Tema: Investigación y Feasibility Study

1. Investigación de APIs

Meta (Facebook)

La integración se realiza a través de la **API Graph de Meta**. Requiere una App de Desarrollador de Meta y una **Página de Facebook** específica a la que se desea publicar. La autenticación es estándar, vía **OAuth 2.0**, y exige una gestión cuidadosa de los tokens de acceso con permisos como pages_manage_posts.

Ejemplo de código

```
// Se usa 'axios' para la petición POST
// Esto es un ejemplo conceptual de publicación de texto
import axios from 'axios';

const PAGE_ID = 'tu_page_id';
const ACCESS_TOKEN = 'tu_access_token';
const message = 'Contenido adaptado para Facebook...';

const url = `https://graph.facebook.com/v19.0/${PAGE_ID}/feed`;

try {
  const response = await axios.post(url, {
    message: message,
    access_token: ACCESS_TOKEN
  });
  console.log('Publicado en Facebook:', response.data);
} catch (error) {
  console.error('Error publicando en Facebook:', error.response.data);
}
```

Meta (Instagram)

La integración también se realiza a través de la **API Graph de Meta**. Requiere una App de Desarrollador de Meta y una **Cuenta de Instagram Empresarial** (Business Account) que debe estar vinculada a una Página de Facebook. La autenticación es estándar, vía **OAuth 2.0**, y requiere permisos específicos como instagram_content_publish.

Ejemplo de código

```
import axios from 'axios';

const INSTAGRAM_BUSINESS_ID = 'tu_id_de_negocio_ig';
const ACCESS_TOKEN = 'tu_access_token';
const imageUrl = 'https://url.de.tu.imagen.jpg'; // Imagen debe ser pública
const caption = 'Contenido visual adaptado para Instagram #Tech';

const step1_url =
`https://graph.facebook.com/v19.0/${INSTAGRAM_BUSINESS_ID}/media`;

try {
  // --- PASO 1: Crear el contenedor ---
  const createContainerResponse = await axios.post(step1_url, {
    image_url: imageUrl,
    caption: caption,
    access_token: ACCESS_TOKEN
  });

  const creationId = createContainerResponse.data.id;
  if (!creationId) throw new Error('No se pudo obtener el ID de creación');

  console.log('Contenedor creado:', creationId);

  // --- PASO 2: Publicar el contenedor ---
  const step2_url =
`https://graph.facebook.com/v19.0/${INSTAGRAM_BUSINESS_ID}/media_publish`;

  const publishResponse = await axios.post(step2_url, {
    creation_id: creationId,
    access_token: ACCESS_TOKEN
  });

  console.log('Publicado en Instagram:', publishResponse.data);

} catch (error) {
  console.error('Error publicando en Instagram:',
error.response?.data?.error || error.message);
}
```

LinkedIn

La integración utiliza la **Share API** (API de compartición). Requiere una App de Desarrollador de LinkedIn con el producto "Share on LinkedIn" activado. La

autenticación se basa en **OAuth 2.0** (flujo de 3 pasos) para obtener el token de acceso del usuario.

Ejemplo de código

```
// Publicación de un post de texto en LinkedIn
import axios from 'axios';

const ACCESS_TOKEN = 'tu_access_token_linkedin';
// El 'author' URN se obtiene después de la autenticación
const AUTHOR_URN = 'urn:li:person:XXXXXX';

const url = 'https://api.linkedin.com/v2/ugcPosts';

const body = {
  author: AUTHOR_URN,
  lifecycleState: 'PUBLISHED',
  specificContent: {
    'com.linkedin.ugc.ShareContent': {
      shareCommentingAllowed: true,
      shareMediaCategory: IMAGE, // Cambiar a 'ARTICLE' o 'IMAGE' si aplica
      shareContent: {
        text: 'Nos complace anunciar el lanzamiento de nuestra nueva funcionalidad de reportes...'
      }
    }
  },
  visibility: {
    'com.linkedin.ugc.MemberNetworkVisibility': 'PUBLIC'
  }
};

try {
  const response = await axios.post(url, body, {
    headers: {
      'Authorization': `Bearer ${ACCESS_TOKEN}`,
      'Content-Type': 'application/json',
      'X-Restli-Protocol-Version': '2.0.0' // Header requerido por LinkedIn
    }
  });
  console.log('Publicado en LinkedIn:', response.data);
} catch (error) {
  console.error('Error publicando en LinkedIn:', error.response.data);
```

```
}
```

WhatsApp Business API

La integración tiene dos vías principales: la API de Meta (Cloud API) o un proveedor tercero (BSP) como **Twilio**. Se recomienda **Twilio** por su facilidad de configuración y su *sandbox* gratuito para desarrollo. La API está diseñada para mensajería conversacional.

Ejemplo de código

```
// Envío de un mensaje de WhatsApp usando Twilio
import twilio from 'twilio';

const ACCOUNT_SID = 'ACxxxxxxxxxxxxxxxxx'; // Tu Account SID de Twilio
const AUTH_TOKEN = 'xxxxxxxxxxxxxxxxxx';  // Tu Auth Token de Twilio

const client = twilio(ACCOUNT_SID, AUTH_TOKEN);

try {
  const message = await client.messages.create({
    body: 'Hola! 👋 Te cuento una novedad importante: acabamos de lanzar reportes avanzados en la plataforma.',
    from: 'whatsapp:+14155238886', // Número de Sandbox de Twilio
    to: 'whatsapp:+591XXXXXXX'     // Tu número personal verificado
  });

  console.log('Mensaje enviado por WhatsApp (SID):', message.sid);
} catch (error) {
  console.error('Error enviando WhatsApp:', error.message);
}
```

TikTok

La integración se realiza con la **Content Posting API**. Esta es la API más restrictiva; requiere que la App de Desarrollador pase por un proceso de revisión y aprobación estricto. La funcionalidad principal está diseñada para la **publicación de videos**, no solo de texto.

Plan B (requerido por el proyecto): Si la API no es accesible para desarrollo, el sistema debe al menos generar el *caption* (descripción) optimizado para TikTok, que el usuario copiaría en una subida manual.

Ejemplo de código

```
// Ejemplo conceptual de INICIO de subida de video
import axios from 'axios';

const ACCESS_TOKEN = 'tu_access_token_tiktok';

// 1. Obtener URL de subida
const url = 'https://open.tiktokapis.com/v2/post/publish/video/init/';

const body = {
  post_info: {
    title: 'Nueva funcionalidad increíble',
    description: '¿Viste lo nuevo? 🔥 Reportes que te van a volar la cabeza 🤯 #TechTok',
    privacy_level: 'PUBLIC'
  },
  source_info: {
    source: 'FILE_UPLOAD'
  }
};

try {
  const response = await axios.post(url, body, {
    headers: {
      'Authorization': `Bearer ${ACCESS_TOKEN}`,
      'Content-Type': 'application/json'
    }
  });
  // El 'upload_url' recibido en response.data se usaría para
  // subir el archivo de video real (ej. con un PUT)
  console.log('TikTok listo para subida:', response.data);

} catch (error) {
  console.error('Error con TikTok API:', error.response.data);
}
```

2. Características de Redes Sociales (Tabla):

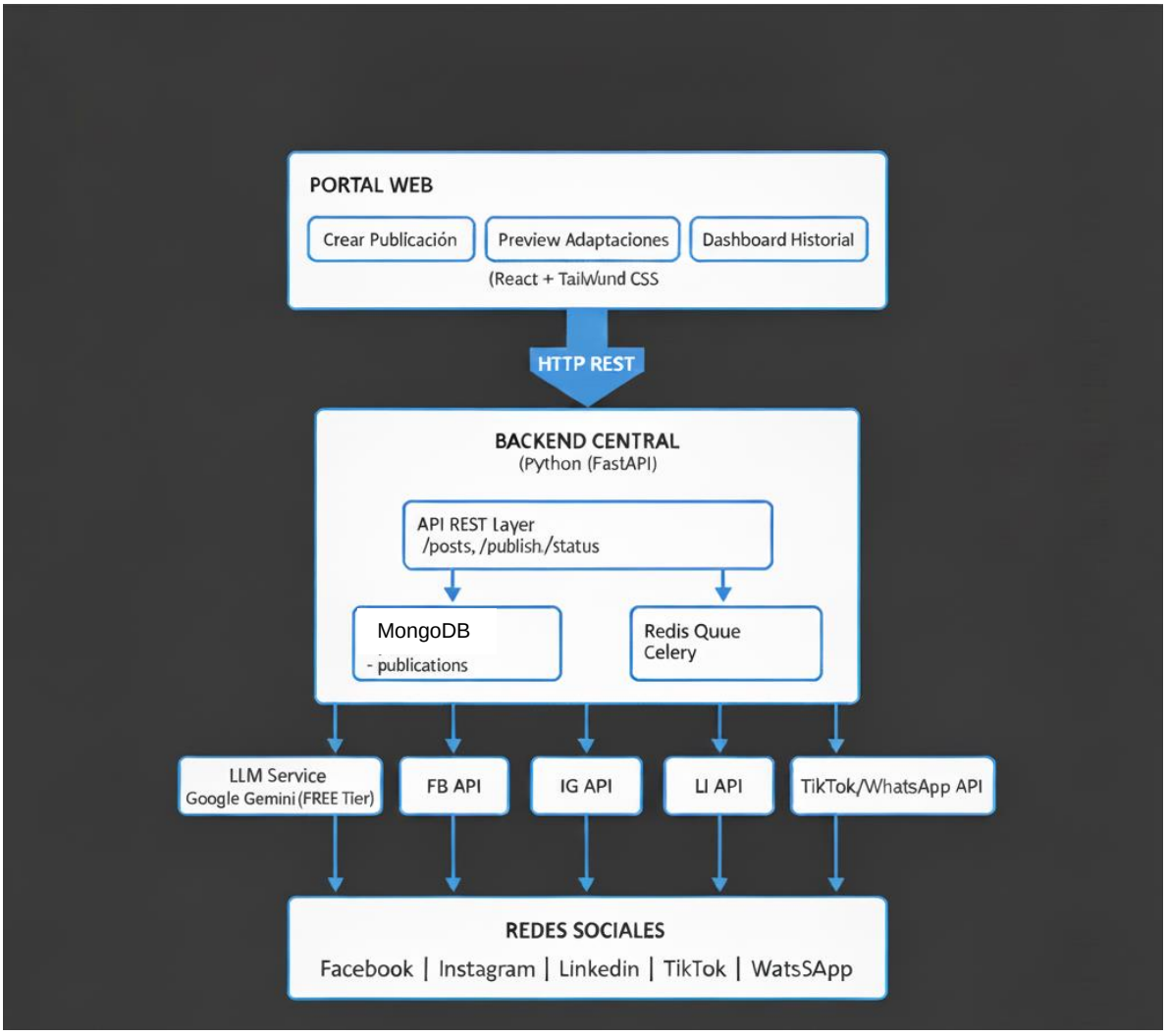
Red Social	Max caracteres	Tono	Hashtags	Emojis	Formato especial	Notas

Facebook	63.206	Casual/Formal	Opcional	Sí	Links, imágenes	Texto largo permitido
Instagram	2.200	Visual/Casual	Importante	Sí	Hashtags al final	Enfoque en imagen
LinkedIn	3.000	Profesional	Moderado	Poco	Artículos	Tono corporativo
TikTok	2.200 (caption)	Joven/Trending	Importante	Sí	Video corto	Requiere video
WhatsApp	65.536	Directo	Raro	Sí	Formato libre	Conversacional

3. Selección de LLM

Modelo	Costo	Latencia	Calidad	Accesibilidad	Recomendado
GPT-4o-mini	$$	Rápido	Alta	API Key	Producción
GPT-3.5	$	Muy rápido	Media-Alta	API Key	Desarrollo
Claude Sonnet	$$	Rápido	Alta	API Key	Producción
Llama 3.1 (Ollama)	Gratis	Medio	Media	Local	Desarrollo/Demo
Gemini	$	Rápido	Alta	API Key	Alternativa

4. Propuesta de Arquitectura y Stack



Es un stack tecnológico elegido:

- **FastAPI (Backend):** Rápido y asíncrono, ideal para manejar múltiples llamadas a APIs (redes sociales) al mismo tiempo.
- **MongoDB (Base de Datos):** Flexible (NoSQL), perfecto para guardar las respuestas JSON de cada API, que tienen estructuras diferentes.
- **Celery (Colas):** Para que las publicaciones se procesen en segundo plano. El usuario no tiene que esperar; Celery maneja los reintentos si algo falla.
- **React (Frontend):** Para construir una interfaz de usuario moderna e interactiva para el portal web.

- **Gemini (LLM):** Proporciona una IA potente y gratuita para la tarea clave de adaptar el contenido, ideal para el desarrollo y la demo.

El flujo de trabajo es el siguiente:

1. **Inicio (React):** El usuario crea el post en el portal web de React.
2. **Adaptación (FastAPI + Gemini):** FastAPI recibe el post, llama a la API de Gemini para adaptar el texto a cada red social (Facebook, TikTok, etc.).
3. **Cola (Celery):** FastAPI guarda todo en **MongoDB** y le pasa las tareas de publicación a **Celery**. El usuario recibe una respuesta inmediata y el trabajo pesado se hace en segundo plano.
4. **Publicación (Workers):** Los *workers* de Celery toman las tareas y llaman a las APIs externas (Meta, LinkedIn, etc.).
5. **Resultado (MongoDB -> React):** Celery actualiza **MongoDB** con el estado (publicado, fallido, o error). El dashboard de React lee esa información para mostrar el estado en tiempo real.