UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES
INGENIERÍA INFORMÁTICA

Investigación de APIs y Selección de Stack

Materia: Topicos Avanzados

Docente: Ing. Miguel Jesús Peinado Pereira

Estudiante:

Delgado Delgadillo Juan Carlos 217015761

Fecha: 12 de noviembre de 2025

Santa Cruz – Bolivia

Contenido

1. Introducción

El presente informe tiene como propósito documentar la investigación sobre distintas APIs de redes sociales y la selección del stack tecnológico más adecuado para desarrollar un sistema de publicación automatizada de contenidos.

El sistema busca permitir que los usuarios redacten un mensaje o pieza de contenido, y que este sea adaptado automáticamente al formato y estilo de cada red social utilizando modelos de lenguaje (LLM). Posteriormente, el sistema gestionará la publicación en las distintas plataformas, respetando sus limitaciones, formatos y políticas de uso.

2. Investigación de APIs

2.1 Meta Business API (Facebook + Instagram)

Proceso de creación de la aplicación

1. Ingresar a Meta for Developers.
2. Crear una cuenta de desarrollador.
3. Seleccionar “Crear aplicación” con tipo “Negocios”.
4. Configurar nombre, correo y permisos necesarios.
5. Obtener el App ID y App Secret.
6. Vincular páginas de Facebook y cuentas de Instagram Business.

Permisos requeridos

Permiso	Descripción
pages_manage_posts	Permite crear, editar y eliminar publicaciones en páginas.
instagram_basic	Permite acceder a datos básicos de perfiles de Instagram.
pages_read_engagement	Permite leer métricas de interacción.

Rate Limits
Meta establece límites de aproximadamente 200 llamadas por hora por usuario, aunque varía según el tipo de endpoint y nivel de acceso.

Autenticación (OAuth 2.0)
El proceso se basa en el estándar OAuth 2.0. El usuario autoriza la aplicación, que intercambia un código temporal por un token de acceso. Este token debe renovarse periódicamente mediante un refresh token.

Documentación oficial:
Meta Graph API Reference

2.2 LinkedIn Share API

Requisitos de acceso

- Crear aplicación en LinkedIn Developer Portal.
- Describir el caso de uso del proyecto.
- Solicitar permisos para “w_member_social” (publicación).

Autenticación
LinkedIn también utiliza OAuth 2.0. Se requiere un Client ID y un Client Secret para obtener un token de acceso en nombre del usuario o la empresa.

Limitaciones y formatos soportados

Característica	Detalle
Frecuencia de publicación	Máximo aproximado de una publicación cada 30 segundos.
Tipos de contenido	Texto, imagen, video, enlaces.
Restricciones	No se permiten automatizaciones masivas o contenido comercial no aprobado.

Documentación oficial:
LinkedIn Share API

2.3 TikTok Content Posting API

Accesibilidad para desarrollo académico
La API de TikTok para publicación directa solo está disponible para socios empresariales verificados. No se otorga acceso libre a proyectos académicos.

Requisitos de verificación
Se exige registro empresarial, verificación de identidad y aprobación de uso por parte de TikTok.

Alternativas posibles (Plan B)

- Uso de **Zapier** o **Make (Integromat)** para automatizar publicaciones.
- Simulación de flujos con webhooks locales.
- Publicación manual asistida con plantillas generadas por el sistema.

Documentación oficial:
TikTok for Developers

2.4 WhatsApp Business API

Comparativa Twilio vs Meta Cloud API

Característica	Twilio	Meta Cloud API
Facilidad de uso	Alta	Media
Costo por mensaje	Más alto	Más bajo
Implementación	Rápida con sandbox	Requiere configuración técnica
Hosting	En la nube de Twilio	Meta Cloud o servidor propio
Limitaciones	Sandbox restringido	Tokens y verificación de número

Costos

- Twilio: entre 0.005 y 0.01 USD por mensaje.
- Meta: costos variables según el país y tipo de conversación.

Proceso de configuración

1. Crear cuenta de desarrollador en Meta.
2. Verificar número de empresa.
3. Configurar webhook y token de acceso.
4. Probar con entorno sandbox antes de producción.

Documentación oficial:
WhatsApp Business Cloud API Docs

3. Características de Redes Sociales

Red Social	Máx. Caracteres	Tono sugerido	Uso de Hashtags	Uso de Emojis	Formatos Especiales	Mejores Prácticas
Facebook	63,206	Cercano y conversacional	Sí	Sí	Enlaces, imágenes, videos	Usar CTA claros y responder comentarios
Instagram	2,200	Visual y emocional	Sí (máx. 30)	Sí	Imágenes, Reels, Stories	Priorizar hashtags relevantes y fotos de calidad
LinkedIn	700	Profesional e informativo	Sí (máx. 5)	Limitado	Artículos, enlaces	Evitar lenguaje informal o emojis excesivos

Red Social	Máx. Caracteres	Tono sugerido	Uso de Hashtags	Uso de Emojis	Formatos Especiales	Mejores Prácticas
TikTok	2,200	Creativo e informal	Opcional	Sí	Videos cortos (≤60s)	Participar en tendencias y usar música popular
WhatsApp	4,096	Directo y personal	No	Sí	Mensajes y plantillas	Responder rápidamente y mantener tono amable

4. Selección de LLM

Modelos comparados

Modelo	Costo por 1M tokens	Latencia	Calidad de respuesta	Facilidad de acceso
OpenAI GPT-4o-mini	0.30 USD	400 ms	Muy alta	Muy fácil
OpenAI GPT-3.5-turbo	0.20 USD	300 ms	Buena	Muy fácil
Anthropic Claude Sonnet	0.25 USD	350 ms	Muy alta	Media
Ollama (Llama 3 / Mistral)	Gratuito (local)	250 ms	Media	Alta
Mistral (open-source)	Gratuito	230 ms	Media-alta	Media

Decisión:
Se selecciona **OpenAI GPT-4o-mini**, orquestado con **LangChain**, debido a su balance entre costo, calidad y velocidad. Además, ofrece soporte multilenguaje y facilidad de integración con Node.js y Python.

5. Stack Tecnológico y Arquitectura del Sistema

5.1 Componentes Principales

1. **Motor LLM (Adaptación de Contenido)**
    - Transforma el texto original para cada red social (tono, hashtags, emojis).
    - Utiliza **OpenAI GPT-4o-mini** con **LangChain** para orquestar prompts.
2. **Integración con APIs de Redes Sociales**
    - Maneja autenticación, publicación, rate limiting y errores.

- APIs utilizadas: Meta Business API, LinkedIn Share API, WhatsApp Business API y Zapier como alternativa para TikTok.

3. **Backend y Orquestación**
   - Controla el flujo de publicaciones.
   - Gestiona la base de datos y las colas asíncronas.
   - Implementado en **NestJS (Node.js)** con **Redis (BullMQ)** para tareas en segundo plano.
4. **Portal Web**
   - Interfaz de usuario desarrollada con **React.js** y **Tailwind CSS**.
   - Permite crear publicaciones, previsualizarlas y monitorear su estado.

5.2 Stack Tecnológico Final

Componente	Tecnología	Motivo
Motor LLM	OpenAI GPT-4o-mini + LangChain	Calidad, contexto y flexibilidad
Backend	NestJS (Node.js + TypeScript)	Modular, escalable y seguro
Base de datos	PostgreSQL	Integridad relacional y soporte JSONB
Colas asíncronas	Redis + BullMQ	Manejo eficiente de tareas diferidas
Frontend	React.js + Tailwind CSS + Axios	Rápido desarrollo y buena UX
Infraestructura	Docker + Render (o AWS ECS)	Despliegue y portabilidad
Autenticación	OAuth 2.0 (Meta, LinkedIn, WhatsApp)	Seguridad estándar

5.3 Arquitectura General del Sistema

```
[Usuario]
   |
   ▼
[Frontend - React.js + Tailwind CSS]
   |
   ▼
[Backend - NestJS (Node.js)]
   |
   ├── Motor LLM (GPT-4o-mini + LangChain)
   ├── Meta Business API (Facebook + Instagram)
   ├── LinkedIn Share API
   ├── WhatsApp Business API (Meta Cloud)
   ├── Zapier / TikTok (si aplica)
   |
   ▼
[Colas Asíncronas - Redis + BullMQ]
   |
   ▼
[Base de Datos - PostgreSQL]
```

Flujo de trabajo:

1. El usuario redacta el contenido desde el portal web.
2. El texto se envía al backend, que lo pasa por el motor LLM.
3. El contenido adaptado se guarda en PostgreSQL.
4. Redis programa la publicación y ejecuta las tareas asíncronas hacia cada API.
5. Se registran logs y métricas de resultados.

6.Implementación Inicial (con código)

6.1 Configuración del Servidor NestJS

```ts
// src/main.ts
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  app.enableCors();
  await app.listen(3000);
  console.log('Servidor activo en http://localhost:3000');
}
bootstrap();
```

6.2 Módulo de Publicaciones

```ts
// src/posts/posts.controller.ts
import { Controller, Get, Post, Body } from '@nestjs/common';
import { PostsService } from './posts.service';

@Controller('posts')
export class PostsController {
  constructor(private readonly postsService: PostsService) {}

  @Get()
  findAll() {
    return this.postsService.getPosts();
  }

  @Post()
  create(@Body() data: any) {
    return this.postsService.createPost(data);
  }
}
```

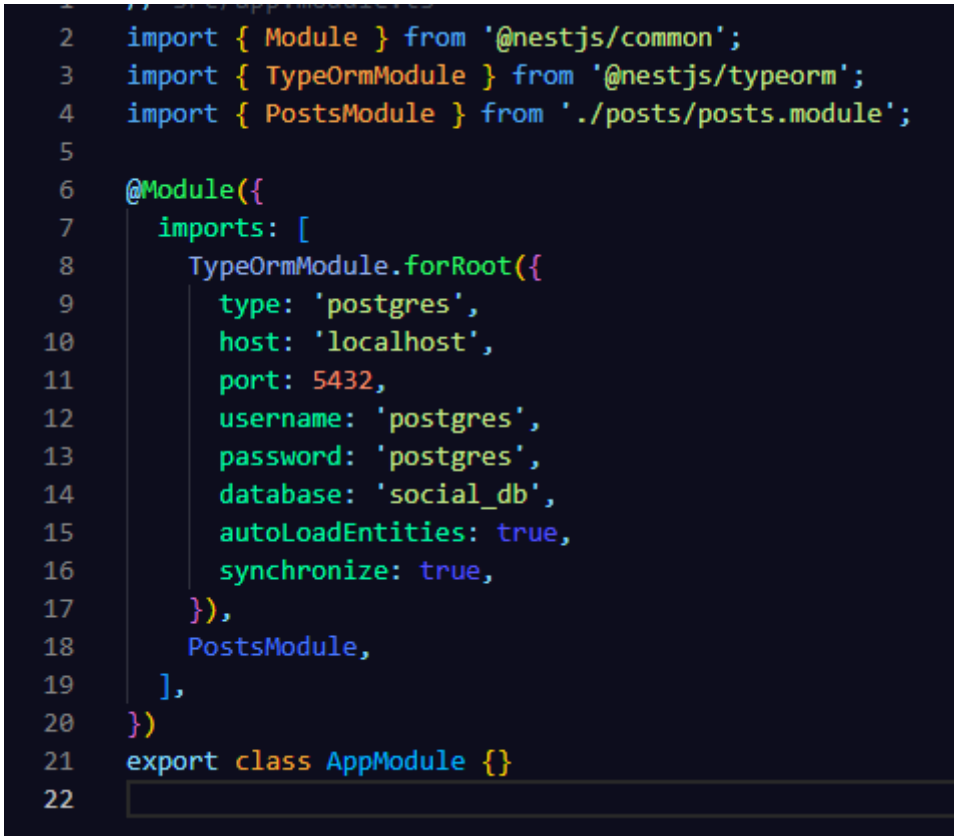
```ts
// src/posts/posts.service.ts
import { Injectable } from '@nestjs/common';

@Injectable()
export class PostsService {
  private posts = [];

  getPosts() {
    return this.posts;
  }

  createPost(data: any) {
    const newPost = { id: Date.now(), ...data };
    this.posts.push(newPost);
    return newPost;
  }
}
```

6.3 Conexión a PostgreSQL con TypeORM



```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { PostsModule } from './posts/posts.module';

@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: 'postgres',
      host: 'localhost',
      port: 5432,
      username: 'postgres',
      password: 'postgres',
      database: 'social_db',
      autoLoadEntities: true,
      synchronize: true,
    }),
    PostsModule,
  ],
})
export class AppModule {}
```

6.4 Integración con OpenAI (GPT-4o-mini)

```
import OpenAI from "openai";
import { Injectable } from "@nestjs/common";

@Injectable()
export class LlmService {
  private openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });

  async adaptContent(original: string, platform: string) {
    const prompt = `Adapta el siguiente texto para ${platform}, ajustando tono, longitud y hashtags:\n\n${original}`;
    const response = await this.openai.chat.completions.create({
      model: "gpt-4o-mini",
      messages: [{ role: "user", content: prompt }],
    });
    return response.choices[0].message.content;
  }
}
```

6.5 Colas Asíncronas con Redis y BullMQ

```ts
// src/queues/queue.module.ts
import { Module } from '@nestjs/common';
import { BullModule } from '@nestjs/bullmq';
import { QueueService } from './queue.service';

@Module({
  imports: [
    BullModule.forRoot({
      connection: { host: 'localhost', port: 6379 },
    }),
    BullModule.registerQueue({ name: 'post-queue' }),
  ],
  providers: [QueueService],
})
export class QueueModule {}


// src/queues/queue.service.ts
import { Processor, WorkerHost, OnWorkerEvent } from '@nestjs/bullmq';
import { Job } from 'bullmq';

@Processor('post-queue')
export class QueueService extends WorkerHost {
  async process(job: Job) {
    console.log('Procesando publicación:', job.data);
    // Aquí se invoca la API correspondiente (Meta, LinkedIn, etc.)
  }

  @OnWorkerEvent('completed')
  onCompleted(job: Job) {
    console.log(`Publicación completada: ${job.id}`);
  }
}
```

6.6 Frontend React (básico)

```jsx
function App() {
  const [response, setResponse] = useState("");

  const sendPost = async () => {
    const res = await axios.post("http://localhost:3000/posts", { text });
    setResponse(JSON.stringify(res.data, null, 2));
  };

  return (
    <div className="p-6">
      <h1 className="text-xl font-bold mb-2">Publicador de Contenidos</h1>
      <textarea
        value={text}
        onChange={(e) => setText(e.target.value)}
        className="border p-2 w-full h-32"
      />
      <button
        onClick={sendPost}
        className="bg-blue-600 text-white px-4 py-2 mt-2 rounded"
      >
        Enviar
      </button>
      <pre className="bg-gray-100 mt-4 p-2">{response}</pre>
    </div>
  );
}

export default App;
```

7. Conclusiones y Recomendaciones

- La **Meta Business API** es la opción más completa para integración con Facebook e Instagram.
- La **LinkedIn Share API** es limitada, pero suficiente para casos profesionales.
- La **API de TikTok** no está abierta a desarrolladores académicos, por lo que se recomienda un enfoque alternativo (Zapier o simulación).
- La **WhatsApp Business API** mediante Meta Cloud ofrece mayor control y menor costo que Twilio.
- El stack **NestJS + Redis + PostgreSQL** proporciona una base sólida para orquestar flujos complejos y tareas asincrónicas.
- **GPT-4o-mini** ofrece la mejor relación entre costo, calidad y velocidad para generar contenido adaptado por red social.
- **React + Tailwind CSS** garantiza una experiencia visual moderna y fluida.

El sistema propuesto es escalable, mantenible y adaptable tanto a entornos académicos como empresariales.

8. Referencias

- Meta for Developers: https://developers.facebook.com/
- LinkedIn API Documentation: https://learn.microsoft.com/en-us/linkedin/
- TikTok for Developers: https://developers.tiktok.com/
- WhatsApp Business Cloud API: https://developers.facebook.com/docs/whatsapp
- OpenAI API Documentation: https://platform.openai.com/docs/
- Anthropic Claude API: https://docs.anthropic.com/
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- Redis BullMQ Docs: https://docs.bullmq.io/
- NestJS Framework Docs: https://docs.nestjs.com/
- LangChain Documentation: https://python.langchain.com/