sheets en **2 semanas**, con **1 semana adicional** para pruebas y ajustes menores.

Curva de aprendizaje del usuario
Se espera que el usuario domine los comandos básicos (`/registrar`, `/resumen`, `/hecho`) y el formato de mensajes en **2–3 días** de uso real.

Frecuencia de uso esperada
Uso del bot entre **1 y 3 veces por día**, según número de nuevas entregas, y al menos **1 comando `/resumen` diario** para revisar pendientes.

Indicadores de éxito temprano (primera semana)

- 100% de las nuevas entregas registradas mediante el bot.
- Tiempo diario de organización **≤10 minutos** desde el primer ciclo de uso.
- 0 entregas olvidadas durante la primera semana de prueba.

MÉTRICAS DE CALIDAD

Precisión de la IA al parsear mensajes (campos materia/tipo/fecha/esfuerzo)
Meta de **≥95% de precisión** en la extracción de campos a partir de mensajes en lenguaje natural, usando prompts estrictos y ejemplos en Gemini.

Consistencia de resultados
Variación máxima aceptable de **±5%** en la precisión del parseo entre diferentes días y tipos de mensajes, siempre que se mantenga el formato general esperado.

Mejora continua

- Ajustar el prompt y los ejemplos de Gemini en función de los errores detectados en las primeras **2–3 semanas**.

- Revisar periódicamente la base de datos de Google Sheets para identificar patrones de error (materias o tipos de entrega más problemáticos) y reforzar esos casos en el diseño del prompt.

- Medir cada cuatrimestre la tasa de errores y el tiempo de organización para asegurar que se mantiene el objetivo de **0–1 entrega mal gestionada al mes** y **≈5–6 min/día** de organización.

RESULTADOS PILOTO MEDIDOS (2 SEMANAS)

En un piloto inicial de **2 semanas** de uso real:

- **Tareas registradas:** 11 entregas (5 en la primera semana y 6 en la segunda) capturadas íntegramente mediante el bot.

- **Tiempo promedio de registro por tarea: ~3.5 minutos** cronometrados.

- **Tiempo diario de organización (registro + `/resumen` + marcado de completadas):** entre **5 y 6 minutos**, cercano al objetivo planteado de **<5 min/día**.

- **Entregas olvidadas o tardías: 0 de 11**; Classroom se mantuvo “limpio”, sin tareas vencidas sin entregar.

VALIDACIÓN DE REALISMO

Las reducciones de tiempo y errores se basan en el **baseline medido** (≈35 min/día y 3–5 entregas olvidadas/mes) y en el desempeño real del flujo ya implementado (Telegram + Make + Gemini + Google Sheets) durante el piloto de 2 semanas.

Las métricas de precisión (≥95%) son coherentes con el desempeño típico de modelos de IA modernos usados como parsers con prompts estructurados y ejemplos.

El timeline de **3 semanas** es compatible con el alcance actual: un solo usuario, una única hoja de cálculo y un escenario de Make ya funcional, donde solo se requieren ajustes incrementales en vez de una reescritura completa.

SECCIÓN 7 – FACTIBILIDAD Y RIESGOS

FACTIBILIDAD TÉCNICA (1–5)

- **Complejidad IA: 3/5**
  La IA solo debe interpretar texto estructurado (“Registrar: Materia…”) y devolver JSON con campos fijos. No requiere RAG, memoria larga ni decisiones complejas. La complejidad es media: el reto está en diseñar un prompt robusto, no en el modelo en sí.

- **Integraciones: 4/5**
  Todas las piezas son estándar y ya probadas: Telegram Bot + Webhook de Make + Google Gemini + Google Sheets. Las integraciones son directas vía módulos nativos, sin APIs raras ni autenticaciones exóticas. El riesgo existe en mapeos de columnas/IDs, pero es manejable.

- **Timeline (3 semanas): 3/5**
  El flujo principal ya existe, pero hay trabajo pendiente en: estabilizar el parser, ajustar los comandos, limpiar la base de datos y probar casos límite. Tres semanas alcanzan *solo* si se prioriza bien y no se agregan features

extra.

- **Experiencia personal: 3/5**
  El estudiante ya ha construido escenarios en Make y ha peleado con errores reales (IDs, routers, fórmulas), pero sigue en curva de aprendizaje y a veces se enreda en la complejidad. Tiene la capacidad, pero necesita disciplina para no sobrecomplicar.

- **FACTIBILIDAD GENERAL: 4/5**
  Técnicamente es totalmente alcanzable con el stack y el plazo, siempre que se mantenga el alcance acotado al registro/resumen/hecho y se deje fuera cualquier cosa “nice to have” que pueda romper el timeline.

TOP 5 RIESGOS

1. **Riesgo técnico – El parser de IA devuelve campos incompletos o mal mapeados**
    - **Probabilidad:** Media
    - **Impacto:** Alto (tareas mal registradas = errores directos en la base)
    - **Mitigación:** Diseñar prompt estricto con ejemplos claros; probar al menos 30 mensajes reales distintos; registrar logs de errores y ajustar el prompt según patrones.
    - **Plan B:** Si la IA falla en ciertos casos, definir un formato más rígido (campos separados por “;”) y validar en Make que todo venga lleno antes de guardar. Si falta algo, devolver mensaje de error al usuario.
2. **Riesgo de integración – Fallos entre Make, Google Sheets y Telegram (IDs, columnas, permisos)**
    - **Probabilidad:** Media

- **Impacto:** Medio–Alto (rompe /hecho, /resumen o duplicará filas)
- **Mitigación:** Congelar la estructura de la hoja (no mover columnas), documentar los IDs y rangos usados, probar los comandos con varios registros de prueba antes de uso real.
- **Plan B:** Si una automatización se rompe, permitir un comando de emergencia para exportar los datos actuales y corregir manualmente en Sheets mientras se repara el flujo.

3. **Riesgo de tiempo – No completar pruebas y estabilización antes del video/entrega**
   - **Probabilidad:** Alta (hay otras materias, proyectos y exámenes)
   - **Impacto:** Alto (demo inestable, errores en vivo, mala evaluación)
   - **Mitigación:** Definir un MVP claro y bloquear tiempo específico en calendario solo para este proyecto. Terminar el núcleo funcional en la semana 1–2 y usar la semana 3 solo para pruebas y mediciones.
   - **Plan B:** Si no se llega a estabilizar todo el flujo, grabar el demo mostrando el camino más estable (registrar + resumen) y dejar /hecho o features extra como “en desarrollo”.
4. **Riesgo de adopción – El usuario deja de usar el bot por fricción o flojera**
   - **Probabilidad:** Media
   - **Impacto:** Medio–Alto (sin uso real no hay métricas ni validación)
   - **Mitigación:** Diseñar comandos mínimos y rápidos, dejar el bot anclado en Telegram, crear un recordatorio diario a cierta hora para usar `/resumen` y revisar pendientes.
   - **Plan B:** Si el bot no se usa de forma natural, complementar con un flujo alterno (por ejemplo, un formulario simple de Google Forms conectado a la misma hoja) para no perder por completo la centralización.

5. **Riesgo de costos/limitaciones de plataforma (créditos de Make o IA)**
    - **Probabilidad:** Media
    - **Impacto:** Medio (el flujo podría dejar de ejecutarse si se acaban créditos)
    - **Mitigación:** Optimizar el escenario para reducir ejecuciones innecesarias; evitar loops; no llamar a la IA cuando el mensaje no sea de registro; usar filtros en Make.
    - **Plan B:** Si los créditos se agotan, migrar temporalmente a un flujo más simple sin IA (campos predefinidos en el mensaje, parseo directo en Make) hasta poder reactivar el módulo de IA.

CRITERIOS DE ÉXITO MÍNIMO

- **Must Have (obligatorio para considerar éxito):**
    - Comando de registro que permita crear entregas desde Telegram y las guarde correctamente en Google Sheets (materia, tipo, título, fecha, esfuerzo).
    - Comando de resumen que muestre, al menos, las entregas pendientes ordenadas por fecha.
    - Tasa de errores en registro ≤ 1 entrega mal registrada por semana durante las pruebas.
- **Nice to Have (deseables, pero no críticos):**
    - Cálculo automático de prioridad usando esfuerzo + dificultad + peso.
    - Resúmenes filtrados por materia o por rango de fechas.
    - Mensajes de respuesta más “bonitos” (formato, emojis, etc.).
- **MVP absoluto (versión mínima que sigue agregando valor):**
    - Un solo comando de registro con IA que reciba el texto de la entrega y cree/actualice una fila en Google Sheets, más un comando `/resumen` básico que liste pendientes en texto plano.
      Aunque no haya prioridades, ni vistas avanzadas, con esto ya se

reduce fuertemente el riesgo de olvidar entregas.

PLAN DE CONTINGENCIA

- **Si falla la IA principal (Gemini no responde bien o cambia el formato):**
    - Cambiar temporalmente el flujo para que el usuario envíe la información en un formato semiestructurado (por ejemplo: `Materia; Tipo; Título; Fecha; Esfuerzo`) y parsear directamente en Make sin IA.
    - Alternativa extra: probar GPT-4 como parser con el mismo contrato de JSON si los créditos/limitaciones lo permiten.
- **Si falla la integración (Make ↔ Sheets ↔ Telegram):**
    - Mantener un respaldo manual de la hoja de cálculo.
    - Usar directamente Google Sheets durante 1–2 días mientras se repara el escenario.
    - Documentar una guía rápida para reconfigurar el webhook y los módulos críticos sin rehacer todo.
- **Si falla la adopción del usuario (no se usa el sistema de forma constante):**
    - Reducir aún más la complejidad de los comandos (dejar solo `/registrar` y `/resumen`).
    - Programar un recordatorio diario en Telegram o en el calendario para usar el bot.
    - Si aun así no se adopta, usar los datos que sí se tengan para el video y el PRD, y dejar claro que el siguiente paso del proyecto es trabajar en hábitos y diseño de experiencia de uso.