**Caso: “ShopIA” — Chatbot de IA integrado al e-commerce de Tienda Aurora**

**1) Contexto**

Tienda Aurora es un e-commerce de moda con 4 sucursales físicas (Región Metropolitana y V Región). El tráfico móvil es 72% y el 35% de las consultas a soporte son repetitivas (envíos, cambios, medios de pago, disponibilidad por talla). La empresa busca reducir tiempos de respuesta, aumentar conversión y ofrecer soporte 24/7, integrando un **chatbot de IA** embebido en el sitio, conectado a **stock en tiempo real**, **ubicación/horarios de sucursales** y **catálogo de ofertas/cupones**.

**2) Problema a resolver**

* Alto volumen de FAQs repetidas satura al equipo de soporte.
* Carritos abandonados por dudas simples no resueltas a tiempo.
* Los clientes preguntan por **disponibilidad** de tallas en **sucursal** específica y por **promociones vigentes**; hoy la respuesta requiere intervención humana.
* Falta un canal automatizado para **consultar estado de pedido** (autenticado) y **escalar a humano** cuando la IA no resuelve.
* Resolución de preguntas frecuentes

**3) Objetivo del caso**

Diseñar, construir y desplegar un **chatbot de IA** que:

1. Responda **FAQs** (envíos, cambios, pagos, tiempos de despacho).
2. Consulte **stock** por producto/variación (talla/color) y **por sucursal** o en bodega central.
3. Informe **sucursales** (dirección, horario, teléfono) y si un producto está disponible en la tienda más cercana al cliente.
4. Recomiende **ofertas** y valide **cupones** (si aplica).
5. Muestre **estado de pedido** para usuarios autenticados (o con verificación por correo + nº de pedido).
6. Escale a agente humano y cree **ticket** cuando no resuelva en 2 turnos.

**4) Alcance funcional del chatbot (MVP)**

* **Canal**: widget web embebido (SPA/SSR).
* **FAQs administrables**: desde panel (archivo/colección), con búsqueda semántica + RAG opcional.
* **Stock & sucursales**:
  + GET /api/stock?sku=...&talla=... → cantidad total y por sucursal.
  + GET /api/sucursales?near=lat,long → sucursal más cercana, horarios y teléfono.
* **Ofertas**:
  + GET /api/ofertas?categoria=... y GET /api/cupones/{code}/validate.
* **Pedidos** (usuario autenticado o verificado):
  + GET /api/pedidos/{id} → estado y traza (creado→pagado→preparado→enviado→entregado).
* **Fallback y escalamiento**: crea ticket POST /api/tickets con resumen y correo del cliente.
* **Trazabilidad y guardrails**: logging de turnos, límites de tokens/costo, encuesta CSAT (1–5).

**5) Historias de usuario (extracto + criterios de aceptación)**

* **US-Chat-01 (FAQs envíos/cambios)**:
  + *Como cliente* quiero preguntar por políticas de envío y cambios.
  + **CA**: La IA responde en ≤2 s con fuente (FAQ oficial) y un enlace a la política; si hay ambigüedad (ej. “cambio por talla usada”), pide precisión.
* **US-Chat-02 (Stock por sucursal)**:
  + *Como cliente* quiero saber si el polerón SKU AUR-H123, talla M, está en la sucursal Viña.
  + **CA**: La IA confirma disponibilidad exacta (ej. “3 unidades”) o sugiere alternativa cercana; ofrece reservar/retirar si el flujo existe.
* **US-Chat-03 (Ofertas activas)**:
  + *Como cliente* quiero conocer ofertas en “chaquetas mujer” y si mi cupón “OTOÑO10” aplica.
  + **CA**: Lista top 3 ofertas con % y vigencia; valida/explica condiciones del cupón (mínimo de compra, categorías excluidas).
* **US-Chat-04 (Estado de pedido)**:
  + *Como cliente autenticado* quiero saber el estado del pedido #10234.
  + **CA**: La IA muestra estado y último hito; si no hay sesión, verifica con correo+ID (enmascara datos).
* **US-Chat-05 (Escalamiento)**:
  + *Como cliente* quiero hablar con un humano si la IA no resuelve.
  + **CA**: Tras 2 turnos fallidos crea ticket con resumen+prioridad y entrega nº de ticket.

Estas historias extienden las US del enunciado (búsqueda, carrito, pagos sandbox, estado de pedido, chatbot).

**6) Reglas de negocio clave**

* **Stock consistente**: respuestas siempre desde la BD transaccional (no cache desactualizado para cantidades).
* **Acceso a pedidos**: solo del usuario actual; verificación por sesión JWT o one-time token (email + nº pedido).
* **Ofertas/cupones**: aplicar reglas por categoría/fecha/uso; explicar por qué un cupón **no** aplica.
* **Políticas**: las respuestas “normativas” deben provenir del repositorio de políticas vigente (versión y fecha).

**7) Arquitectura de referencia**

* **Frontend**: React/Next con widget de chat; i18n CLP, IVA y formatos locales.
* **Backend API**: Node/Nest o Python/FastAPI (REST/GraphQL) con endpoints de catálogo, stock, sucursales, pedidos, tickets.
* **Chatbot IA**:
  + Orquestador (middleware) + proveedor LLM (externo u open-source)
  + **RAG**: embeddings de FAQs/políticas en vector store.
  + **Conectores** seguros al backend para stock, pedidos, ofertas.
* **Datos**: Postgres/MySQL PaaS; almacenamiento de medios (S3/Blob); CDN opcional.
* **Pagos**: Transbank o Mercado Pago en sandbox (IPN/Webhooks).
* **DevOps**: PaaS (Amplify/App Service/App Engine/Render/Railway), CI/CD, logs/métricas/alertas, HTTPS, OWASP Top-10.

**8) Modelo de datos (extensión mínima para el caso)**

* **Sucursal**(id, nombre, dirección, comuna, región, lat, long, horario\_json, teléfono).
* **InventarioSucursal**(id, sucursal\_id, producto\_id, variación\_id, stock).
* **Oferta**(id, título, descripción, categoría\_id?, descuento\_pct, fecha\_inicio, fecha\_fin, condiciones\_md).
* **Cupon**(id, code, descuento\_pct/valor, min\_compra, usos\_max, usos\_hechos, reglas\_json, vigente\_bool).
* **FAQ**(id, categoría, pregunta, respuesta\_md, actualizado\_en).
* **Conversación**(id, usuario\_id?, mensajes\_json, motivo, csat?, creado\_en).
* **Ticket**(id, usuario\_id?, resumen, prioridad, estado, creado\_en).
* (Se acopla al **Usuario/Producto/Pedido/Pago** definidos en el enunciado.)

**9) Diseño de intents y entidades (NLU)**

* **Intents**: consultar\_envio, politica\_cambios, medio\_pago, consultar\_stock, stock\_por\_sucursal, buscar\_ofertas, validar\_cupon, estado\_pedido, horario\_sucursal, direccion\_sucursal, hablar\_humano.
* **Entidades**: sku, producto, talla, color, sucursal, comuna, categoria, cupon, pedido\_id, email.
* **Ejemplos**:
  + “¿Hay **AUR-H123** en **M** en la **sucursal Viña**?” → stock\_por\_sucursal(sku=AUR-H123, talla=M, sucursal=Viña)
  + “¿Qué **ofertas** hay en **chaquetas** esta semana?” → buscar\_ofertas(categoria=chaquetas)
  + “Mi **pedido 10234** ¿va en camino?” → estado\_pedido(pedido\_id=10234)

**10) Flujo de conversación (resumen)**

1. Captura intención + entidades → verifica sesión/consentimiento si requiere datos personales.
2. **RAG**: obtiene base normativa/FAQ; si necesita datos vivos (stock, pedidos, ofertas), llama al **conector** seguro.
3. Respuesta estructurada (con bullets/enlaces); ofrece **acción siguiente** (añadir al carrito, ver sucursal en mapa, aplicar cupón).
4. Si no resuelve en 2 turnos → **ticket** + handoff (correo/WhatsApp/cola de agente).

**11) Seguridad, privacidad y guardrails**

* HTTPS, secretos en vault, CORS estricto, rate-limiting por IP/usuario, auditoría.
* **Principio de minimización**: no mostrar PII; enmascarar email/teléfono; logs sin datos sensibles.
* **Idempotencia** en webhooks de pagos; control de acceso por rol (admin/soporte).
* **Mensajes de descargo** del chatbot (no decisiones financieras, etc.).

**12) KPIs de éxito**

* ↓ 40% tiempo medio de primera respuesta (FRT).
* ↓ 30% tickets de nivel 1 (FAQs).
* ↑ 8–12% tasa de conversión en sesiones con interacción del bot.
* CSAT del flujo de chat ≥ 4/5.
* Respuesta del bot ≤ 2 s P95 en FAQs; ≤ 3 s cuando consulta APIs de stock/pedidos.

**13) Plan de pruebas (enfocado en el chatbot)**

* **Unitarias**: parsing de intents y normalización de entidades (SKU, tallas).
* **Integración**: stock por sucursal y validación de cupones; estado de pedido autenticado.
* **E2E**: flujo “consulta stock → agregar al carrito → checkout sandbox → webhook → bot confirma estado”.
* **Carga**: 100 RPS sobre endpoints públicos (stock, ofertas, faq); latencia P95 < 800 ms.
* **Seguridad**: SAST/DAST básicos, pruebas de autorización en GET /pedidos/{id}.