Agente Inteligente para

E-commerce

de Moda

Nombre: StyleGenius

La organización para este proyecto es un E-commerce de ropa y accesorios que opera exclusivamente a través de su plataforma digital.

Descripción de la organización:

StyleGenius cuenta con un amplio catálogo de productos para diversos estilos y Tamaños. Su plataforma actual integra funcionalidades estándar como catálogos de productos, un sistema de carrito de compras y una sección de reseñas de clientes. A pesar de tener un tráfico de usuarios considerable, la empresa enfrenta desafíos significativos en la conversión de visitas a ventas y en la construcción de una base de clientes leales.

# Identificación y descripción del problema:

El problema central que enfrenta StyleGenius es una alta tasa de abandono de carritos de compra. El análisis inicial sugiere que este problema se origina por varios factores interrelacionados que afectan la experiencia del usuario:

* Falta de personalización: Los usuarios reciben recomendaciones de productos genéricas que no se alinean con sus gustos, historial de navegación o compras previas.
* Incertidumbre en la decisión de compra: Los clientes dudan sobre qué talla elegir, cómo una prenda se ajustará a su tipo de cuerpo y si el estilo es el adecuado para ellos. Las reseñas, aunque útiles, a menudo son difíciles de contextualizar.
* Dificultad para crear conjuntos (outfits): Los usuarios tienen problemas para combinar diferentes prendas y accesorios disponibles en la tienda, lo que genera parálisis por análisis y el abandono del proceso de compra.

Este desafío impacta directamente en la organización al disminuir los ingresos potenciales y reducir la satisfacción y fidelización del cliente, elementos cruciales para la sostenibilidad en el competitivo mercado del E-commerce.

# Objetivos:

La solución de IA propuesta persigue los siguientes objetivos concretos y medibles:

* Objetivo Principal: Reducir la tasa de abandono de carritos en un porcentaje realista alrededor de un 15% en los primeros seis meses post-implementación.
* Objetivos Secundarios:
  + Incrementar la tasa de conversión de ventas.
  + Aumentar la satisfacción del cliente, medida a través de encuestas post-compra y la tasa de recompra.
  + Mejorar la eficiencia operativa automatizando la resolución de consultas frecuentes sobre tallas, estilos y combinaciones.

Datos disponibles o que se pueden obtener:

Para alimentar la solución de IA, se utilizarán (o simularán) las siguientes fuentes de datos:

* Datos Estructurados:
  + Catálogo de productos: Base de datos con SKU, descripciones, tallas disponibles, material, precios, imágenes y stock en tiempo real.
  + Historial de transacciones: Registros de compras por usuario.
  + Datos de navegación del usuario: Productos vistos, clics, tiempo en página, ítems agregados al carrito (incluso si se abandonan).
* Datos No Estructurados:
  + Reseñas de clientes: Comentarios en texto sobre productos, que contienen información valiosa sobre el ajuste ("corre pequeño", "es más holgado de lo que parece"), la calidad y la percepción del estilo.
  + Consultas de soporte: Registros de preguntas frecuentes realizadas por los usuarios al equipo de atención al cliente.

Estos datos son fundamentales para entrenar y enriquecer las respuestas del agente inteligente.

Restricciones o requerimientos particulares:

El desarrollo de la solución deberá considerar las siguientes condiciones:

* Integración Tecnológica: La solución debe ser compatible con la arquitectura de la plataforma de E-commerce existente, posiblemente a través de APIs.
* Privacidad de Datos: Se debe garantizar el manejo ético y seguro de los datos de los usuarios, cumpliendo con las normativas de protección de datos vigentes.
* Rendimiento: El agente de IA debe proporcionar respuestas en tiempo real para no afectar negativamente la experiencia de navegación del usuario.
* Escalabilidad: La arquitectura debe ser capaz de soportar un aumento en el número de usuarios y productos sin degradar su rendimiento.

Motivación para el uso de agentes de IA, LLMs y RAG:

Problema de base de conocimiento dinámica: Las organizaciones necesitan acceder a información actualizada y específica que no está en los modelos pre-entrenados

Prevención de alucinaciones: RAG asegura que las respuestas se basen en documentos verificados, no en la memoria del modelo

Eficiencia en costos: Evita el fine-tuning costoso permitiendo añadir nuevo conocimiento vía documentos

Trazabilidad: Cada respuesta puede ser rastreada a sus fuentes originales

Escalabilidad: La arquitectura permite añadir múltiples fuentes de datos sin modificar el modelo base  
  
Apartados:

* 1. Análisis del caso organizacional:
  2. Acceso a información corporativa actualizada y específica
  3. Capacidad de consulta natural sobre documentos internos
  4. Control sobre las fuentes de información utilizadas
  5. Monitoreo del performance del sistema (similitudes, calidad de respuestas)
  6. Objetivos resueltos con IA:
  7. Búsqueda semántica: Encontrar documentos relevantes basado en significado, no solo palabras clave
  8. Generación contextual: Respuestas precisas basadas en el contexto recuperado
  9. Análisis de similitudes: Identificar relaciones entre documentos del conocimiento organizacional
  10. Implementación técnica utilizada:
  11. Token de GitHub: Creaste un token personal de GitHub con acceso a GitHub AI Models
  12. GPT-4o via GitHub: Utilizas openai/gpt-4o a través de la API de GitHub AI Models
  13. Infraestructura de tracing: LangSmith para monitoreo y debugging
  14. Embeddings custom: Implementación propia usando el modelo de chat para generación de vectores
  15. Formulación de prompts:   
      python
  16. system\_message = """Eres un asistente especializado que responde preguntas basándose ÚNICAMENTE en la información proporcionada en el contexto. Sigue estas reglas estrictamente:
  17. 1. Responde basándote exclusivamente en la información del contexto proporcionado
  18. 2. Si el contexto no contiene información suficiente, indica claramente qué aspectos no puedes responder
  19. 3. Sé preciso y conciso
  20. 4. Si diferentes documentos contienen información contradictoria, menciona esta discrepancia
  21. 5. Cita la información relevante del contexto cuando sea apropiado"""
  22. Justificación de la redacción:
  23. Instrucciones explícitas: Reglas numeradas y claras para evitar ambigüedades
  24. Enfatizar limitaciones: Previene alucinaciones al forzar al modelo a reconocer cuando no tiene información
  25. Manejo de conflictos: Instrucción específica para tratar información contradictoria
  26. Transparencia: Promueve la citación de fuentes para verificación
  27. Prompt de embedding (optimizado para consistencia):
  28. python
  29. prompt = f"""Convierte el siguiente texto en una representación numérica densa.
  30. Responde SOLO con una lista de números separados por comas, sin texto adicional."""
  31. Justificación: Instrucción simple y repetitiva que asegura formato consistente
  32. Diseño e implementación de pipeline RAG:   
      text
  33. Fuentes de Datos → Procesamiento → Almacenamiento → Recuperación → Generación
  34. Integración de fuentes:
  35. Internas: Documentos cargados via interfaz Streamlit con metadatos personalizados
  36. Externas: Modelo GPT-4o via GitHub AI Models API
  37. Metadatos: Sistema flexible para añadir información contextual (fuente, tema, autor)
  38. Flujo detallado:
  39. Ingesta: Texto + metadatos → Generación de embedding → Almacenamiento en memoria
  40. Recuperación: Query → Embedding → Búsqueda por similitud coseno → Top-K documentos
  41. Generación: Contexto + Query → Prompt engineering → LLM → Respuesta contextual
  42. Visualización: Análisis de similitudes, métricas de calidad, trazabilidad
  43. Arquitectura de la solución:   
      text
  44. ┌─────────────────┐ ┌──────────────────┐ ┌─────────────────┐
  45. │ INTERFAZ │ │ SISTEMA RAG │ │ MODELO LLM │
  46. │ STREAMLIT │◄──►│ │◄──►│ GITHUB AI │
  47. │ │ │ - Document Store │ │ │
  48. │ - Carga docs │ │ - Embedding Engine│ │ - GPT-4o │
  49. │ - Consultas │ │ - Vector Search │ │ - Embeddings │
  50. │ - Visualización │ │ - Prompt Manager │ │ │
  51. └─────────────────┘ └──────────────────┘ └─────────────────┘
  52. │ │
  53. │ │
  54. └───────┐ ┌───────┘
  55. │ │
  56. ┌──────────────────┐
  57. │ MONITOREO │
  58. │ LANGSMITH │
  59. │ - Tracing │
  60. │ - Metrics │
  61. └──────────────────┘
  62. Componentes clave e integración:
  63. Frontend (Streamlit): Interfaz completa para gestión del ciclo de vida RAG
  64. Motor RAG (GitHubAIClient + RAGSystem): Núcleo de procesamiento de embeddings y búsqueda
  65. Modelo LLM (GitHub AI Models): Proveedor de inteligencia artificial
  66. Sistema de monitoreo (LangSmith): Trazabilidad y debugging de ejecuciones
  67. Almacenamiento vectorial: Embeddings en memoria con búsqueda por similitud coseno
  68. Flujo de datos integrado:
  69. Usuario carga documento → Generación embedding → Almacenamiento en sesión
  70. Usuario hace consulta → Búsqueda semántica → Recuperación contexto → Generación respuesta
  71. Sistema monitorea → Logs en LangSmith → Métricas de performance → Feedback visual
  72. Esta arquitectura permite escalar añadiendo bases de datos vectoriales persistentes (Pinecone, Chroma) y múltiples fuentes de datos organizacionales.
  73. Justificación de las Decisiones de Diseño:
  74. La arquitectura y las herramientas seleccionadas fueron escogidas estratégicamente para resolver los desafíos específicos de un E-commerce con un catálogo dinámico y la necesidad de respuestas precisas y personalizadas.
  75. Adopción de la Arquitectura RAG (Retrieval-Augmented Generation)
  76. Decisión: Se optó por un pipeline RAG en lugar de un reentrenamiento (*fine-tuning*) del modelo de lenguaje.
  77. Justificación: El principal motivo es la naturaleza dinámica de los datos en StyleGenius. El stock, los precios y las reseñas de clientes cambian constantemente. Un modelo con *fine-tuning* quedaría obsoleto rápidamente y su reentrenamiento sería costoso y lento. RAG permite que el agente consulte una base de conocimiento externa (la base de datos vectorial) en tiempo real. Esto asegura que cada recomendación se base en la información más reciente, minimizando el riesgo de "alucinaciones" y evitando la frustración del cliente por productos agotados.
  78. Uso del Framework LangChain para la Orquestación
  79. Decisión: Se utilizó LangChain como el orquestador central de la lógica del agente.
  80. Justificación: LangChain actúa como el "sistema nervioso" de la solución, simplificando la conexión entre componentes dispares: la interfaz de usuario, el motor de búsqueda vectorial, el formateo de prompts y la llamada al LLM. Esto acelera el desarrollo, reduce el código repetitivo y crea una lógica modular (Chains) que es más fácil de depurar y mantener. Su integración nativa con herramientas de monitoreo como LangSmith fue un factor decisivo.
  81. Implementación de LangSmith para Trazabilidad y Monitoreo
  82. Decisión: Integrar LangSmith desde la fase de desarrollo.
  83. Justificación: Para una solución de IA en un entorno comercial, la observabilidad no es opcional. LangSmith proporciona una traza detallada de cada interacción, permitiéndonos visualizar qué documentos fueron recuperados, cómo se construyó el prompt final y por qué el LLM generó una respuesta específica. Esta capacidad es fundamental para diagnosticar errores, evaluar la calidad de la recuperación de datos y optimizar el rendimiento del agente de forma continua.
  84. 2. Esquemas y Diagramas de la Arquitectura (IE8)
  85. El siguiente diagrama visualiza la arquitectura de la solución y el flujo de información, desde la consulta del usuario hasta la respuesta generada.
  86. Descripción del Flujo de Datos:
  87. Consulta: El usuario realiza una pregunta en lenguaje natural a través de la Interfaz de Streamlit.
  88. Recuperación y Aumento: El Sistema RAG, orquestado por LangChain, convierte la consulta en un vector y busca en la Base de Datos Vectorial los documentos (descripciones de productos, reseñas) más relevantes semánticamente. Esta información recuperada se "aumenta" al prompt original.
  89. Generación: El prompt enriquecido se envía al Modelo LLM (GPT-4o), que genera una respuesta precisa y contextualizada. Todo el proceso es monitoreado por LangSmith.
  90. Conclusiones Técnicas y Referencias
  91. Conclusiones: La arquitectura diseñada es robusta, escalable y aborda directamente los requerimientos del caso StyleGenius. La elección deliberada de RAG sobre otras técnicas garantiza la fiabilidad y actualidad de la información, que es el pilar para construir la confianza del cliente. La modularidad aportada por LangChain y la capacidad de monitoreo de LangSmith transforman este proyecto de un simple prototipo a una solución preparada para un entorno de producción, sentando las bases para futuras mejoras como la memoria conversacional y la integración con otras herramientas.
  92. Referencias (Formato APA 7):
  93. LangChain. (2024). *LangChain Documentation*. Recuperado de <https://python.langchain.com>
  94. OpenAI. (2024). *GPT-4o*. OpenAI. Recuperado de <https://openai.com/index/hello-gpt-4o/>