UNIVERSIDAD AUTÓNOMA GABRIEL RENÉ MORENO

Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones

Sistema Multi-Red Social con LLM

MATERIA: tópicos avanzados de programacion

Estudiante	**Registro**
Velasquez Paz Arnulfo	220030383

1. Introducción

1.1. Objetivo del Documento

El presente documento constituye el entregable principal de la **Clase 1: Investigación y Feasibility Study**, en el marco del proyecto "Adaptación Automática de Contenido para Redes Sociales".

El objetivo primordial de este informe es presentar un análisis exhaustivo de viabilidad técnica, tecnológica y operativa para el desarrollo del sistema propuesto. Este estudio sirve como pilar fundamental para la toma de decisiones estratégicas en las fases subsecuentes del proyecto.

Para lograr esto, el documento detalla los hallazgos de la investigación de los componentes críticos, a saber: las Interfaces de Programación de Aplicaciones (APIs) de las cinco redes sociales objetivo, y los modelos de lenguaje de gran escala (LLM) disponibles para la adaptación de contenido.

Finalmente, este estudio culmina con una **propuesta de arquitectura de sistema y un stack tecnológico**. Dicha propuesta está diseñada para ser **viable, pragmática y alcanzable** dentro del cronograma estipulado de seis clases, priorizando la simplicidad de implementación y la rápida iteración, sin sacrificar la funcionalidad central del proyecto. Este documento servirá como la hoja de ruta técnica validada para el desarrollo del prototipo.

1.2. Descripción del Problema

En el panorama digital contemporáneo, la gestión efectiva de la presencia en múltiples plataformas de redes sociales se ha convertido en un desafío operativo de primer orden para empresas, marcas y creadores de contenido. El problema central no radica únicamente en la necesidad de publicar, sino en la **fragmentación de audiencias y formatos**.

Cada plataforma social (Facebook, Instagram, LinkedIn, TikTok y WhatsApp) opera bajo sus propias reglas, con características únicas de formato, límites de caracteres, tonos de comunicación y expectativas de la audiencia:

- **LinkedIn** exige un tono profesional y contenido de valor estructurado.
- **Instagram** se centra en lo visual, con textos (captions) más breves y un uso intensivo de hashtags.
- **TikTok** depende de ganchos (hooks) de video cortos, un tono juvenil y hashtags de tendencia.
- **Facebook** permite una mayor flexibilidad en la longitud del contenido.
- **WhatsApp** requiere un tono conversacional, directo y conciso.

La adaptación manual de una misma pieza de contenido (como una noticia, el lanzamiento de un producto o un anuncio de evento) para cada una de estas redes es un proceso **manual, lento, costoso y altamente propenso a inconsistencias**. Requiere que un operador humano no solo copie y pegue, sino que re-escriba y re-contextualice el mensaje cinco veces distintas, un cuello de botella que limita la agilidad y el alcance.

Este proyecto busca resolver esta ineficiencia crítica mediante el diseño de un portal web centralizado que utiliza **automatización inteligente**. El sistema propuesto ingerirá un contenido base y, mediante el uso de un LLM, generará automáticamente las adaptaciones optimizadas para cada red, coordinando finalmente su publicación.

1.3. Alcance del Estudio de Viabilidad

El alcance de este estudio de viabilidad se limita estrictamente a la fase de investigación, análisis y planificación correspondiente a la **Clase 1** del cronograma del proyecto. El propósito es definir *qué* es posible construir y *cómo* se construirá de la manera más eficiente.

Este documento evaluará la factibilidad técnica de integrar las cinco plataformas sociales designadas y el componente de inteligencia artificial. El análisis se centrará en los siguientes puntos clave:

1. **Análisis de APIs de Redes Sociales:**
    - Investigación de la documentación oficial, requisitos de acceso (verificación de aplicaciones), procesos de autenticación (OAuth 2.0) y limitaciones (rate limits) de las APIs de Meta (Facebook, Instagram), LinkedIn, TikTok y WhatsApp.
    - Identificación de los principales "bloqueadores" técnicos (ej. complejidad de la API de TikTok).
    - Definición de **estrategias de mitigación y "Planes B"** (notablemente para TikTok y WhatsApp) que aseguren la viabilidad del proyecto sin comprometer el cronograma.
2. **Análisis de Modelos LLM:**
    - Comparación de las arquitecturas de LLM disponibles (APIs remotas de pago vs. modelos locales auto-alojados).
    - Justificación de la selección de un modelo local (**Gemma vía Ollama**) como la solución más pragmática, eliminando costos, dependencia de red y claves de API, en favor de un control total en el entorno de desarrollo.
3. **Propuesta de Arquitectura y Stack Tecnológico:**
    - Definición de un stack tecnológico completo (Backend: FastAPI, Frontend: React, Base de Datos: postgre).
    - Diseño de una arquitectura de sistema **simplificada y viable**, priorizando una implementación de **procesamiento** de sistemas de colas asíncronas.
    - Presentación del esquema de base de datos inicial (compatible con postgre) y el flujo de datos end-to-end propuesto.

Exclusiones del Alcance: Este documento *no* incluye la implementación de código fuente, el diseño de ingeniería de prompts de LLM, el diseño de la interfaz de usuario (UI/UX) ni la configuración de despliegue. Dichos elementos constituyen entregables de clases posteriores.

2. Investigación y Análisis de APIs (Viabilidad de Integración)

Esta sección detalla la investigación de viabilidad para cada una de las interfaces de programación de aplicaciones (APIs) de las plataformas sociales objetivo. El análisis se centra en los requisitos de acceso, los procesos de autenticación, los puntos finales (endpoints) clave y las estrategias de implementación pragmáticas para asegurar la finalización del proyecto dentro del cronograma de seis clases.

2.1. Meta Business API (Facebook & Instagram)

La integración con Facebook e Instagram se gestiona de forma unificada a través de la plataforma de Meta para Desarrolladores.

2.1.1. Documentación Oficial y Ecosistema

La interacción con ambas plataformas se centraliza mediante la **Meta Graph API**. Esta es una API masiva basada en HTTP que permite a las aplicaciones leer y escribir datos en el ecosistema de Meta. La documentación oficial es extensa y está bien mantenida, e incluye una herramienta vital para el desarrollo, el **Graph API Explorer**, que permite probar consultas y obtener tokens de acceso temporales. El ecosistema es maduro pero complejo, con un fuerte énfasis en la seguridad, los permisos y la verificación de aplicaciones.

2.1.2. Proceso de Configuración: Requisitos

Para interactuar con la API se deben cumplir los siguientes requisitos previos:

1. **Cuenta de Desarrollador de Meta:** Es necesario registrarse en el portal de `developers.facebook.com`.
2. **Creación de una App de Meta:** Se debe crear una aplicación de tipo "Negocio" (Business).
3. **Página de Facebook:** La aplicación debe estar vinculada a una Página de Facebook existente (para las publicaciones de Facebook).
4. **Cuenta de Instagram Business:** Dicha Página de Facebook debe estar correctamente conectada a una cuenta de Instagram de tipo "Business" o "Creator" (para las publicaciones de Instagram).

2.1.3. Sistema de Autenticación: OAuth 2.0

El acceso a la API está protegido por el protocolo **OAuth 2.0**. Este es un estándar de la industria que requiere un flujo de autorización explícito por parte del usuario. En un escenario de producción, el usuario sería redirigido a una página de inicio de sesión de Facebook para otorgar permisos a nuestra aplicación. La aplicación recibiría entonces un *token de acceso* de corta duración, que debe ser intercambiado por un *token de larga duración* (60 días) para su almacenamiento y uso futuro. Este es un proceso de múltiples pasos que añade una complejidad de desarrollo considerable.

2.1.4. Estrategia de Implementación (Enfoque Pragmático)

Dada la complejidad del flujo OAuth 2.0 completo y los estrictos requisitos de verificación de aplicaciones, se adopta una estrategia de implementación en dos fases para garantizar la viabilidad del proyecto:

- **Fase 1 (Desarrollo y Demo - Alcance del Proyecto):** Para el desarrollo inicial y la presentación final, **se evitará implementar el flujo OAuth 2.0 completo**. En su lugar, utilizaremos **Tokens de Acceso de Desarrollador** generados manualmente a través de la herramienta **Graph API Explorer**. Estos tokens tienen una duración corta (1-2 horas) pero otorgan todos los permisos necesarios para que la cuenta del desarrollador (que administra la App y la Página) pueda publicar. Esto nos permite validar rápidamente la lógica de publicación sin construir la infraestructura de autenticación.
- **Fase 2 (Producción - Fuera del Alcance):** Como una mejora futura, se implementaría el flujo OAuth 2.0 completo, permitiendo que cualquier usuario conecte sus propias cuentas de Facebook e Instagram. Esto también requeriría pasar por el proceso de "App Review" de Meta.

2.1.5. Permisos (Scopes) Requeridos

Para que la aplicación funcione, se deben solicitar permisos específicos (scopes) durante el

proceso de autenticación. Los permisos mínimos necesarios son:

- **Para Facebook:** `pages_show_list` (para listar las páginas del usuario), `pages_read_engagement` y `pages_manage_posts` (para publicar en la página).
- **Para Instagram:** `instagram_basic` e `instagram_content_publish` (para publicar contenido en la cuenta de Instagram Business conectada).

2.1.6. Endpoints Clave

La publicación en cada plataforma utiliza un endpoint y un flujo distintos:

- **Facebook:** La publicación de texto e imágenes es relativamente directa. Se utiliza el endpoint `POST /{page-id}/feed` para publicaciones de solo texto, o `POST /{page-id}/photos` para publicaciones con imágenes, adjuntando la URL de la imagen y el texto en el cuerpo de la solicitud.
- **Instagram:** La publicación es un **flujo de 2 pasos**. No se puede publicar directamente.
    1. **Creación de Contenedor:** Primero, se debe hacer una llamada a `POST /{ig-user-id}/media` con la URL de la imagen y el *caption* (texto adaptado). La API devuelve un ID de contenedor (`creation_id`).
    2. **Publicación del Contenedor:** Segundo, se debe hacer una llamada a `POST /{ig-user-id}/media_publish` utilizando el `creation_id` obtenido en el paso anterior. Solo entonces la publicación aparece en Instagram.

2.1.7. Limitaciones y Riesgos (Rate Limits y Verificación de App)

- **Verificación de App (Riesgo Alto):** El mayor riesgo es el proceso de "App Review" de Meta. Para que una aplicación en modo "Live" obtenga permisos como `instagram_content_publish` de usuarios reales, debe pasar una estricta verificación. Nuestra estrategia de "Fase 1" (usar tokens de desarrollador en modo "Development") mitiga este riesgo por completo para el alcance del proyecto.
- **Límites de Velocidad (Rate Limits):** Meta impone límites de velocidad basados en el uso. Para el bajo volumen de una aplicación de demostración, esto no representa un riesgo significativo.

2.2. LinkedIn API (Marketing Developer Platform)

La integración con LinkedIn permite la publicación de contenido profesional en el feed de un usuario o de una página de empresa.

2.2.1. Documentación Oficial y Proceso de Aprobación

La funcionalidad de publicación reside en la **LinkedIn Marketing Developer Platform**, específicamente la **Share API**. La documentación está alojada en la plataforma de Microsoft Docs.

- **Proceso de Aprobación (Riesgo Medio):** El acceso no es instantáneo. Se debe crear una aplicación en el Portal de Desarrolladores de LinkedIn y solicitar acceso a los "Productos" de la API. Para este proyecto, se requiere la aprobación del producto **"Share on LinkedIn"**. Este proceso de aprobación puede tardar varios días y es un hito crítico temprano en el cronograma.

2.2.2. Sistema de Autenticación: OAuth 2.0 (3-legged)

Al igual que Meta, LinkedIn utiliza el flujo de **autenticación OAuth 2.0 (3-legged)**. El proceso es conceptualmente idéntico: el usuario debe autorizar la aplicación, la cual recibe un *token de acceso* para realizar publicaciones en su nombre.

2.2.3. Permisos Requeridos: w_member_social

El permiso (scope) principal y esencial para este proyecto es `w_member_social`. Este permiso otorga a la aplicación la capacidad de publicar contenido (shares) en nombre del miembro autenticado.

2.2.4. Endpoint Clave: POST /ugcPosts (API de Share v2)

El endpoint central para crear contenido es `POST /ugcPosts` (User Generated Content Posts). Este único endpoint es versátil y se utiliza para crear publicaciones de solo texto, publicaciones con imágenes o publicaciones con enlaces (artículos). La estructura del cuerpo de la solicitud (payload JSON) define el tipo de publicación que se está creando.

2.2.5. Estrategia de Implementación (Enfoque Pragmático)

De forma análoga a la estrategia de Meta y para mitigar el riesgo de implementar un flujo OAuth 2.0 completo bajo un cronograma ajustado:

- **Implementación:** Se utilizará un **token de acceso de desarrollador autogenerado** desde el portal de la aplicación en LinkedIn. El desarrollador (dueño de la app) puede generar un token para su propia cuenta sin necesidad de implementar la interfaz de usuario de autenticación. Esto es suficiente para probar y demostrar la funcionalidad de publicación en el perfil del desarrollador.

2.3. TikTok API (Content Posting API)

La integración con TikTok representa, con diferencia, el mayor desafío técnico del proyecto, requiriendo un análisis de viabilidad separado y una estrategia de mitigación específica.

2.3.1. Análisis de Viabilidad: Alto Nivel de Restricción

Tras una investigación de la documentación de `TikTok for Developers`, la viabilidad de una integración directa y automatizada se considera **extremadamente baja a nula** para el alcance de este proyecto. A diferencia de las plataformas basadas en texto o imágenes, la `Content Posting API` de TikTok está diseñada principalmente para la ingesta de video y está fuertemente restringida.

2.3.2. Proceso de Verificación: Inviable para proyectos académicos/individuales a corto plazo

El acceso a la `Content Posting API` no es público ni se otorga mediante un simple registro de aplicación. Está reservado para un ecosistema cerrado de *partners* de marketing verificados y grandes empresas. El proceso de solicitud es opaco, largo y no está diseñado para desarrolladores individuales o proyectos académicos. Intentar esta integración sería un cuello de botella que pondría en riesgo la totalidad del cronograma del proyecto.

2.3.3. Plan B (Recomendado y Seleccionado): Flujo de Trabajo Semiautomático

Dada la inviabilidad de la integración directa, se ha seleccionado un **"Plan B" pragmático** que consiste en un flujo de trabajo semiautomático.

- **Justificación:** Esta estrategia **evita por completo la barrera principal: la API de video y su proceso de verificación**. Se reenfoca el valor del sistema, pasando de la "publicación automática" a la "asistencia inteligente de contenido". El sistema seguirá realizando el 90% del trabajo cognitivo, que es la adaptación del contenido, eliminando el 100% del bloqueo técnico.
- **Implementación:** El backend **no** intentará establecer ninguna conexión con la API de TikTok. En su lugar, el módulo LLM (Gemma) será instruido con un prompt específico para generar los componentes de texto optimizados para un video de TikTok. Los entregables del sistema para esta red serán:
    1. El **Caption (descripción)** perfectamente adaptado (corto, enérgico, con emojis).
    2. Una lista de **Hashtags de Tendencia** relevantes.
    3. Un **Video Hook (gancho)**: la primera frase clave del guion del video, diseñada para capturar la atención en los primeros 5 segundos.
- **Flujo de Usuario:** El portal web presentará estos tres componentes de texto al usuario. El flujo será el siguiente:
    1. El usuario graba y edita su video en su dispositivo móvil.
    2. El usuario abre la aplicación de TikTok para subir el video.
    3. En el portal de nuestra plataforma (en su móvil o PC), el usuario copia el *caption* y los *hashtags* generados.
    4. El usuario pega este contenido directamente en el campo de descripción de la app de TikTok antes de publicar.

2.4. WhatsApp Business API

Para la integración con WhatsApp, la plataforma de mensajería instantánea más grande del mundo, se analizaron dos enfoques principales.

2.4.1. Análisis Comparativo: Plataforma de Meta (Directa) vs. Proveedores PaaS (Twilio)

- **Meta (Directa):** Esta es la API oficial y directa de WhatsApp. Si bien es poderosa y más económica a gran escala, su complejidad de configuración es prohibitiva para un proyecto a corto plazo. Requiere:
    1. Configuración de un **Facebook Business Manager**.
    2. Creación y verificación de un **WABA (WhatsApp Business Account)**.
    3. Proceso de **verificación de un número de teléfono** (que no puede ser el número personal actual) y aprobación de plantillas de mensajes.
    4. El auto-alojamiento y gestión de una **infraestructura de webhooks** para recibir respuestas y estados de entrega.
- **Twilio API for WhatsApp (PaaS):** Twilio es un Proveedor de Plataforma como Servicio (PaaS) que actúa como intermediario. **Abstrae por completo** toda la complejidad de la configuración de Meta. Twilio gestiona el WABA, los webhooks y la relación con Meta, ofreciendo a cambio una API REST limpia, simple y extremadamente bien documentada para enviar y recibir mensajes.

2.4.2. Selección: Twilio API for WhatsApp

Se ha seleccionado **Twilio API for WhatsApp** como la solución de implementación. La

decisión se basa puramente en la viabilidad y la velocidad de desarrollo. El ligero costo adicional por mensaje que impone Twilio (en un escenario de producción) es un precio insignificante a pagar a cambio de reducir la complejidad de configuración de semanas a minutos.

2.4.3. Estrategia de Implementación (Enfoque Pragmático)

Para maximizar la eficiencia y eliminar cualquier costo durante el desarrollo y la demostración, se utilizará el **Twilio Sandbox (Gratuito)**.

- **Uso del Sandbox:** El Sandbox de Twilio proporciona un número de teléfono de WhatsApp compartido y un entorno de prueba inmediato. Los desarrolladores pueden comenzar a enviar y recibir mensajes a sus propios números (previamente verificados en el sandbox) en menos de 5 minutos, sin necesidad de adquirir o verificar un número de teléfono dedicado.
- **Eliminación de Complejidad:** Esta estrategia **elimina la necesidad de verificar un número de teléfono dedicado y de gestionar una infraestructura de webhooks compleja**, ya que el sandbox se gestiona desde el panel de control de Twilio. Esto es ideal para el desarrollo, las pruebas y la demostración final.

2.5. Resumen de Viabilidad de APIs

El análisis de las cinco plataformas revela un panorama mixto de complejidad, pero con caminos viables y pragmáticos para todas. La estrategia general es **evitar las complejidades de autenticación de usuario final (OAuth 2.0) y las verificaciones de API restrictivas (TikTok)**, en favor de un sistema funcional para la demostración.

Plataforma	Complejidad de API	Riesgo de Aprobación	Estrategia de Implementación (Demo)
Facebook	Media	Alto (para usuarios)	Token de Desarrollador (Riesgo Cero)
Instagram	Alta (Flujo 2 pasos)	Alto (para usuarios)	Token de Desarrollador (Riesgo Cero)
LinkedIn	Media	Medio (aprobación app)	Token de Desarrollador (Riesgo Cero)
TikTok	Extrema	Inviable	Plan B (Semiautomático) (Riesgo Cero)
WhatsApp	Extrema (Directa)	Alto (verificación)	Twilio Sandbox (PaaS) (Riesgo Cero)

2.5.2. Estrategia de Implementación por Fases (Demo vs. Producción)

Este proyecto se enfoca exclusivamente en la **Fase de Demo**:

- **Fase 1 (Demo - Alcance del Proyecto):** El objetivo es demostrar la funcionalidad *end-to-end* (desde la ingesta de contenido hasta la publicación) utilizando las estrategias de mitigación definidas: Tokens de Desarrollador autogenerados (para Meta/LinkedIn), el Twilio Sandbox (para WhatsApp) y el flujo semiautomático (para TikTok). El sistema será 100% funcional para la cuenta del desarrollador.
- **Fase 2 (Producción - Fuera del Alcance):** Una versión de producción futura requeriría implementar los flujos OAuth 2.0 completos para que los usuarios conecten sus propias cuentas, pasar por los procesos de "App Review" de Meta y LinkedIn, y migrar de Twilio Sandbox a un número de WhatsApp verificado.

3. Análisis de Características de Redes Sociales

3.1. Propósito del Análisis (Input para el LLM)

El éxito de este proyecto no reside en la mera conexión de APIs, sino en la **calidad de la adaptación del contenido**. El componente LLM (Gemma) es una herramienta poderosa, pero requiere instrucciones precisas.

El propósito de este análisis es definir los **requisitos, restricciones y "personalidades"** de cada red social. Esta información no es un mero ejercicio comparativo; es la **materia prima esencial para la ingeniería de prompts** (las instrucciones dadas al LLM). El LLM será instruido para actuar como cinco "expertos en marketing de redes sociales" diferentes, y este análisis define las reglas de actuación para cada uno.

3.2. Tabla Comparativa de Requisitos de Contenido (Entregable)

(Se incluirá la tabla comparativa completa provista en el documento original del proyecto, que detalla los límites de caracteres, tono, uso de hashtags, emojis y formatos especiales para Facebook, Instagram, LinkedIn, TikTok y WhatsApp).

3.3. Implicaciones Clave para el Diseño de Prompts

El análisis de la tabla anterior genera las siguientes directrices directas para la ingeniería de prompts del LLM:

- **Facebook (Flexibilidad):** El prompt permitirá al LLM generar contenido más largo y estructurado (varios párrafos). Se le pedirá que mantenga un tono que puede ser tanto casual como informativo (semi-formal) y que integre *links* y *llamadas a la acción* de forma natural.
- **Instagram (Brevedad y Visual):** El prompt será estricto. Forzará al LLM a ser conciso (idealmente menos de 300 caracteres para priorizar la legibilidad). Se le exigirá un tono casual, un uso intensivo de emojis relevantes, y que la salida incluya una lista separada de 5-10 *hashtags* relevantes agrupados al final del texto.

- **LinkedIn (Profesionalismo y Estructura):** El prompt instruirá al LLM para que adopte un tono estrictamente profesional y corporativo. Se prohibirá el lenguaje coloquial y el uso excesivo de emojis (permitiendo solo emojis profesionales). Se le pedirá que estructure la información claramente, usando párrafos cortos o listas con viñetas para facilitar la lectura.
- **TikTok (Ganchos y Tendencias):** Este será el prompt más creativo. El objetivo principal del LLM será generar el **"gancho"** (el *video hook*). Se le pedirá que genere una primera frase extremadamente corta y llamativa. Además, se le solicitará un *caption* breve y enérgico (tono juvenil) y una lista de *hashtags* que no solo sean relevantes, sino que estén *en tendencia*.
- **WhatsApp (Tono Conversacional Directo):** El prompt instruirá al LLM para que abandone el tono de "publicación" y adopte un tono de "mensaje directo" (1 a 1). El texto debe ser conversacional, empezar con un saludo (ej. "¡Hola!", "¡Qué tal! ") y ser directo, conciso y fácil de leer, utilizando emojis para añadir un toque humano y amigable.

4. Selección y Justificación del Modelo de Lenguaje (LLM)

El componente LLM (Large Language Model) es el "cerebro" de este proyecto. Es la tecnología responsable de realizar la tarea cognitiva central: la adaptación inteligente del contenido. La elección de este componente es, por lo tanto, crítica y tiene implicaciones directas en la arquitectura del sistema, los costos operativos, la velocidad de respuesta y la viabilidad general del proyecto.

4.1. Requisitos del Sistema LLM

El LLM seleccionado no solo debe ser un generador de texto competente, sino que debe cumplir con dos requisitos funcionales estrictos:

1. **Capacidad de Adaptación de Tono y Formato:** El modelo debe ser capaz de ingerir un texto base y re-escribirlo para cinco contextos radicalmente diferentes (las "personalidades" de cada red social definidas en la Sección 3). Esto incluye ajustar el tono (ej. profesional vs. juvenil), la longitud (límites de caracteres), la estructura (ej. párrafos vs. ganchos) y los elementos auxiliares (ej. hashtags, emojis).
2. **Generación de Salida Estructurada (Formato JSON):** Este es un requisito técnico no negociable. Para que el *backend* (FastAPI) pueda consumir la respuesta del LLM de manera fiable, el modelo debe devolver su salida en un formato JSON predecible y estrictamente formateado. Una respuesta de texto simple no es aceptable, ya que requeriría un *parsing* (análisis) frágil y propenso a errores. El modelo debe ser capaz de seguir instrucciones de formato de salida complejas.

4.2. Modelo Seleccionado: Gemma (ejecutado vía Ollama)

Tras un análisis comparativo de las opciones disponibles, el modelo de lenguaje seleccionado para este proyecto es **Gemma**, en su versión optimizada para instrucciones.

Este modelo no será consumido a través de una API de terceros, sino que será **ejecutado localmente** utilizando la plataforma **Ollama**. Ollama es un servidor de modelos de código abierto que permite empaquetar, distribuir y ejecutar LLMs (como Gemma, Llama, Mistral) en el entorno de desarrollo local, exponiendo una API REST simple para su consumo.

4.3. Justificación de la Elección (Enfoque Pragmático)

Esta decisión se basa en una estrategia de **viabilidad y pragmatismo**, diseñada para maximizar las probabilidades de éxito dentro del cronograma de 6 clases. Las ventajas de este enfoque son determinantes:

- **Costo Cero:** Al ser un modelo de código abierto ejecutado localmente, no hay ningún costo asociado por llamada de API. Esto es crucial en un proyecto académico que requiere **iteración ilimitada**. La "ingeniería de prompts" (el proceso de refinar las instrucciones dadas al LLM) es la tarea más crítica, y puede requerir cientos de pruebas. Hacer esto en una API de pago sería prohibitivo.
- **Control Local y Sin Dependencia (Confiabilidad):** El sistema completo (Frontend, Backend y LLM) se ejecuta en la máquina local. Esto elimina cualquier dependencia de servicios de terceros o de la conectividad a Internet. La demostración final no fallará porque una API externa esté caída, tenga latencia o la conexión Wi-Fi sea inestable. Esto reduce drásticamente los riesgos del proyecto.
- **Privacidad:** Aunque el contenido de ejemplo sean noticias, este enfoque garantiza que ningún dato (ni el contenido de entrada ni los prompts) abandone el entorno local. Este es un estándar de buenas prácticas profesionales.
- **Baja Latencia:** Las llamadas a APIs externas de LLM están sujetas a la latencia de red. Una llamada local desde el *backend* de FastAPI al servidor de Ollama (ambos en `localhost`) es casi instantánea. Esto se traduce en una experiencia de usuario (UX) superior, donde las adaptaciones se generan en 1-2 segundos en lugar de 5-10.
- **Facilidad de Integración:** Ollama expone una API REST (`/api/generate`) simple y, fundamentalmente, **compatible con el formato de la API de OpenAI**. Para el *backend* de FastAPI, hacer una llamada a `http://localhost:11434/api/generate` es tan simple como cualquier otra solicitud HTTP. Esta compatibilidad también facilita el uso de librerías existentes.
- **Calidad Suficiente:** Si bien modelos comerciales más grandes pueden ser más potentes en tareas de razonamiento complejo, para la tarea específica de este proyecto (reformulación, adaptación de tono y generación de JSON), el modelo Gemma es **altamente capaz y completamente suficiente** para producir resultados de alta calidad.

4.4. Análisis de Alternativas (OpenAI, Claude)

Se consideraron las principales APIs comerciales, como las de **OpenAI (modelos GPT-4o-mini, GPT-3.5-Turbo)** y **Anthropic (modelos Claude 3.x)**.

Sin embargo, se descartaron por las siguientes razones, que entran en conflicto directo con nuestro enfoque pragmático:

1. **Costos Operativos:** Son servicios de pago por uso (pay-per-token). Como se mencionó, esto penaliza la fase de experimentación e "ingeniería de prompts", que es la más importante.
2. **Dependencia de API Keys:** Introducen complejidad en la gestión de credenciales. Las claves de API deben ser generadas, financiadas, almacenadas de forma segura y gestionadas como variables de entorno, añadiendo un punto de fricción al desarrollo.
3. **Latencia de Red:** El rendimiento de la aplicación dependería de factores externos, inaceptables para una demostración controlada.
4. **Barrera de Entrada:** Requieren la creación de cuentas, configuración de facturación y gestión de límites de uso.

En conclusión, si bien estos modelos son excelentes, introducen riesgos y complejidades (costo, latencia, dependencia) que son innecesarios para los objetivos de este proyecto. La pila Ollama/Gemma proporciona un balance óptimo de rendimiento, control y viabilidad.

4.5. Tabla Comparativa de Modelos (Entregable)

La siguiente tabla resume el análisis de las alternativas y justifica la selección final:

Modelo	**Costo**	**Latencia**	**Calidad (para esta tarea)**	**Facilidad de Setup**	**Dependencia Externa**
Gemma (vía Ollama)	Cero	Muy Baja (Local)	Alta / Suficiente	Media (Instalación local)	Ninguna
GPT-4o-mini (API)	Bajo ($$)	Baja-Media (Red)	Muy Alta	Alta (Solo API Key)	Alta (API OpenAI)
Claude 3 Sonnet (API)	Medio ($$$)	Baja-Media	Muy Alta	Alta (Solo API Key)	Alta (API Anthropic)
GPT-3.5-Turbo (API)	Muy Bajo ($)	Baja-Media (Red)	Media (Peor en JSON)	Alta (Solo API Key)	Alta (API OpenAI)
Selección Final:	Gemma (vía Ollama)	Muy Baja (Local)	Mejor balance de costo, control, latencia y viabilidad para un proyecto académico de 6 clases.		

5. Propuesta de Arquitectura y Stack Tecnológico

Basado en el análisis de viabilidad de las APIs (Sección 2) y la selección del modelo LLM (Sección 4), se propone la siguiente arquitectura de sistema y stack tecnológico. Esta propuesta está optimizada para la consecución exitosa del proyecto dentro del cronograma de 6 clases.

5.1. Principios de Diseño

La arquitectura se fundamenta en tres principios de diseño clave, priorizando la entrega funcional sobre la complejidad de producción:

1. **Simplicidad (Máxima):** Se priorizará la facilidad de implementación, configuración y mantenimiento por encima de la "perfección" de ingeniería. Esto implica una reducción deliberada de la sobrecarga de infraestructura, como la eliminación de servidores de bases de datos independientes o sistemas de colas complejos.
2. **Viabilidad (En 6 Clases):** Este es el principio rector. Cada decisión tecnológica se ha tomado con el objetivo de asegurar que un prototipo funcional *end-to-end* sea completable y demostrable en la fecha de presentación final. Esto requiere "cortar" estratégicamente las características de mayor complejidad y menor valor demostrativo (ej. flujos OAuth 2.0 completos).
3. **Modularidad:** El sistema se diseñará con una clara separación de responsabilidades. El *Frontend* (React), el *Backend* (FastAPI) y el *Servicio LLM* (Ollama) operarán como componentes desacoplados que se comunican a través de APIs REST, permitiendo el desarrollo y la depuración de forma independiente.

5.2. Stack Tecnológico Seleccionado (Enfoque Pragmático)

Se ha seleccionado un stack tecnológico moderno, minimalista y de alta productividad para facilitar un desarrollo rápido.

- **Backend: FastAPI (Python)**
    - **Justificación:** Se selecciona FastAPI por ser un framework de Python minimalista, asíncrono y de alto rendimiento. Su característica de generación automática de documentación de API (`/docs`) acelera drásticamente el desarrollo y las pruebas. La elección de Python como lenguaje de backend es estratégica, ya que es el lenguaje nativo del ecosistema de IA/ML, simplificando la interacción con el servidor local de Ollama (incluso si es vía REST) y permitiendo el uso de cualquier librería de procesamiento de texto auxiliar que pudiera ser necesaria.
- **Frontend: React (con Vite + TypeScript)**
    - **Justificación:** React es el estándar de la industria para construir interfaces de usuario interactivas. Se utilizará **Vite** como herramienta de empaquetado, ya que proporciona un servidor de desarrollo ultrarrápido con *Hot Module Replacement* (HMR), crucial para una iteración ágil del UI. El uso de **TypeScript** añadirá robustez al código, asegurando que los tipos de datos que fluyen entre el frontend y el backend sean consistentes.
- **Base de Datos: SQLite**

    - **Justificación:** Esta es una decisión pragmática clave para maximizar la viabilidad. En lugar de utilizar un servidor de base de datos complejo como PostgreSQL (que requeriría instalación, configuración, gestión de usuarios y ejecución como un servicio separado o en Docker), se optará por **SQLite**. SQLite es una base de datos *serverless* que almacena la base de datos completa en un **único archivo** (ej. `project.db`) dentro del directorio del backend. Requiere **cero configuración** y es perfectamente capaz de manejar la carga de un solo usuario para una demostración. El schema SQL propuesto es 100% compatible.
- **Orquestación (Colas): Procesamiento Síncrono (Para la Demo)**
    - **Justificación:** Esta es la segunda decisión pragmática clave. Un sistema de producción real *requeriría* un sistema de colas asíncronas (como **Redis** + **Celery/BullMQ**) para manejar tareas de larga duración como la publicación en 5 APIs distintas. Sin embargo, la implementación de un sistema de colas (con *broker*, *workers* y gestión de estado de tareas) añade una capa masiva de complejidad de infraestructura.
    - **Para la demo:** Se optará por un flujo de **procesamiento síncrono**. Cuando el usuario presione "Publicar", el *endpoint* de FastAPI (P0ST `/api/posts/:id/publish`) ejecutará las 5 llamadas de API en un bucle, una tras otra, y no devolverá una respuesta hasta que todas hayan terminado. El usuario experimentará un **tiempo de espera en el frontend** (ej. 10-15 segundos). Este *trade-off* de experiencia de usuario es aceptable para una demo a cambio de una reducción drástica en la complejidad del sistema. Esta mejora se documentará como el principal punto de "trabajo futuro".

5.3. Diagrama de Arquitectura (Simplificado)

(Se incluirá en el entregable final un diagrama de arquitectura basado en el propuesto en el documento del proyecto, pero modificado para reflejar las decisiones pragmáticas. Específicamente, el diagrama eliminará los componentes "Redis Queue" y "Workers", mostrando una conexión directa y síncrona desde el "Backend Central (FastAPI)" hacia las APIs externas. Además, el componente "PostgreSQL" será reemplazado por un "SQLite DB" representado como un archivo local dentro del propio servicio de Backend).

5.4. Diseño de Base de Datos (Schema Básico con SQLite)

El diseño de la base de datos se mantendrá idéntico al schema SQL propuesto en el documento del proyecto, ya que es fundamental para la lógica de la aplicación y es totalmente compatible con el motor de SQLite.

(Se incluirá el schema SQL de las tablas `posts` y `publications` provisto en el documento original del proyecto).

5.5. Flujo de Datos (Simplificado y Síncrono)

El flujo de trabajo completo del usuario, desde la creación hasta la publicación, seguirá estos pasos basados en la arquitectura síncrona:

1. **Creación:** El usuario envía el título y el contenido desde el frontend (React) al backend (FastAPI) vía `POST /api/posts`.
2. **Almacenamiento (Draft):** FastAPI guarda el nuevo post en la base de datos SQLite con el estado `'draft'`.
3. **Adaptación (Llamada al LLM):** El usuario presiona "Adaptar" en el frontend, lo que dispara una llamada `POST /api/posts/:id/adapt`.
4. **Generación de Contenido:** El backend de FastAPI construye el *prompt* y realiza una llamada HTTP local (`POST /api/generate`) al servidor de Ollama que está ejecutando el modelo Gemma.
5. **Recepción de JSON:** Ollama devuelve el contenido adaptado para las 5 redes en un formato JSON estructurado.
6. **Almacenamiento (Adaptaciones):** FastAPI parsea el JSON y guarda las 5 adaptaciones como nuevas filas en la tabla `publications` de SQLite.
7. **Publicación (Llamada Síncrona):** El usuario presiona "Publicar", disparando una llamada `POST /api/posts/:id/publish`. El frontend muestra un indicador de "cargando".
8. **Bucle Síncrono:** El backend de FastAPI itera en un bucle síncrono sobre las publicaciones pendientes:
    - Llama a la API de Meta (Facebook) usando el Token de Desarrollador.
    - Llama a la API de Meta (Instagram) (flujo de 2 pasos).
    - Llama a la API de LinkedIn usando el Token de Desarrollador.
    - Llama a la API de Twilio (WhatsApp) usando el Sandbox.
    - Actualiza el estado de la publicación de TikTok a `'manual_pending'`.
9. **Respuesta:** Una vez que el bucle termina (tras 10-15 segundos), FastAPI devuelve una respuesta `200 (OK)` al frontend.
10. **Actualización de UI:** El frontend quita el indicador de "cargando" y refresca el estado, mostrando los resultados (Éxito, Fallido) de cada publicación.

6. Conclusión del Estudio

6.1. Resumen de Viabilidad (Alta)

El estudio de viabilidad concluye que el desarrollo del proyecto "Adaptación Automática de Contenido para Redes Sociales" es **altamente viable** dentro del cronograma de 6 clases, **siempre y cuando se adhiera estrictamente a la arquitectura y las decisiones pragmáticas** definidas en este documento.

Las estrategias de mitigación (Tokens de Desarrollador, Twilio Sandbox, Plan B para TikTok, SQLite y procesamiento síncrono) eliminan eficazmente los principales bloqueadores técnicos y focos de complejidad que podrían poner en riesgo el proyecto. La funcionalidad central (adaptación por LLM y publicación en 4 de 5 redes) es completamente alcanzable.

6.2. Riesgos Identificados

A pesar de la alta viabilidad, persisten los siguientes riesgos gestionables:

1. **Aprobación de la App de LinkedIn (Riesgo Medio):** El único proceso de aprobación externo que se debe realizar es la solicitud del producto "Share on LinkedIn". Se debe iniciar este proceso de inmediato, ya que puede tardar varios días.
2. **Gestión de Tokens (Riesgo Bajo):** Los tokens de desarrollador de Meta y LinkedIn tienen caducidad (corta o larga). Se debe tener un proceso para refrescarlos manualmente antes de la demostración final.
3. **Timeout Síncrono (Riesgo Medio):** El flujo de publicación síncrono (Sección 5.5, Paso 8) podría ser lento. Si la suma de las llamadas a las APIs supera los 30 segundos, podría provocar un *timeout* en el navegador o en el servidor. Esto se mitigará asegurando que la lógica de publicación sea eficiente.
4. **Configuración de Ollama (Riesgo Bajo):** La instalación de Ollama y la descarga del modelo Gemma es un prerrequisito que puede suponer un pequeño obstáculo de configuración inicial.

6.3. Siguientes Pasos

Con la estrategia de viabilidad y arquitectura definidas, el siguiente paso inmediato es comenzar la implementación del primer entregable funcional, como se describe en el cronograma:

- **Acción Inmediata:** Iniciar la solicitud de aprobación de la App de LinkedIn.
- **Siguiente Hito (Clase 2):** "Primer Prototipo - LLM y Adaptación de Contenido". El trabajo comenzará en el *backend* (FastAPI) para construir el *endpoint* que recibe un contenido, se comunica con el servidor local de Ollama (Gemma) y devuelve con éxito el JSON estructurado con las 5 adaptaciones de contenido.