Proyecto Chatbot Conversacional Multimodal para Servicio al Cliente (Banco Digital)

Grupo 1

1. Resumen ejecutivo

Se diseñó e implementó un **chatbot multimodal** orientado a banca digital que atiende consultas 24/7, procesa **texto** e **imágenes** (OCR + clasificación de documentos) y soporta **transacciones bancarias simuladas** (balance y transferencias) con **memoria de diálogo**, **escalamiento inteligente** a agentes humanos, **métricas de satisfacción** y **auditoría**.

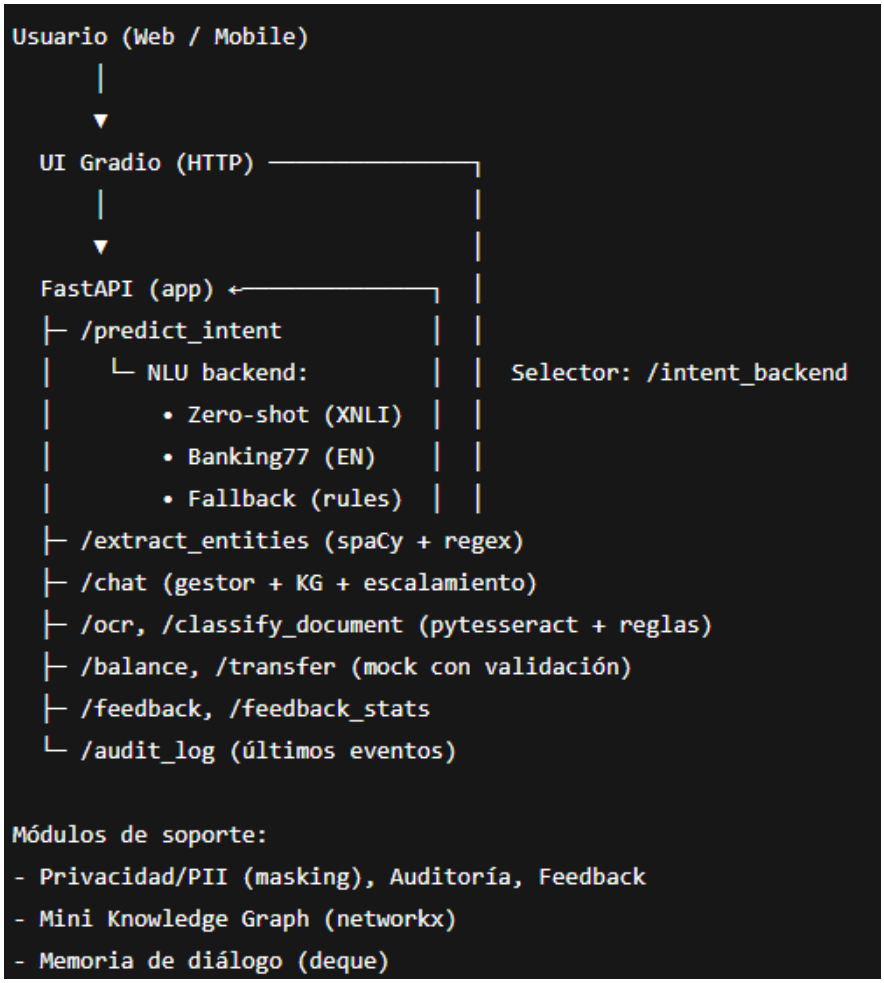
La solución expone una **API HTTP (FastAPI)** y una **UI en Gradio** (dos variantes: local por funciones y por **HTTP** contra la API). El pipeline NLU soporta **modelos reales** conmutables en caliente (zero-shot multilingüe o Banking77) y **fallback** robusto por palabras clave/regex.

2. Alcance y entregables

- **API FastAPI** con endpoints:
  /health, /predict_intent, /extract_entities, /chat, /ocr, /classify_document, /balance, /transfer, /feedback, /feedback_stats, /audit_log, /intent_backend (GET/POST).
- **NLU (intents)** con **selector dinámico**:
  - **Zero-shot multilingüe** (joeddav/xlm-roberta-large-xnli) — recomendado para español.
  - **Banking77** (inglés, mrm8488/bert-mini-finetuned-banking77).
  - **Fallback** por palabras clave.
- **NER**: spaCy (es_core_news_sm) + **regex** complementaria (monto, cuenta, fecha).
- **Gestor de diálogo** con **memoria**, enriquecimiento con **Mini-Knowledge Graph (networkx)** y **reglas de escalamiento** por baja confianza/intents críticos.
- **Visión por computador**: OCR (pytesseract) y **clasificación** heurística (cheque, identificación, voucher/depósito, otro).
- **Transacciones mock** con validación de **límites** y **fondos**, mascarado de PII y **auditoría**.
- **UI Gradio**:
  - UI “local” (invoca funciones internas).
  - UI **HTTP** (llama a la API usando BASE_URL).
  - **Panel de backend NLU** para cambiar modo en caliente.
- **Pruebas funcionales** (celdas de smoke test + pruebas de endpoints) y **demos** de OCR/Clasificación.

- **Privacidad/Seguridad**: sanitización (emails/números), trazabilidad (auditoría), CORS abierto (para demo).

3. Arquitectura de referencia



4. Componentes técnicos y decisiones

4.1 NLU (Intents)

- **Zero-shot multilingüe (XNLI)**: robusto para español; etiquetas bilingües + **overrides** en español (sinónimos/bigramas) para subir precisión y confianza mínima (≥0.75) cuando se detecta patrón fuerte.
- **Banking77 (EN)**: rápido y liviano; útil para prompts en inglés; mapeo refinado a intents canónicos.
- **Fallback**: cobertura por keywords en español para entornos sin GPU/transformers.
- **Selector dinámico**: endpoint /intent_backend (GET/POST) para cambiar en vivo entre fallback | zero_shot | banking77.

4.2 NER

- spaCy (es_core_news_sm) para entidades generales; **regex** para patrones bancarios (monto, cuenta, fecha).
- Resultado combinado: mayor recall en escenarios bancarios comunes.

4.3 Diálogo con memoria y escalamiento

- **Memoria breve** (deque) para antecedentes inmediatos.
- **Reglas de escalamiento**: triggers por baja confianza (<0.70) o intents críticos (complaint, charge_dispute, etc.).
- **Enriquecimiento por KG**: límites de transferencia y tiempos de envío de tarjeta → respuestas más informativas.

4.4 Visión (OCR + Clasificación)

- OCR con pytesseract (idioma configurable spa/eng).
- Clasificación **heurística** por palabras clave (cheque/ID/voucher/otro). Puntos de extensión para ML supervisado futuro.

4.5 Transacciones mock + Seguridad

- Validación de montos, límites diarios y fondos.
- **Máscara de PII** (emails/números) en logs y respuestas apropiadas.
- **Auditoría** de eventos (chat, intent, OCR, transfer, errores).

4.6 UI Gradio

- **UI HTTP** para consumir la API con BASE_URL (tanto local como túnel público/ngrok si se usa).
- Tabs: Chat, Transacciones, Documentos (OCR/Clasificación), Métricas, Auditoría.
- **Panel de backend**: gestión del NLU en caliente.

5. Instalación y ejecución (Colab)

1. **Ejecutar las celdas** del notebook en orden (base mínima → módulos → API → arranque → pruebas → UI).
2. **Arranque del servidor**: uvicorn en **hilo** y **puerto libre automático**. BASE_URL queda disponible (ej.: http://127.0.0.1:<PORT>).
3. **Probar /health** desde la UI (botón “Probar /health”) o vía requests.
4. **UI HTTP**: ejecutar la celda de Gradio (HTTP), verificar/editar BASE_URL y usar pestañas.

Opcional: Exponer vía **ngrok** configurando NGROK_TOKEN y conectando el túnel al mismo PORT.

6. **Validación y pruebas**
   - **Smoke test** de /health con espera activa hasta que el server esté listo.
   - **NLU/NER**: pruebas contra /predict_intent y /extract_entities con textos en español.
   - **Chat**: validación de respuesta, intent y banderas de escalamiento.
   - **Transacciones**: balance → transferencia OK / error por límite → balance actualizado.
   - **OCR/Clasificación**: imagen sintetizada para validar OCR y etiqueta.
   - **Feedback/Auditoría**: registro de satisfacción y lectura de últimos eventos.

7. **Métricas y evaluación**
   - **Satisfacción del cliente**: avg_score y helpful_rate expuestos en /feedback_stats y en la UI.
   - **Calidad del NLU**: inspección de confidence y tasa de escalamiento por intent.
   - **Telemetría/Auditoría**: conteo y tipo de eventos, errores por endpoint, trazabilidad de transacciones mock.

   Recomendado: centralizar métricas en un **dashboard** y definir **SLOs** (latencia, disponibilidad, exactitud NLU, % resoluciones sin escalamiento).

8. **Riesgos y limitaciones**
   - **Zero-shot** puede tener **latencia alta** sin GPU.
   - pytesseract requiere imágenes nítidas; sin dataset propio, la clasificación es **heurística**.
   - Transacciones son **mock**: no hay integración a core bancario.
   - Faltan **controles de seguridad/identidad** (auth, rate limiting, WAF) para producción.
   - KG es **mínimo** y requiere expansión (productos, tarifas, normativas).

9. **Conclusiones**
   - Se logró un **MVP robusto**, modular y **conmutable** entre modelos de NLU, que cubre texto e imágenes, soporta flujos bancarios básicos y ofrece telemetría esencial.
   - La arquitectura es **extensible** y lista para reemplazar módulos por equivalentes productivos (OCR comercial, NLU finetuned, RAG/LLM, KG empresarial).
   - La UI y la API permiten **validación rápida** con negocio y **demos** a stakeholders.

10. Recomendaciones (roadmap a producción)

1. **Seguridad**
    - **API Key/JWT/OAuth2**, CORS restringido.
    - **Rate limiting**, protecciones anti-abuso y validación de entrada estricta.
    - **Redacción PII** avanzada y almacenamiento **cifrado** de auditoría.

2. **NLU/NER**
    - Curar **dataset propio** y **finetuning** supervisado.
    - Evaluación (accuracy/F1 por intent) y **canary** entre backends.

3. **Diálogo**
    - Integrar **LLM** con memoria vectorial y **RAG** sobre FAQs/políticas.
    - Mejorar **gestión de contexto** y **slot filling**.

4. **Visión**
    - Entrenar clasificador de documentos (CNN/ViT/docformer) y mejorar OCR (EasyOCR/PaddleOCR).
    - Validaciones antifraude (manipulación de imágenes).

5. **Transacciones**
    - Abstraer “core banking adapter”, sandbox end-to-end y luego APIs reales (contratos y pruebas).

6. **Observabilidad**
    - Métricas/logs centralizados; **tracing**; dashboard de **CSAT**, AHT, escalamiento.

7. **DevOps/MLOps**
    - Docker, CI/CD, **tests automáticos** y monitoreo de **drift**.

8. **Conformidad**
    - Revisar normativa (KYC/AML), privacidad (GDPR/LOPD), retención y auditorías.