Arquitectura de un sistema basado en ia para la gestión de incidencias en call centers con aws

Yoshimar Hernández Badillo

25 de febrero de 2025

Resumen

En el ámbito laboral, específicamente en el área de atención a clientes, los *call centers* se enfrentan a al desafío de gestionar grandes cantidades de incidencias y un manejo eficiente en tiempos de respuesta y calidad de las mismas, además de una asignación adecuada de recursos técnicos (trabajadores e infraestructura). Este reto se puede complicar bajo contextos de alta demanda donde la complejidad de los problemas es mayor dependiendo del negocio y la cantidad de clientes; incidencias que se resuelven con una respuesta estandarizada hasta algunas que requieren intervención técnica especializada.

Palabras clave: Arquitectura Cloud, Optimización, Lambda, LLMs, AWS.

1. Objetivos

- Clasificación inteligente: Desarrollar un chatbot que sea capaz de analizar la consulta del cliente por medio de lenguaje natural, para posteriormente clasificarla en distintos niveles de complejidad.
    - Primer nivel: Incidencias sencillas y recurrentes que pueden ser resueltas mediante un número determinado de pasos a seguir o procedimientos ya estandarizados.
    - Segundo y tercer nivel: Problemas que requieren atención directa de un técnico especializado o una solución en sitio.
- Optimización de recursos: Reducir el tiempo de respuesta y evitar la saturación de solicitudes que son enviadas al personal técnico bajo el primer nivel principalmente.
- Mejora continua: Recopilar y analizar los datos de interacción (descripciones de las incidencias, soluciones aplicadas, y tiempos de resolución) para optimizar tanto el modelo de IA como los procesos del *call center*.

Con la implementación del chatbot se busca agilizar la atención al cliente, pero también optimizar los costos operativos y aumentar la satisfacción del usuario final (cliente), ofreciéndole respuestas más rápidas a casos sencillos y escalando de forma **inteligente** aquellas incidencias que requieren un enfoque más especializado e **interacción humana**. Dos conceptos fundamentales en este modelo de negocio: optimización e interacción humana.

2. Selección de tecnologías

Para realizar adecuadamente el desarrollo de esta aplicación, se consideró que como empresa MICROFORMAS se trabaja con **AWS** como infraestructura en la nube, por lo que se cuenta con conocimiento, familiaridad y recursos disponibles dándole prioridad a esta tecnología. También se llevó a cabo un análisis de los recursos con los que cuenta **AWS** con el objetivo de construir una arquitectura escalable, flexible y rentable, dando prioridad al equilibrio entre capacidad de respuesta y bajos costos.

2.0.1. Tecnologías evaluadas

- Procesamiento del Lenguaje Natural (PLN): Se optó por usar **Amazon Bedrock**, pues cuenta con acceso a modelos de lenguaje ya preentrenados de alto rendimiento (como **Amazon Titan**, **Claude** o **GPT-3.5**). Uno de los puntos considerados en esta tecnología fue la posibilidad de poder hacer *fine tuning* utilizando datasets propios. Esto ayuda a que las respuestas generadas por el chatbot no solo sean mostradas en lenguaje natural, sino también alineadas con los procesos internos, valores y misión de la empresa (*call center*).
- Infraestructura Serverless: Para gestionar de manera adecuada el volumen de solicitudes y garantizar escalabilidad, se decidió hacer uso de **AWS Lambda** y **API Gateway**. Ambos servicios permiten ejecutar funciones de procesamiento según sea la demanda y contar con endpoints seguros manejando un modelo de pago por consumo, ayudando a optimizar recursos en este tipo de escenarios de carga variable.
- Almacenamiento y procesamiento de datos: El desarrollo del chatbot requiere almacenar y gestionar tanto datos estructurados como no es-

tructurados. Se optó por usar **DynamoDB** para el manejo de datos estructurados (como por ejemplo un catálogo de incidencias y respuestas predefinidas) y de **Amazon S3** para el almacenamiento de descripciones y otros datos históricos ya mencionados. Servicios como **AWS Glue** y **Athena** se elegieron como opciones para la realización de procesos ETL y consultas.

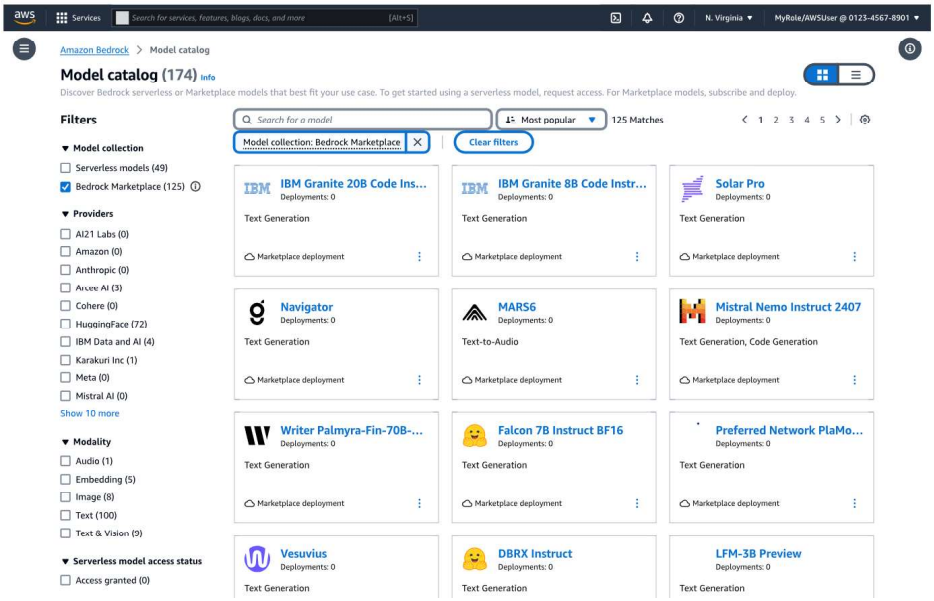

Figura 1: AWS Bedrock

2.1. Optimización y depuración de la solución:

Durante el análisis de dichas tecnologías se pasó por un proceso de selección con el fin de quedarse con las más adecuadas para el proyecto, dándole prioridad, como se mencionó anteriormente, a la optimización de recursos y costos. Algunas de las decisiones clave fueron:

- Centralización del procesamiento en **Amazon Bedrock**: Al utilizar este recurso para el análisis y generación de respuestas, se evita la complejidad (infraestructura, costos y tiempo) asociados al entrenamiento de modelos de IA desde cero. Uno de los recursos que quedaron limitados para su uso

en este proyecto fue **Amazon SageMaker**. La ventaja de poder hacer *fine tuning* en Bedrock permite entrenar y personalizar el modelo con datos propios.

- Estrategia híbrida de respuestas: Para reducir el uso del modelo (LLM: **Amazon Titan, Claude, GPT-3.5**), y por ende reducir el consumo de tokens (que se ve reflejado en la parte económica), se propuso almacenar en **DynamoDB** un conjunto de respuestas predefinidas para incidencias comunes y sencillas. De esta manera, **cuando la consulta del cliente coincida con una respuesta predefinida (por los históricos del *call center*), se responde directamente sin necesidad de invocar al LLM**, resultando en una solución más eficiente pero sobre todo económica.
- Adopción de una Arquitectura Serverless: La elección de **AWS Lambda** y **API Gateway** no solo simplifica la gestión de la infraestructura, sino que permite un escalado automático basado en el número de interacciones diarias. Esto es especialmente relevante en un entorno de *call center*, donde la carga puede variar significativamente a lo largo del día y de la semana.

Esta combinación de tecnologías garantizan una solución robusta, flexible y alineada con los objetivos de la empresa.

2.2. Implementación de los recursos de AWS

La solución integral y la que se seleccionó después de analizar las distintas opciones exitentes en el mercado, se estructura en varias capas integrando los distintos servicios de AWS logrando un flujo eficiente y escalable. A continuación, se detalla cómo se implementarán los recursos:

- Capa de usuario:
  - Aplicación web/móvil (*Frontend*): Se desarrollará una aplicación web progresiva (PWA) utilizando tecnologías como **React**, garantizando una experiencia de usuario fluida y consistente tanto en móvil como en escritorio. Esta aplicación se desplegará usando **AWS Amplify**, facilitando la integración continua.

- Capa de API y lógica de negocio:
    - API Gateway: Este servicio contará con endpoints seguros que permitirán la comunicación entre el *frontend* y los servicios de *backend*, asegurando que las solicitudes sean correctamente enrutadas y validadas.
    - AWS Lambda: Se implementarán varias funciones **Lambda** para manejar la lógica del sistema:
        - Función de preconsulta: Antes de mandar llamar al modelo de IA, se realizará una consulta a **DynamoDB** para identificar si la incidencia que generó el cliente coincide con una respuesta predefinida obtenida de los datos históricos.
        - Función de clasificación y respuesta: En caso de que no se encuentre dicha coincidencia, se invoca a **Amazon Bedrock** para analizar la solicitud del cliente en lenguaje natural. El modelo clasifica la incidencia en uno de tres niveles:
            - Primer nivel: Resuelve incidencias simples mediante una respuesta generada obtenida del *dataset* o directamente por el modelo de IA, paso por paso.
            - Segundo/Tercer nivel: Si la respuesta inicial no resuelve el problema o el error es más complejo, se activa la opción de escalar la incidencia a un técnico.
            - Función de enrutamiento de llamadas: En situaciones en las que se requiera una intervención humana, esta función **Lambda** se encargará de gestionar la conexión del cliente con un técnico, usando la API (telefonía) o la infraestructura del *call center*, evitando el uso de servicios adicionales como **Amazon Connect**.
- Capa de inteligencia artificial:
    - Amazon Bedrock: Este servicio es el núcleo de la solución de PLN. Se utiliza para:
        - Analizar y comprender las consultas en lenguaje natural.
        - Clasificar las incidencias en función de su complejidad.

        - Generar respuestas de primer nivel cuando sea posible.

- Capa de datos y almacenamiento:

    - DynamoDB: Se empleará para almacenar datos estructurados, tales como el catálogo de incidencias conocidas y las respuestas predefinidas (muy similar a las FAQs). Esta base de datos NoSQL nos permite un acceso rápido y escalable, bastante idóneo para gestionar las incidencias en tiempo real.
    - Amazon S3: **S3** se usará para almacenar los datos no estructurados, como descripciones de incidencias y registros históricos. Esto ayudará al análisis posterior y la mejora continua del sistema.
    - AWS Glue y Athena: Ambos servicios se utilizarán para procesar y analizar grandes volúmenes de datos. **AWS Glue** para la extracción, transformación y carga (ETL) de los datos almacenados previamente en **S3**, mientras que **Athena** permitirá realizar consultas SQL para la obtención de insights sobre el rendimiento del sistema y las incidencias.

- Capa de monitoreo y retroalimentación (opcional):

    - Kinesis Data Firehose: Este servicio de AWS facilitará el manejo de datos en tiempo real, ayudando a capturar flujos de información sobre las interacciones y el comportamiento del sistema.
    - CloudWatch y QuickSight: Se podrían implementar para la monitorización y visualización de métricas, generando *dashboards* y alertas que permitan identificar cualquier anomalía o mejora en la atención al cliente.

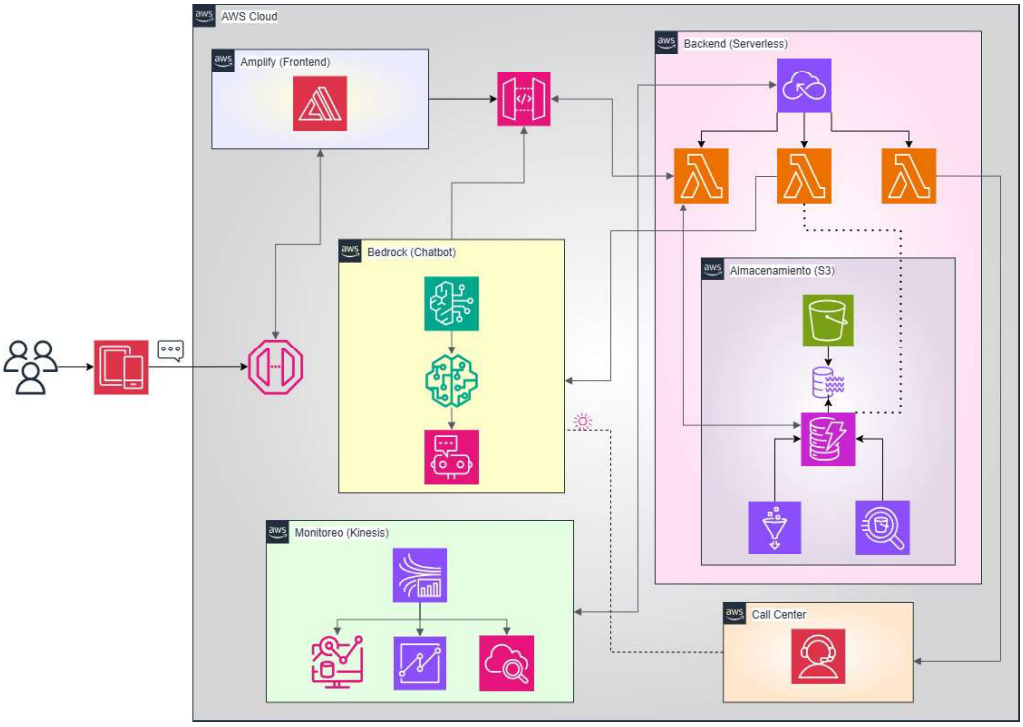

Figura 2: Diagrama de la arquitectura en AWS

2.3. Caso de uso como ejemplo en call centers y soporte técnico

Para ejemplificar y visualizar de mejor manera la aplicación, a continuación se muestra el siguiente caso de uso:

Escenario: Un cliente de una tienda departamental tiene el siguiente problema en su punto de venta: la impresora, a pesar de tener papel, no imprime los tickets. El sistema ha mostrado en pantalla un “Error 456” al intentar imprimir.

1. Interacción inicial: El cliente ingresa a la aplicación (móvil o de escritorio) y describe su problema en lenguaje natural a través del chatbot.

2. Preconsulta y clasificación: La solicitud se envía a través de **API Gateway** a una función **Lambda** que primero consulta en la base de datos (**DynamoDB**) si el problema corresponde a una incidencia conocida con solución predefinida.

    - Si encuentra una coincidencia, el sistema responde inmediatamente con los pasos a seguir para resolver el problema, evitando el uso del modelo de IA y, por lo tanto, reduciendo costos.
    - Si no se encuentra coincidencia alguna, la solicitud se envía a **Amazon Bedrock** para que, mediante PLN, analice el mensaje y clasifique la incidencia en los niveles existentes.

3. Generación de respuesta y escalado:

    - Primer nivel: Si la incidencia es sencilla, el modelo genera una respuesta paso a paso. El cliente recibe las instrucciones y, en caso de que sean suficientes, la interacción concluye.
    - Segundo/Tercer nivel: Si la solución propuesta no resuelve el problema o si la incidencia es compleja, el chatbot automáticamente marca la consulta como de nivel superior y muestra un botón de "Conectar con un técnico". Al presionar este botón, se invoca otra función **Lambda** encargada de enrutar la solicitud a un técnico mediante una API del sistema telefónico o del *call center*.

Beneficios:

- Optimización de recursos: Al resolver automáticamente las incidencias más sencillas, se reduce la carga de trabajo del personal técnico para incidencias de primer nivel.
- Mejora de la experiencia del cliente: Se garantiza una respuesta rápida y, en caso necesario, se escalada la incidencia de forma oportuna sin perder la interacción humana.
- Recopilación de datos: Cada interacción se almacena para analizar y retroalimentar el modelo, permitiendo mejoras continuas en el proceso.

2.4. Estimación de costos aproximados para distintos volúmenes de clientes diarios

Diseñar la aplicación bajo una solución *serverless* permite manejar el gasto en función del consumo real, ideal para entornos con alta variabilidad en la carga. La estrategia híbrida de usar respuestas predefinidas en **DynamoDB** y recurrir al modelo de IA solo cuando es necesario reduce significativamente el consumo de tokens y, por ende, los costos.

Aspectos a considerar:

- Amazon Bedrock (LLM): El costo aquí se determina por el número de tokens procesados. Por ejemplo, si se estima un consumo de 200 tokens por interacción (cliente-chatbot) y el costo referencial es de $0.02 por 1,000 tokens, cada interacción costaría aproximadamente $0.004.
  - Con 5,000 interacciones diarias, el costo sería alrededor de $20 diarios, y con 10,000 interacciones, cerca de $40 diarios.
  - Sin embargo, dado que muchas interacciones se resuelven mediante respuestas predefinidas, el uso real del LLM (tokens) se reducirá significativamente.
- API Gateway y AWS Lambda: Estos servicios tienen costos bajos por uso. Para un *call center* de tamaño medio, se pueden estimar costos en el rango de unos cuantos dólares diarios, que se ajustan según el volumen.
- Almacenamiento (DynamoDB y S3): Los costos se basan en el tamaño de la base de datos y el número de consultas. En un escenario moderado como en el caso de uso, los costos mensuales podrían estar entre $10 y $30, escalando con el volumen de datos.
- Servicios complementarios (Glue, Athena, CloudWatch, Kinesis, QuickSight): Estos últimos, como ya se mencionó, son opcionales y dependerán del uso y la cantidad de datos procesados, pero en conjunto podrían sumar entre $20 y $50 mensuales en un *call center* de tamaño medio.

Resumen: Para un escenario con 5,000 a 10,000 interacciones diarias, la arquitectura *serverless* podría mantener un costo total mensual en un rango aproximado de $200 a $350, escalando **linealmente** conforme aumente el

volumen de solicitudes. La flexibilidad del modelo permite ajustar el consumo y los gastos en función de la demanda real, lo que se traduce en una solución costo-beneficio.

2.5. Valor agregado del proyecto

El principal valor agregado de esta solución reside en su capacidad para transformar el proceso de atención en un *call center*, mediante la integración inteligente de tecnologías avanzadas y un diseño orientado a la eficiencia operativa:

- Reducción del tiempo de respuesta: Al automatizar la clasificación y resolución de incidencias de primer nivel, se disminuye el tiempo de espera para el cliente, mejorando la satisfacción y la experiencia general.
- Optimización de recursos técnicos: La aplicación filtra eficazmente las consultas simples, permitiendo que el personal técnico se concentre en problemas más complejos y críticos, lo que **incrementa la productividad y reduce los costos operativos**.
- Escalabilidad y flexibilidad: Por el enfoque *serverless* manejado, la arquitectura se adapta automáticamente a variaciones en el volumen de interacciones, garantizando un servicio continuo sin necesidad de invertir en infraestructura fija.
- Adaptabilidad y personalización: La capacidad de fine tuning en **Amazon Bedrock** permite personalizar el modelo según los datos históricos y la terminología propia del *call center*, **lo que se traduce en respuestas más precisas y alineadas con los protocolos internos de la empresa**.

Figura 3: Interfaz gráfica del chatbot (mockup)

Figura 4: Interfaz gráfica del chatbot (mockup)

2.6. Posibles mejoras y escalabilidad

Aunque el proyecto ya es robusto y escalable, siempre habrán oportunidades de mejora que pueden potenciar aún más el rendimiento y la adaptabilidad del sistema:

- Integración de análisis predictivo: Implementar modelos de *machine learning* para predecir incidencias recurrentes y anticipar necesidades de atención, permitiendo una intervención incluso antes de que se generen cuellos de botella.
- Expansión a múltiples canales: Ampliar la aplicación para integrar otros canales de comunicación (chat en vivo, redes sociales, envío de imágenes y su respectivo análisis OCR), diversificando la interacción y ampliando la cobertura de atención al cliente.
- Mejora en la gestión de incidencias complejas: Profundizar en el análisis de incidencias de segundo y tercer nivel para identificar patrones de problemas que puedan ser resueltas mediante el modelo de IA entrenado de una manera más óptima para dichas resoluciones, sin dejar de lado la opción de llamar a un técnico.

En resumen, la solución propuesta no solo resuelve de forma inmediata los desafíos actuales, sino que también establece una base sólida y escalable para futuras mejoras, asegurando que la empresa pueda adaptarse a las demandas cambiantes del mercado y a la evolución de las tecnologías de atención al cliente.