**Guía2. Desarrollo Proyecto APT**

**Asignatura Capstone**

|  |
| --- |
| **1. Resumen avance Proyecto APT** |
| A continuación, encontrarás distintos campos que deberás completar con la información solicitada. |

|  |  |
| --- | --- |
| Resumen de avance proyecto APT | Durante esta fase se han completado hitos importantes en el desarrollo del Tótem Retail Inteligente:   * **Backend**: se implementó la API con **FastAPI**, estructurada en routers (sesiones, productos, recomendaciones, CV y ASR). Se habilitó también un **canal de WebSockets** para la comunicación en tiempo real. * **Visión por computador**: se integró **Mediapipe** para análisis de prendas, colores y estimación de edad, además de endpoints de prueba para reconocimiento en tiempo real. * **Procesamiento de lenguaje natural (NLU)**: se incorporó un módulo heurístico que procesa la entrada de voz del cliente y permite identificar intenciones y entidades. * **Base de datos**: se diseñó y normalizó el modelo de datos, implementando las entidades principales (Producto, Variante, Categoría, Sesión) en PostgreSQL, asegurando integridad y trazabilidad. * **Frontend**: se configuró el entorno Flutter Web y se desarrolló un prototipo de interfaz conectada al backend. * **Pruebas**: endpoints funcionales verificados en Postman y Swagger UI. |
| Objetivos | Si bien los objetivos generales se han mantenido, se realizaron los siguientes ajustes:   * Se agregó el objetivo de integrar **WebSockets** para la interacción en tiempo real. * Se ajustó el alcance del frontend, priorizando la **interfaz web básica** en Flutter para la demostración en esta fase, dejando optimizaciones de diseño para fases posteriores. |
| Metodología | La metodología mantiene un enfoque ágil e incremental, con los siguientes ajustes:   * **Priorización del backend**: Se decidió asegurar un backend estable antes de avanzar en el frontend. * **Validación práctica**: Se utilizan **Swagger UI** y **Postman** para validar el correcto funcionamiento de los endpoints. * **Iteraciones cortas (sprints)**: Cada iteración se centra en agregar una funcionalidad completa y verificable (ej. módulo de sesiones, módulo de productos, etc.). * **Uso de control de versiones en GitHub** como repositorio central y herramienta de integración colaborativa. |
| Evidencias de avance | **Código fuente del backend en FastAPI**   * Demuestra la implementación de módulos funcionales (sesiones, productos, recomendaciones, CV y ASR) y la correcta modularización mediante routers. * Evidencia la aplicación de buenas prácticas en desarrollo de software, como separación de responsabilidades, manejo de dependencias y uso de entornos virtuales.   **Capturas de Swagger UI con endpoints funcionando**   * Permite visualizar la documentación generada automáticamente por FastAPI y comprobar que los endpoints están correctamente tipados y responden a las solicitudes. * Garantiza la transparencia y facilita la validación por parte del equipo docente.   **Pruebas en Postman**   * Respaldan la validación manual de operaciones CRUD y servicios de recomendación. * Confirman la interacción entre la base de datos y la API, asegurando la coherencia en el flujo de datos.   **Modelo de base de datos en PostgreSQL**   * Muestra el diseño relacional con entidades principales (Producto, Variante, Categoría, Sesión). * Justifica la calidad en el modelado de datos mediante la normalización y la definición de claves primarias y foráneas.   **Prototipo inicial de frontend en Flutter Web**   * Evidencia la integración con el backend, demostrando conectividad en tiempo real y visualización básica de información. * Justifica el avance en la interfaz de usuario, aun en estado inicial, como parte del cumplimiento de objetivos. |

**Evidencias recomendadas para adjuntar**

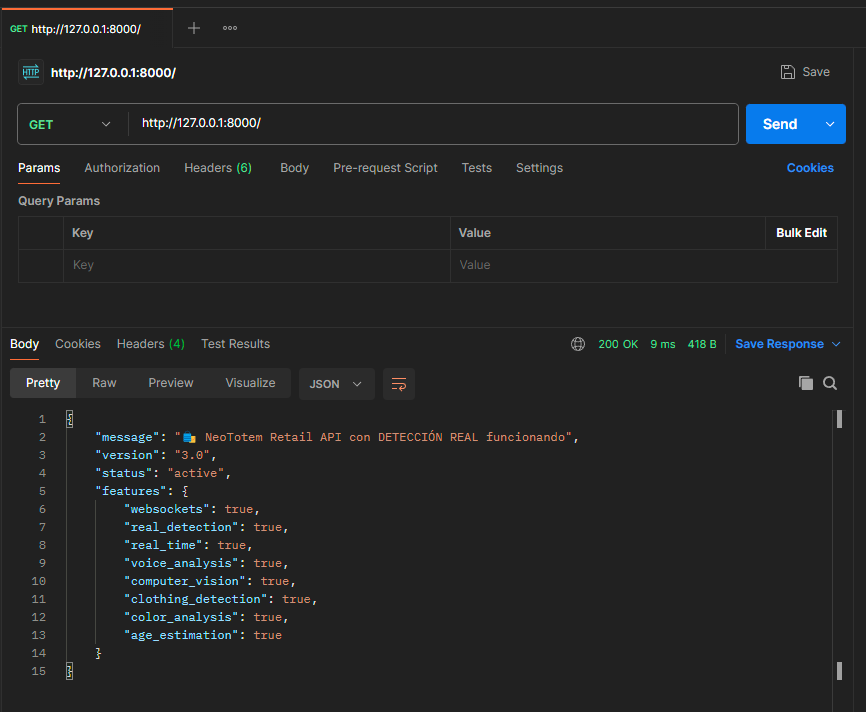
1. **Código fuente del backend (FastAPI)**

https://github.com/JuliMart/apt-totem-backend.git

1. **Capturas de Swagger UI**

****

1. **Evidencia en Postman**

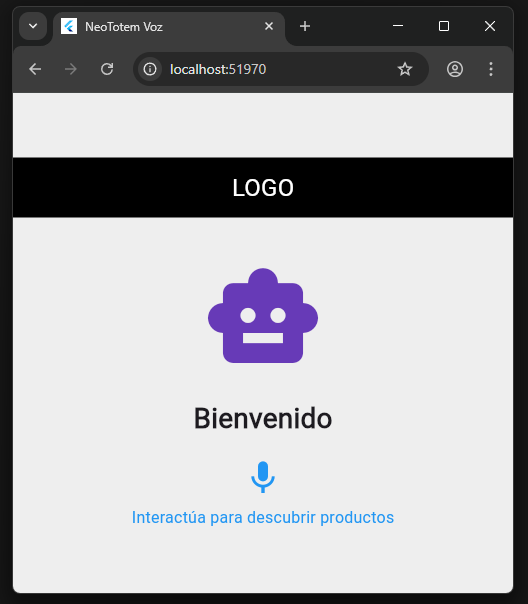
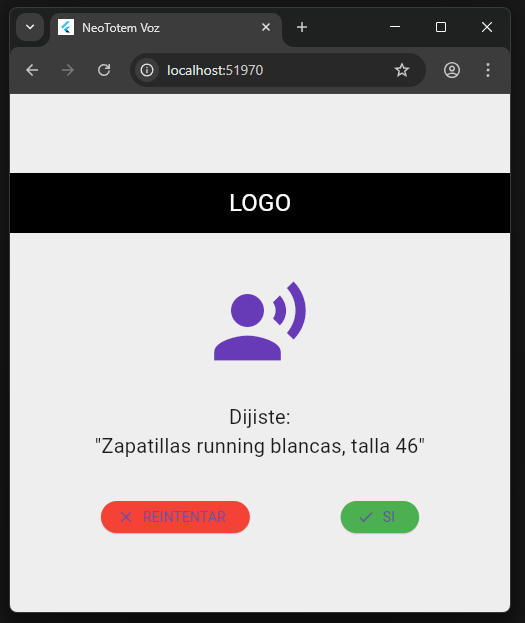
****

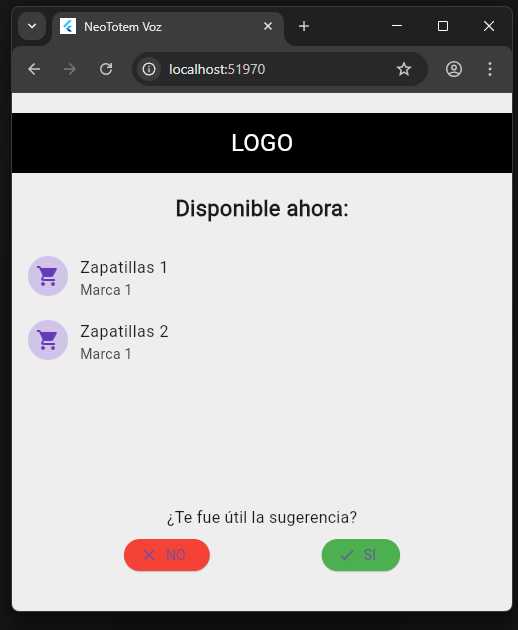
1. **Modelo de base de datos**

Organiza y relaciona entidades como productos, categorías, sesiones, consultas de voz, detecciones y recomendaciones.  
Incluye el diagrama MER y el script DDL con la definición de tablas, llaves y relaciones.

****

1. **Prototipo de frontend en Flutter Web**

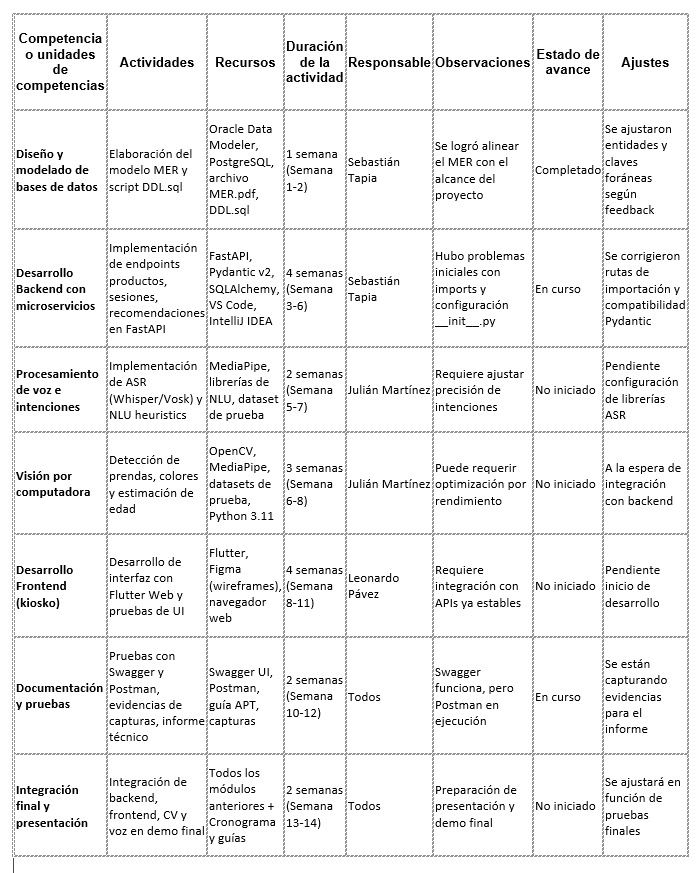
****

****

1. Ejecución del servidor Uvicorn

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Plan de Trabajo | | | | | | | |
| Competencia o unidades de competencias | Actividades | Recursos | Duración de la actividad | Responsable[[1]](#footnote-1) | Observaciones | Estado de avance | Ajustes |
| *Nombra las competencias o unidades de competencias que se relacionan con las diferentes actividades requeridas para el desarrollo del proyecto APT.* | *Nombra las actividades que se necesitan para desarrollar el proyecto APT.* | *Nombra los recursos necesarios para llevar a cabo las actividades definidas.* | *Señala la duración de cada actividad.* | *Escribe el nombre del integrante del equipo responsable de cada actividad.* | *Señala las dificultades o facilitadores que se podrían presentar durante la ejecución de cada una de las actividades propuestas.* | *Describe el estado de avance de cada actividad.*  *Tipos de estado:*  *En curso/ Con retraso/ No iniciado/ Completado/ Ajustada* | *Señala los ajustes o reformulaciones que has realizado.* |

|  |
| --- |
| **2. Monitoreo del Plan de Trabajo** |
| Examina cuidadosamente tu plan de trabajo, enfocándote especialmente en la columna de estado de avance y ajustes. |



|  |
| --- |
| **3. Ajustes a partir del monitoreo** |
| Profundiza en las observaciones de tu plan de trabajo. Analiza las actividades planificadas y señala qué aspectos facilitaron u obstaculizaron la ejecución del plan. Plantea cómo abordaste y/o abordarás los obstáculos. Por último, señala los ajustes que realizaste al plan de trabajo a partir de este análisis. |

|  |
| --- |
| Factores que han facilitado y/o dificultado el desarrollo de mi plan de trabajo:  **Factores que han facilitado:**   * La **definición temprana del alcance** del proyecto APT, lo que permitió organizar las fases (backend, CV/voz y frontend). * El uso de **herramientas de apoyo** como FastAPI, Swagger UI y Postman, que han facilitado la validación de endpoints y la documentación automática. * La **colaboración en equipo** mediante GitHub, que permitió mantener control de versiones y trabajo distribuido. * El **cronograma de fases**, que funcionó como guía de referencia para ir avanzando en paralelo con las guías APT.   **Factores que han dificultado:**   * Problemas técnicos relacionados con la **configuración de imports y archivos \_\_init\_\_.py**, que retrasaron la correcta carga de los routers en FastAPI. * La **compatibilidad con Pydantic v2**, que obligó a ajustar configuraciones en los modelos de respuesta. * Dificultades iniciales en la **conexión y pruebas de la base de datos**, que demandaron más tiempo del esperado en la etapa de backend. * El **inicio tardío de los módulos de CV y voz** (MediaPipe, OpenCV, NLU), que aún no han podido ser integrados de manera completa.   **Acciones para solucionarlas:**   * Se corrigieron los imports reestructurando las carpetas y dejando vacíos los \_\_init\_\_.py. * Se adaptaron los modelos a Pydantic v2 para evitar errores de validación. * Se decidió **probar la BD en SQLite** de manera local antes de migrar a PostgreSQL, reduciendo complejidad inicial. * Se acordó distribuir tareas: Sebastián (Backend/BD), Julián (CV y voz) y Leonardo (Frontend Flutter), con reuniones semanales para monitorear avances y evitar acumulación de retrasos. |

|  |
| --- |
| Actividades ajustadas o eliminadas:  Durante la ejecución del proyecto APT se realizaron **ajustes menores** en lugar de eliminaciones completas de actividades:   * **Ajuste en la validación de endpoints:** Inicialmente se pensaba probar todos los endpoints con datos reales desde la base de datos en PostgreSQL. Sin embargo, se decidió **iniciar las pruebas en SQLite local** para acelerar la validación, dejando la migración a PostgreSQL para fases posteriores. * **Ajuste en la integración de MediaPipe y NLU:** Se planificó realizar una integración temprana de los módulos de **visión por computadora y análisis de voz**, pero debido a la complejidad técnica se ajustó la planificación: primero se priorizó dejar el backend funcional y documentado en Swagger/Postman, posponiendo la integración completa de CV/voz para etapas posteriores. * **Ajuste en la metodología de pruebas:** En lugar de pruebas unitarias completas (difíciles de implementar en el tiempo disponible), se priorizó el uso de **Swagger UI y Postman** como evidencias principales de funcionamiento y validación de endpoints.   **Justificación de no eliminar actividades completas:** El proyecto mantuvo su alcance original (backend con endpoints, base de datos, integración futura con CV/voz y frontend). Los ajustes realizados permitieron **asegurar entregables medibles y verificables en los plazos establecidos**, garantizando el cumplimiento de la planificación. |

|  |
| --- |
| Actividades que no has iniciado o están retrasadas:   * **Integración de módulos avanzados de visión por computadora (MediaPipe/OpenCV) y análisis de voz (NLU):** Estas actividades se encuentran **en retraso** porque el equipo priorizó primero la implementación y validación del backend (FastAPI, endpoints CRUD, base de datos) para asegurar un producto funcional mínimo. **Estrategia:** se avanzará con pruebas unitarias simples y la integración modular progresiva, dejando configurados los endpoints y simulando resultados hasta que se disponga del modelo entrenado e integrado. * **Pruebas completas de usabilidad y demo final (fase frontend con Flutter):** Esta parte aún **no se ha iniciado**, ya que depende directamente de la estabilidad del backend y de la disponibilidad de los datos en la base de datos. **Estrategia:** utilizar un enfoque incremental, comenzando con un prototipo sencillo en Flutter conectado a los endpoints ya probados en Postman, para luego extender a funcionalidades interactivas (detección de prendas, recomendaciones, etc.). * **Migración total a PostgreSQL en entorno productivo:** Actualmente se trabaja con SQLite como base para pruebas rápidas. La migración a PostgreSQL **está pendiente** debido a la complejidad en la configuración y tiempos de entrega intermedia. **Estrategia:** se aplicará una migración progresiva con el script DDL ya preparado, de modo que la transición no afecte el funcionamiento del backend. |

1. En caso de que el Proyecto APT sea grupal, en esta columna deben indicar el nombre de los responsables de cada tarea o actividad. Esto posteriormente permitirá diferenciar la evaluación por cada integrante. [↑](#footnote-ref-1)