UNIVERSIDAD PRIVADA DOMINGO SAVIO

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

ACTIVIDAD FINAL - FIREBASE: PLATAFORMA DE CHATBOT CON MEMORIA CONTEXTUAL

AUTOR: Daniel Maldonado Cespedes

Cochabamba - Bolivia
2025

CONTENIDO

ANTECEDENTES

1.1 Análisis de la Problemática

En la actualidad, los servicios de atención al cliente requieren herramientas capaces de ofrecer respuestas rápidas, personalizadas y disponibles las 24 horas del día. El problema surge cuando los chatbots tradicionales funcionan de forma aislada: responden preguntas simples, pero no recuerdan el historial de la conversación ni el contexto previo. Esto genera frustración en los usuarios, ya que deben repetir constantemente su información, lo que afecta la calidad de la experiencia y la eficiencia del servicio.

El caso que se nos presenta consiste en diseñar e implementar una **plataforma de chatbot con memoria contextual**, es decir, un sistema que pueda mantener el historial de conversaciones de cada usuario, permitiendo que el diálogo fluya de manera natural y continua. Con este enfoque, si el cliente inicia una conversación y luego regresa más tarde, el chatbot será capaz de retomar la interacción desde donde quedó, sin perder información.

1.1.1 Problemática identificada

- Los chatbots actuales suelen **no guardar el contexto** o lo hacen de forma limitada.
- Los usuarios necesitan repetir datos básicos cada vez que inician sesión.
- El sistema debe garantizar **baja latencia** y **alta escalabilidad** para atender a múltiples usuarios simultáneamente.
- Es imprescindible asegurar la **confidencialidad de los datos**, de forma que cada usuario tenga acceso únicamente a su propio historial.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar una **plataforma de chatbot con memoria contextual utilizando Firebase**, capaz de almacenar en tiempo real los mensajes de los usuarios, el historial de sesiones y las respuestas generadas por la inteligencia artificial, manteniendo un alto nivel de seguridad y eficiencia.

1.2.2 Objetivos específicos

- Modelar una base de datos en Firestore que permita almacenar usuarios, sesiones y mensajes de manera desnormalizada para mejorar el rendimiento.
- Implementar reglas de seguridad estrictas en Firebase para garantizar que cada usuario solo acceda a su propio historial de chat.
- Integrar un modelo de inteligencia artificial capaz de procesar los mensajes y generar respuestas coherentes basadas en el contexto previo.
- Asegurar que el sistema sea escalable y ofrezca baja latencia en la interacción.
- Monitorear y evaluar el rendimiento del sistema, documentando resultados y mejoras aplicadas.

1.3 Justificación del proyecto

La implementación de este chatbot con memoria contextual responde a una necesidad real en el ámbito empresarial: mejorar la experiencia de atención al cliente mediante un servicio digital rápido, personalizado y seguro. Gracias al uso de Firebase como base tecnológica, se obtiene un sistema de **alta disponibilidad, almacenamiento en tiempo real, seguridad robusta y escalabilidad automática**, características esenciales para un producto que debe responder de forma inmediata a múltiples usuarios en paralelo.

DISEÑO

2.1 Diseño del Modelo de Datos en Firebase Firestore

Uno de los puntos clave del proyecto es diseñar la base de datos de forma que sea **rápida, escalable y segura**. Para ello, se opta por una **estructura desnormalizada**, donde la información se almacena de manera que reduzca la necesidad de consultas complejas.

2.1.1 Estructura propuesta

La base de datos se organiza en **colecciones anidadas** de la siguiente forma:

```
/users/{userId}
/users/{userId}/sessions/{sessionId}
/users/{userId}/sessions/{sessionId}/messages/{messageId}
```

2.1.1.1 Colección users

Contiene la información básica de cada usuario.

Ejemplo de documento en users/{userId}:

```
{
  "uid": "abc123",
  "nombre": "Juan Pérez",
  "email": "juanperez@gmail.com",
  "fechaRegistro": "2025-09-24T14:32:00Z"
}
```

2.1.1.2 Subcolección sessions

Cada usuario puede tener múltiples sesiones de chat (ejemplo: consultas diferentes en distintos días).

Ejemplo de documento en users/{userId}/sessions/{sessionId}:

```
{
  "sessionId": "s001",
  "fechaInicio": "2025-09-24T15:00:00Z",
  "activo": true,
  "titulo": "Soporte técnico"
}
```

2.1.1.3 Subcolección messages

Dentro de cada sesión, se almacenan los mensajes del usuario y las respuestas del chatbot.

Ejemplo de documento en:

users/{userId}/sessions/{sessionId}/messages/{messageId}:

```
{
  "messageId": "m001",
  "autor": "usuario",
  "texto": "Hola, tengo un problema con mi pedido.",
  "timestamp": "2025-09-24T15:01:30Z"
}
```

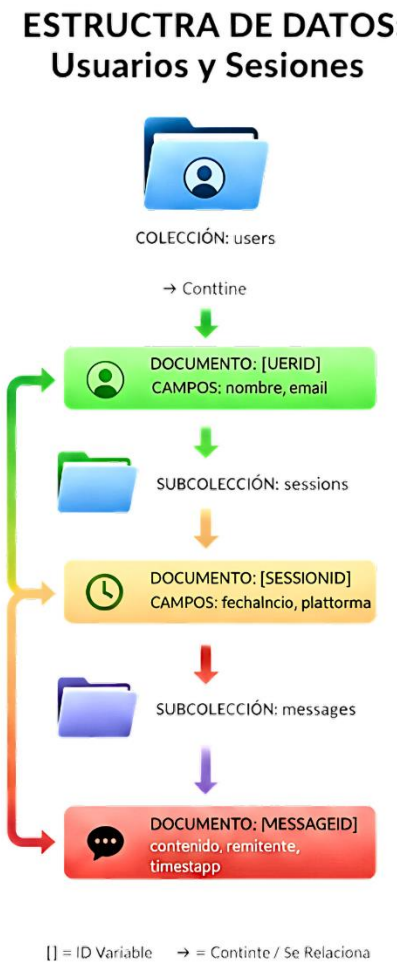
Y otro mensaje generado por el chatbot:

```
{
  "messageId": "m002",
  "autor": "chatbot",
  "texto": "Hola Juan, ¿podrías indicarme el número de pedido?",
  "timestamp": "2025-09-24T15:01:40Z"
}
```

2.2 Ventajas de este modelo

- **Consultas rápidas** → al estar los mensajes agrupados en la sesión del usuario, se pueden recuperar en tiempo real sin uniones complejas.
- **Escalabilidad** → Firebase Firestore permite manejar millones de documentos de esta forma sin afectar el rendimiento.
- **Contexto mantenido** → basta con leer los últimos mensajes de una sesión para reconstruir el historial.
- **Seguridad simplificada** → las reglas pueden aplicarse directamente sobre la colección `/users/{userId}`.

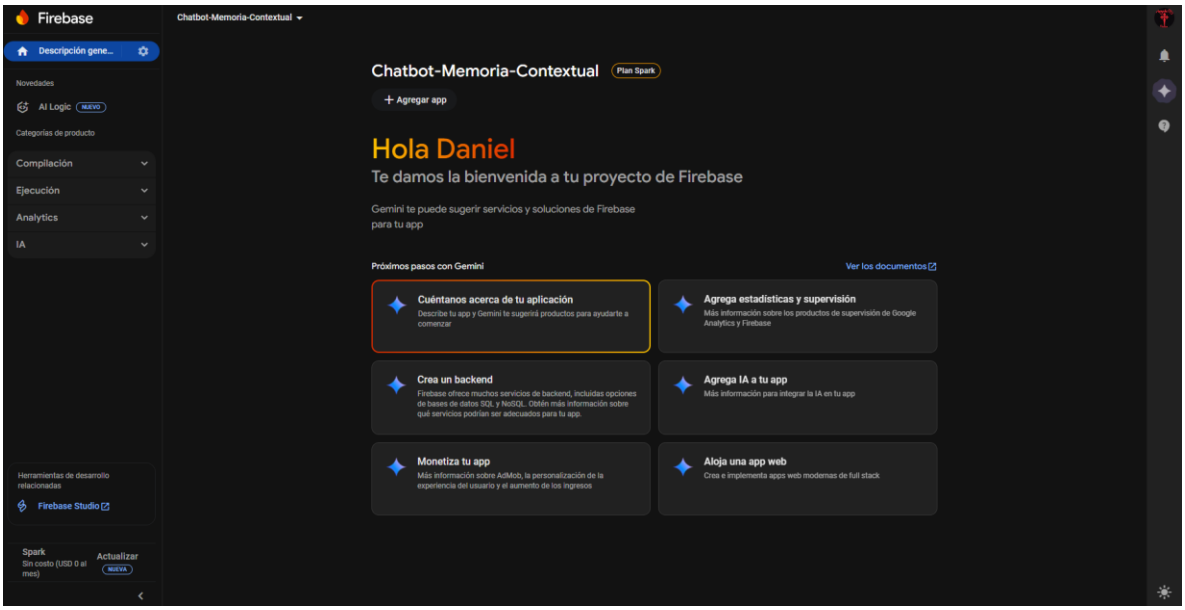
2.3 Diagrama de la estructura



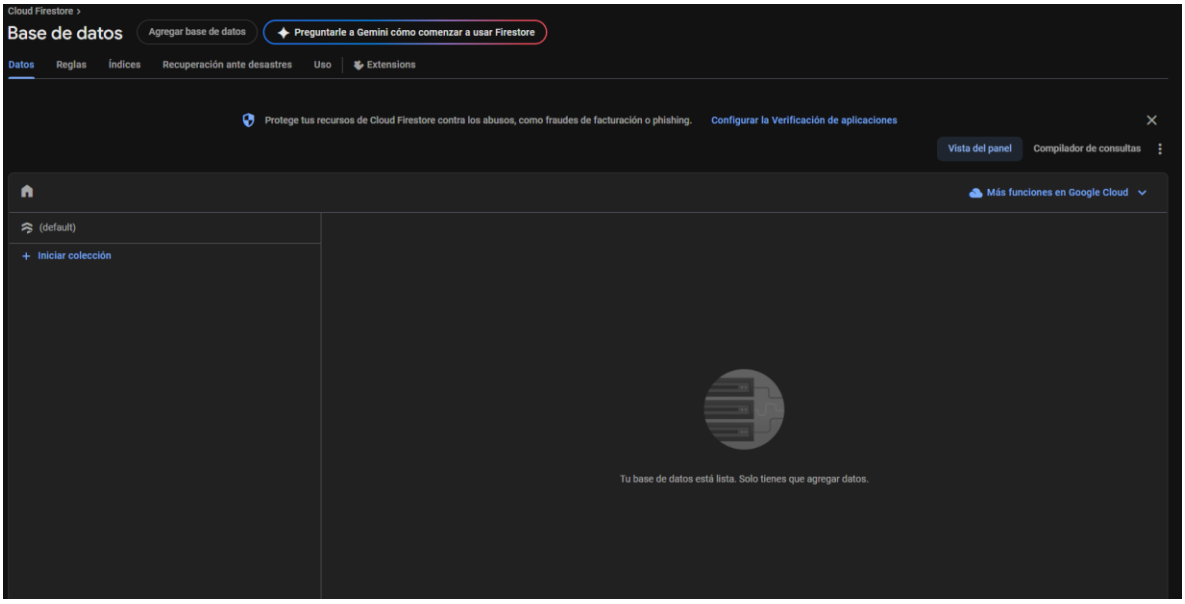
INGENIERIA DEL PROYECTO

INGENIERÍA DEL PROYECTO

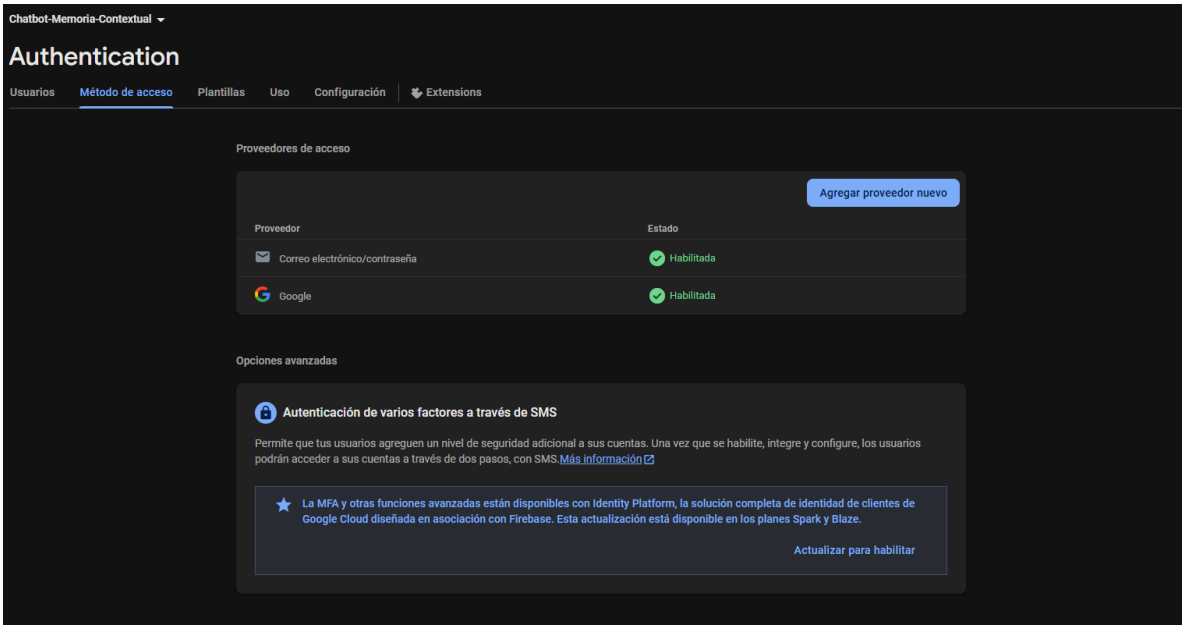
3.1 Creación del proyecto en FIREBASE



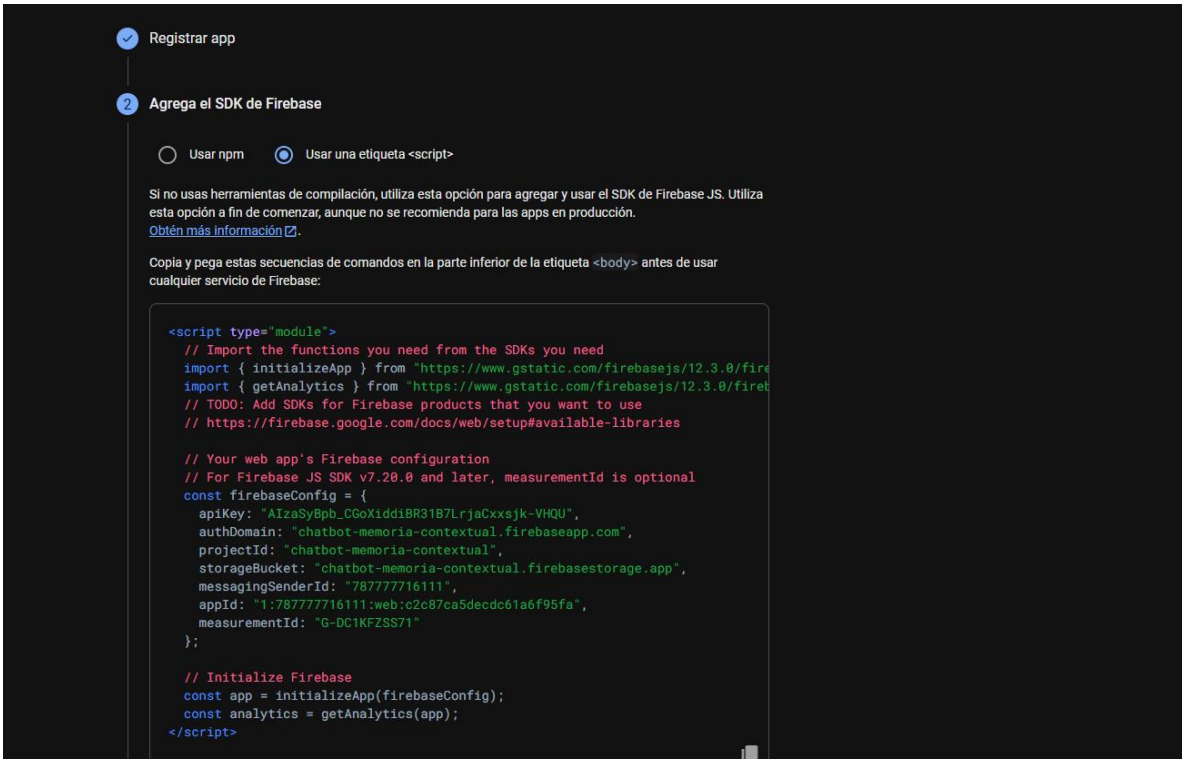
3.2 Creación de la base de datos



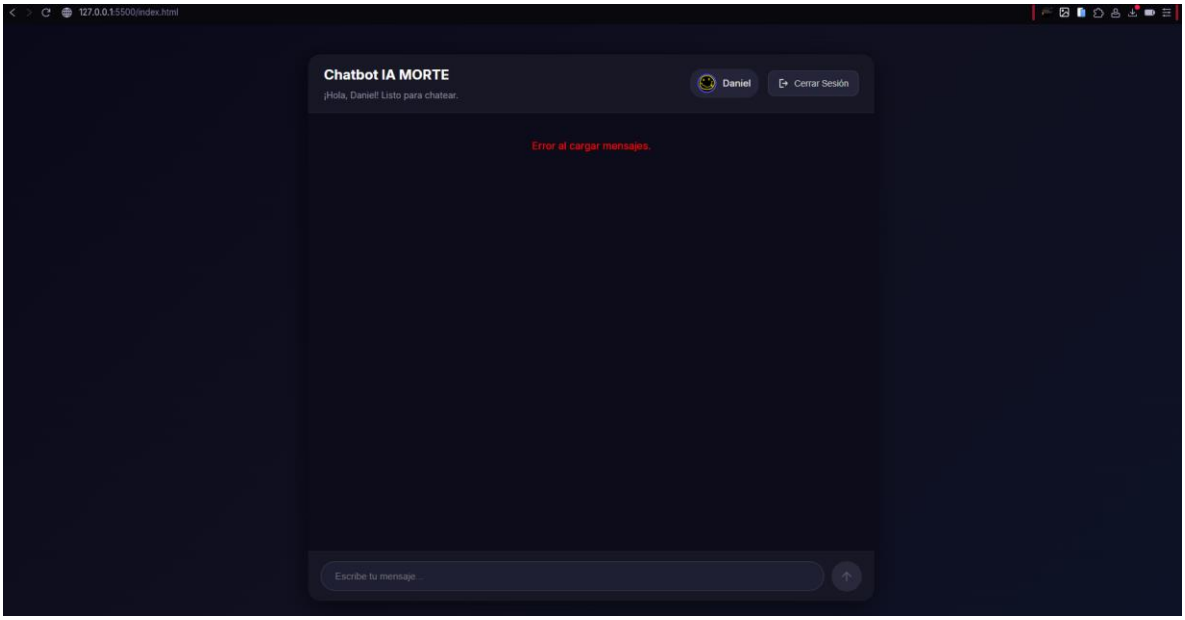
3.3 Agregamos la autenticación con correos y Google



3.4 Conectamos el proyecto con la app web



3.5 Verificamos la conexión correcta



3.6 REGLAS DE SEGURIDAD EN FIRESTORE

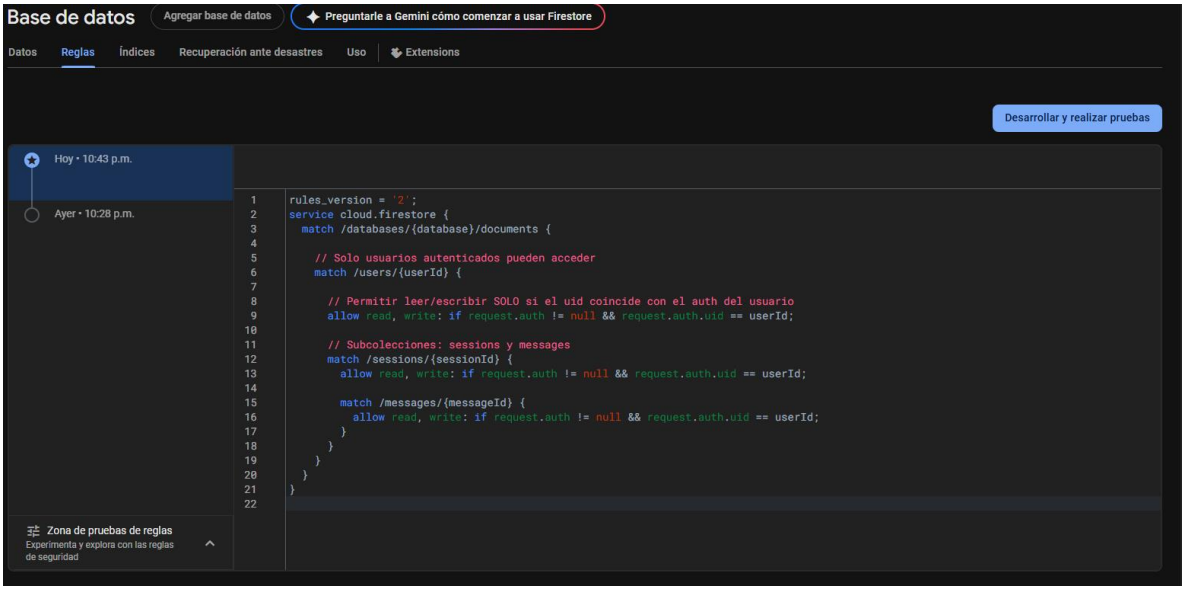
3.6.1 Importancia de las reglas

Las **reglas de seguridad** de Firestore permiten controlar quién puede leer y escribir en cada colección/documento.

En este proyecto, necesitamos:

- Que cada usuario acceda **solo a su propia colección de chats**.
- Que **nadie pueda ver ni modificar datos de otros usuarios**.
- Que las operaciones (lectura y escritura) estén vinculadas a la autenticación de Firebase.

3.6.2 Regla básica para colección users



3.6.2.1 Explicación de la regla

- `request.auth != null` → asegura que el usuario esté autenticado.
- `request.auth.uid == userId` → compara el UID del usuario autenticado con el UID en la ruta `/users/{userId}`.
- Esto garantiza que un usuario solo pueda acceder a **su propia rama de datos**.
- Las subcolecciones `sessions` y `messages` heredan la restricción, reforzando la seguridad.

3.7 Implementación del Chatbot con IA

3.7.1 Flujo General del Chatbot

1) **Usuario escribe mensaje** → se guarda en Firestore (colección `chats/{userId}/messages`).
2) **Cloud Function (trigger) escucha nuevos mensajes** → detecta si el mensaje es del usuario.
3) **La función llama a una API de IA (ej. OpenAI)** → genera la respuesta.

4) **La respuesta se guarda en Firestore** → en la misma sesión bajo `sender: 'bot'`.
5) **El frontend ya escucha en tiempo real** → muestra la respuesta automáticamente.

3.7.2 Configuración de Firebase Functions

En la terminal (dentro de la carpeta `PROYECTO - FIREBASE`) ejecutamos:

```
firebase init functions
```

Y ahí te va a preguntar varias cosas:

- **¿Qué lenguaje?** → elegí **JavaScript**.
- **¿Querés usar ESLint?** (te ayuda a detectar errores) → poné **No** para que sea más simple.
- **¿Querés instalar dependencias ahora?** → poné **Yes**.

3.8 CHATBOT EN FUNCIONAMIENTO