✅ Configuración de Google Gemini AI Completada

📋 Resumen de la Configuración

La integración de **Google Gemini AI** ha sido configurada exitosamente en la aplicación **Observador 4D**.

🔑 API Key Configurada

- **Variable de entorno**: `GOOGLE_AI_API_KEY`
- **Ubicación**: `/home/ubuntu/observador_4d/nextjs_space/.env`
- **Estado**: ✅ Configurada y verificada
- **API Key**: `AIzaSyBjGmb9ooZx8nVFad73XM25Kfa-qKqhUUY`

🧪 Pruebas Realizadas

Script de Prueba

- **Ubicación**: `/home/ubuntu/observador_4d/nextjs_space/scripts/test-gemini.ts`
- **Resultado**: ✅ Conexión exitosa con Google Gemini AI
- **Modelo utilizado**: `gemini-2.5-flash`
- **Respuesta de prueba**: “Estoy bien, gracias, ¿y tú?”

Comando para ejecutar el script de prueba:

```
cd /home/ubuntu/observador_4d/nextjs_space
npx tsx scripts/test-gemini.ts
```

🚀 Componentes Creados

1. API Route

- **Ruta**: `/api/gemini/chat`
- **Archivo**: `/home/ubuntu/observador_4d/nextjs_space/app/api/gemini/chat/route.ts`
- **Métodos soportados**:
- `POST` : Enviar prompts a Gemini AI
- `GET` : Información sobre el endpoint

Ejemplo de uso del API:

```
// POST request
const response = await fetch('/api/gemini/chat', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    prompt: '¿Qué es la inteligencia artificial?',
    model: 'gemini-2.5-flash' // opcional
  })
});

const data = await response.json();
console.log(data.response);
```

2. Componente de Prueba

- **Componente**: `GeminiTest`
- **Archivo**: `/home/ubuntu/observador_4d/nextjs_space/components/gemini/GeminiTest.tsx`
- **Características**:
- Input para escribir prompts
- Botones con ejemplos de prompts
- Área de respuesta con formato
- Estados de loading y error
- Diseño responsive con Tailwind CSS

3. Página de Prueba

- **Ruta**: `/test-gemini`
- **Archivo**: `/home/ubuntu/observador_4d/nextjs_space/app/test-gemini/page.tsx`
- **URL de acceso**: `http://localhost:3000/test-gemini` (o tu dominio de producción)

🌐 Acceso a la Página de Prueba

Desarrollo Local

```
http://localhost:3000/test-gemini
```

Producción

```
https://tu-dominio.com/test-gemini
```

📚 Modelos Disponibles

Los siguientes modelos de Gemini están disponibles para usar:

1. **gemini-2.5-flash** (recomendado) ⭐
   - Modelo estable y rápido
   - Hasta 1 millón de tokens de contexto
   - Ideal para la mayoría de casos de uso

2. **gemini-2.5-pro-preview-06-05**
   - Versión preview del modelo Pro
   - Mayor capacidad de razonamiento
   - Mejor para tareas complejas

3. **gemini-2.5-flash-preview-05-20**
   - Versión preview de Flash
   - Últimas características experimentales

Cambiar de modelo:

```
// En el API route
const response = await fetch('/api/gemini/chat', {
  method: 'POST',
  body: JSON.stringify({
    prompt: 'Tu pregunta',
    model: 'gemini-2.5-pro-preview-06-05' // Especifica el modelo
  })
});
```

🔧 Estructura de Archivos Creados

```
/home/ubuntu/observador_4d/nextjs_space/
├── .env                                    # ✅ API Key configurada
├── .gitignore                              # ✅ .env protegido
├── scripts/
│   ├── test-gemini.ts                      # ✅ Script de prueba
│   └── list-models.ts                      # ✅ Listar modelos disponibles
├── app/
│   ├── api/
│   │   └── gemini/
│   │       └── chat/
│   │           └── route.ts                # ✅ API endpoint
│   └── test-gemini/
│       └── page.tsx                        # ✅ Página de prueba
├── components/
│   └── gemini/
│       └── GeminiTest.tsx                  # ✅ Componente de prueba
└── CONFIGURACION_GEMINI_COMPLETADA.md      # 📄 Este archivo
```

⚠️ ADVERTENCIAS DE SEGURIDAD IMPORTANTES

🔒 Protección de la API Key

1. **NUNCA compartas tu API Key públicamente**
   - No la incluyas en código que se suba a GitHub
   - No la compartas en foros, chats o redes sociales
   - No la incluyas en el código del cliente (frontend)

2. **El archivo `.env` está protegido**
   - ✅ Ya está incluido en `.gitignore`
   - ✅ No se subirá a Git automáticamente
   - ⚠️ Verifica siempre antes de hacer commit

3. **Verificar `.gitignore`**
   ```bash
   # Asegúrate de que .env está en .gitignore
   cat .gitignore | grep .env
   ```

4. **Variables de entorno en producción**
   - Configura `GOOGLE_AI_API_KEY` en las variables de entorno de tu plataforma de hosting
   - No incluyas el archivo `.env` en el despliegue
   - Usa servicios como Vercel, Netlify, Railway que soportan variables de entorno seguras

🚨 Límites y Cuotas

- **Límite de requests**: Depende de tu plan de Google AI
- **Monitoreo**: Revisa tu uso en Google AI Studio (https://ai.google.dev/usage)
- **Rate limiting**: Implementa límites en tu aplicación para evitar exceder cuotas

🔐 Mejores Prácticas

1. **Rotación de API Keys**
   - Cambia tu API Key periódicamente
   - Si crees que fue comprometida, genera una nueva inmediatamente

2. **Monitoreo de uso**
   - Revisa regularmente el uso de tu API Key
   - Configura alertas en Google Cloud Console

3. **Restricciones de API Key**
   - Considera restringir tu API Key a dominios específicos
   - Configura restricciones en Google Cloud Console

📖 Ejemplos de Uso

Ejemplo 1: Pregunta Simple

```
const response = await fetch('/api/gemini/chat', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    prompt: '¿Qué es la inteligencia artificial?'
  })
});
const data = await response.json();
console.log(data.response);
```

Ejemplo 2: Análisis de Texto

```
const response = await fetch('/api/gemini/chat', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    prompt: 'Analiza el siguiente texto y extrae los puntos clave: [tu texto aquí]'
  })
});
```

Ejemplo 3: Generación Creativa

```
const response = await fetch('/api/gemini/chat', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    prompt: 'Escribe un poema corto sobre el universo en español'
  })
});
```

🐛 Solución de Problemas

Error: “API Key no configurada”

- Verifica que `GOOGLE_AI_API_KEY` está en `.env`
- Reinicia el servidor de desarrollo

Error: “Quota exceeded”

- Has excedido el límite de requests
- Espera el tiempo indicado o actualiza tu plan

Error: “Model not found”

- Verifica que estás usando un modelo válido
- Ejecuta `npx tsx scripts/list-models.ts` para ver modelos disponibles

La página no carga

- Verifica que el servidor está corriendo

- Revisa la consola del navegador para errores
- Verifica que todas las dependencias están instaladas

📞 Recursos Adicionales

- **Documentación de Google Gemini**: https://ai.google.dev/docs
- **Google AI Studio**: https://ai.google.dev/
- **Monitoreo de uso**: https://ai.dev/usage
- **Límites y cuotas**: https://ai.google.dev/gemini-api/docs/rate-limits

---

✅ Checklist de Verificación

- [x] API Key configurada en `.env`
- [x] `.gitignore` incluye `.env`
- [x] Script de prueba ejecutado exitosamente
- [x] API Route creado y funcional
- [x] Componente de prueba creado
- [x] Página de prueba accesible
- [x] Documentación completada
- [x] Advertencias de seguridad incluidas

🎉 ¡Configuración Completada!

La integración de Google Gemini AI está lista para usar en tu aplicación Observador 4D.

Próximos pasos sugeridos:
1. Accede a `/test-gemini` para probar la integración
2. Integra Gemini AI en tus componentes existentes
3. Implementa casos de uso específicos para tu aplicación
4. Configura rate limiting y manejo de errores avanzado
5. Monitorea el uso de tu API Key regularmente

Fecha de configuración: 13 de noviembre de 2025
Versión de @google/genai: 1.29.0
Modelo principal: gemini-2.5-flash