Memoria del Proyecto

1 Datos esenciales

1.1 Título del proyecto

OpenTrackBank (OTB) por Taulyd : Gestor de cuentas bancarias de codigo abierto

1.2 Alumno, grupo y centro

- Alumno: Santiago Sánchez Delgado
- Grupo: B
- Centro: I.E.S Albarregas

1.3 Módulo y curso académico

- Código del modulo: 0492
- Módulo: DAM
- Curso Académico: 2025-2026

1.4 Docente tutor

Mercedes

1.5 Fecha y versión del documento

V0.0.0.0

1.6 Enlaces (QR/URL): Repositorio, Tablero, Figma, Documentación

- Repositorio Github
- Tablero
- Figma
- Documentacion

1.7 Licencia y avisos legales

- Licencia Apache 2.0

2. Resumen ejecutivo

2.1 Problema/oportunidad

Las persona comunmente suelen tener contratado una o varias cuentas bancarias, o tenerlas asociadas por algun familiar. En el contexto actual, las pequeñas y medianas empresas (PYMEs) enfrentan dificultades para gestionar sus recursos de forma eficiente, especialmente en tareas como el gestion de saldos, costes, u cuentas bancarias. Muchas de ellas aún usan hojas de cálculo o sistemas manuales, lo que genera errores, falta de escalabilidad y pérdida de tiempo.

2.2 Propuesta de solución

Una aplicacion multiplataforma que permita la conexion y gestion de cuentas bancarias personales o familiares. Destinado a toda aquellas personas con cuentas bancarias personales, entrando a mayores de edad segun el país (18 en españa); y familiares, como padres con cuentas bancarias de hijos y por hijos o allegados que gestionen cuentas con personas con dificultades para manejar dispositivos digitales o tecnologías. Proponiendo una aplicación de gestión de recursos , desarrollada en Java, que permita a las PYMEs centralizar la gestión de saldos, costes, u cuentas bancarias, con una interfaz intuitiva y funcionalidades adaptadas a su tamaño y necesidades.

2.3 Objetivo de producto y objetivo de E1

- Objetivo/valor del producto: Comodidad simplificando el manejo de la cuenta/s. Crear una plataforma sencilla, escalable y segura para gestionar los recursos en PYMEs.
- Objetivo de E1: Diseño, prototipado, prototipo y demo de la aplicación. Entregar una demo funcional que permita registrar cuentas, asignar etiquetas, gestionar presupuestos y generar informes básicos en CSV, DOCX y EXCEL, con interfaz de usuario intuitiva y estructura de base de datos.

2.4 Alcance de la demo de E1 (qué se muestra y qué no)

Se mostrara(Incluye):

- Una interfaz de inicio de sesión
- Una vista de las cuentas
- Menú de ajustes

No se mostrara(Excluye):

- La gestion de fichero JSON que simulara la API
- El guardado de hashes en bases de datos no relacional, relacional o fichero
- Pantalla de transacciones

3. Justificación

3.1 Beneficios esperados (técnicos, educativos y de uso)

Es un proyecto noble que busca facilitar una accion cotidiana generando un nuevo estandar e indepencia bancaria y que abarcando a un gran publico aporte un gran beneficio para nuestra sociedad.

La falta de herramientas accesibles y personalizadas para el seguimiento de gastos y presupuestos en entornos domésticos y pequeñas empresas ha generado una brecha significativa en la gestión financiera cotidiana. Muchas personas, especialmente en contextos de baja renta o con necesidades específicas (como estudiantes, jubilados o familias en situación de vulnerabilidad), no cuentan con sistemas intuitivos, fáciles de usar y adaptados a sus necesidades reales.

Este proyecto nace como respuesta a esa necesidad, ofreciendo una aplicación multiplataforma que permite a los usuarios registrar gastos, establecer presupuestos, recibir alertas automáticas y generar informes visuales en tiempo real. La solución se enfoca en la simplicidad, la accesibilidad y la personalización, con un diseño centrado en el usuario y funcionalidades que promueven la conciencia financiera y el control del gasto.

Además, al incorporar funcionalidades como la detección de patrones de gasto, la recomendación de ajustes y la integración con servicios de pago, la aplicación no solo ayuda a gestionar el dinero, sino que también fomenta hábitos financieros saludables y sostenibles, contribuyendo así a la reducción de desigualdades (ODS 10) y al fomento de la economía local (ODS 8).

3.2 Integración con la industria extremeña (sectores, clústeres, casos de uso)

Pequeñas pymes que necesiten gestionar gastos e ingresos mensuales y anuales

3.3 Análisis de productos similares (benchmark breve: 3–5 referencias)

1. BankTrack
2. Holded
3. Agicap
4. Iziago

Producto	Qué hace	Puntos fuertes	A considerar
Banktrack	Plataforma SaaS española para PYMEs (y holdings) que conecta cuentas bancarias, gestiona facturas con IA, crea previsiones de tesorería en tiempo real. (banktrack.com)	Muy adaptada al contexto español (cumplimiento facturación, Verifactu, etc) ; buena visibilidad de caja; integración multibanco. (banktrack.com)	Puede requerir cierto “on-boarding” para activarse bien; si tu empresa trabaja mucho fuera de España o con monedas múltiples quizá haya que ver bien alcance internacional.
Holded	Software de gestión para PYMEs que incluye módulo de tesorería: conecta bancos, gestiona facturas, pagos/cobros, previsión de liquidez. (holded.com)	Muy buena opción para empresas que quieren una plataforma “todo en uno” (facturación + tesorería + contabilidad) ; interfaz adaptada a usuarios no financieros.	Siendo más generalista quizá las funcionalidades de tesorería estén menos profundas que un software especializado; ver que la parte bancaria/balance esté bien.
Agicap	Plataforma de tesorería para PYMEs (“cash-flow management”): seguimiento de caja	Muy enfocada en la tesorería pura, lo que la hace buena para empresas que tienen ese foco (liquidez,	Quizás menor grado de integración con otros módulos de empresa (facturación, contabilidad) que

Producto	Qué hace	Puntos fuertes	A considerar
	real, previsión, automatización bancaria. (agicap.com)	previsión) ; interfaz ágil.	plataformas más amplias; revisar si cumple normas españolas específicas.
Cegid Iziago (módulo tesorería)	Solución para PYMEs que incluye módulo de previsión de tesorería, recuperación automática de extractos bancarios, pagos. (Iziago)	Buena opción si ya tienes otras piezas del ecosistema “Cegid” o quieres software más “maduro”; permite gestionar pagos, adeudos, etc.	Posiblemente más rigidez, costes más altos, quizá más pensado para empresas maduras; interfaz y agilidad pueden ser menores que soluciones “nativas fintech”.

3.4 Participación en ODS (mapa ODS ↔ funcionalidad/impacto)

- ODS-8 -> La aplicación fomenta el autoempleo, la gestión financiera y el control de gastos, lo que impulsa la independencia económica de los usuarios.
- ODS-10 -> Accesibilidad universal, diseño inclusivo, y herramientas para grupos vulnerables (estudiantes, jubilados, familias en situación de riesgo).

4. Historias de usuario

4.1 Convenciones de numeración y formato (p. ej., HU-001, HU-002…)

(Formato: HU-001, HU-002… | Given/When/Then)

4.2 Ejemplos de historias (3–5) con Criterios de Aceptación (Given/When/Then)

HU-001: Modificación de tema visual del hub
Given que el usuario está en el menú de ajustes,
When selecciona “Tema” y elige “Modo oscuro”,
Then debe aplicarse inmediatamente el cambio en toda la interfaz.

Given que el usuario vuelve a “Tema” y elige “Modo claro”,
When guarda los cambios,
Then debe mostrarse un mensaje confirmado de actualización del tema.

Given que el usuario selecciona un tema no disponible,
When intenta aplicarlo,
Then debe mostrarse un error indicando que el tema elegido no es válido.

HU-002: Verificación de estado de conexión con el banco (API)
Given que el usuario está en el menú de ajustes,
When selecciona “Estado de la conexión”,
Then debe mostrar un mensaje indicando si la API está conectada o desconectada.

Given que el usuario intenta realizar una transacción mientras la API está desconectada,
When ingresa a la pantalla de transacciones,
Then debe bloquear las operaciones y mostrar un aviso de conexión perdida.

Given que el usuario recibe una notificación de error en la conexión,
When selecciona “Reintentar”,
Then debe intentar reconectar con la API automáticamente.

HU-003: Visualización del presupuesto mensual con gráficos
Given que el usuario ha ingresado sus gastos y recibos,
When accede a la pantalla de presupuesto,
Then debe mostrarse un gráfico resumen (ej.: barras o torta) de los gastos por categoría.

Given que el usuario filtra el presupuesto por mes,
When selecciona “Abril 2025”,
Then debe mostrar solo los datos correspondientes a ese período en el gráfico.

Given que el usuario no ha ingresado ningún dato de gasto,
When accede a la pantalla de presupuesto,
Then debe mostrarse un mensaje indicando que aún no hay información para generar el gráfico.

4.3 Backlog priorizado (MoSCoW/Kano o similar)

- Must/Should/Could → Imprescindible/Importante/Conveniente

HU-001: Modificación de tema visual del hub -> Conveniente **HU-002: Verificación de estado de conexión con el banco (API)** -> Imprescindible **HU-003: Visualización del presupuesto mensual con gráficos** -> Importante

4.4 Trazabilidad HU ↔ CA ↔ Pruebas ↔ Figma

5. Arquitectura

5.1. Diagrama (C1/C2: contexto y contenedores)

5.2 Decisiones de arquitectura (ADR) y numeración (p. ej., ADR-001, ADR-002)

1. ADR-001: Selección del protocolo de autenticación
    - Contexto: Necesidad de seguridad en la gestión de cuentas bancarias.
    - Decisión: Implementar OAuth 2.0 con PKCE para autenticación segura.
    - Alternativas descartadas:
        - Autenticación por contraseña tradicional (riesgo de phishing).
        - SMS-based 2FA (dependencia de redes móviles y latencia).
    - Consecuencias: Mejora en la seguridad, pero requerimiento de integración con proveedores de OAuth (ej.: Google, Apple).
2. ADR-002: Selección de API para conexión bancaria
    - Contexto: Necesidad de conectar con múltiples entidades bancarias.
    - Decisión: Usar una API estándar RESTful con soporte para Open Banking (PSD2).

- Alternativas descartadas:
  - Integración directa por FTP/SFTP (menos escalable y seguro).
  - Uso de Webhooks personalizados (complejidad en gestión de eventos).
- Consecuencias: Mayor compatibilidad con bancos europeos, pero dependencia de cumplimiento normativo.

5.3. Integraciones, datos y dependencias

- Integraciones externas
  1. API de banca abierta (Open Banking):
     - Conexión a entidades bancarias mediante protocolos estándar como PSD2 (Reglamento Europeo).
     - Ejemplo: API RESTful para obtener saldos, transacciones y estados de cuenta.
  2. Servicios de autenticación:
     - OAuth 2.0 con PKCE para integración con proveedores como Google, Apple o Microsoft.
     - Soporte para passkey/autenticación biométrica (FIDO2).
  3. Exportación de datos a formatos estándar:
     - Librerías para generar archivos PDF, Excel (.xlsx), CSV y DOCX.
  4. Servicios de almacenamiento en la nube (opcional):
     - Integración con AWS S3, Azure Blob Storage o Google Cloud Storage para respaldos de datos críticos.
  5. Dependencias de terceros:
     - Frameworks de UI.
     - Servicios de análisis de transacciones.
- Datos y dependencias internas
  1. Almacenamiento local:
     - Base de datos relacional (ej.: PostgreSQL) para usuarios, configuraciones (tema, idioma), y metadatos de cuentas o local mediante SQLite para dispositivos móviles.
     - Uso de JSON simulado en la demo (como se menciona en el alcance de E1).
  2. Datos de transacciones:
     - Recuperados desde APIs bancarias y almacenados temporalmente para visualización y análisis local.
  3. Dependencias tecnológicas clave:
     - Lenguajes: Java.
     - Herramientas: Figma (diseño), GitHub/GitLab (control de versiones), Docker (contenedores).

5.4. Riesgos técnicos y mitigación

Riesgo 1: Cambios en las APIs bancarias o normativas legales

- Impacto: Incompatibilidad con el sistema al actualizar protocolos de Open Banking.
- Mitigación:
    - Usar versiones compatibles de las APIs y mantener actualizaciones regulares.
    - Implementar mecanismos de fallback (ej.: notificación al usuario si una API falla).

Riesgo 2: Seguridad en la gestión de datos sensibles

- Impacto: Exposición de credenciales o transacciones por fallos de autenticación/criptografía.
- Mitigación:
    - Encriptar datos en tránsito (TLS 1.3) y en reposo (AES-256).
    - Validar estrictamente las autorizaciones (RBAC) y realizar auditorías de seguridad periódicas.

Riesgo 3: Rendimiento bajo carga alta

- Impacto: Latencia excesiva al cargar transacciones o generar reportes.
- Mitigación:
    - Implementar caché en memoria (Redis) para datos frecuentes (ej.: saldos).
    - Usar escalabilidad horizontal en el backend (microservicios + Kubernetes).

Riesgo 4: Dependencia de terceros obsoletas

- Impacto: Fallo en librerías críticas (ej.: OAuth, exportación a Excel).
- Mitigación:
    - Monitorear actualizaciones de dependencias (ej.: Dependabot en GitHub).
    - Mantener código compatible con múltiples versiones de bibliotecas.

Riesgo 5: Diferencias de rendimiento entre JavaFX (escritorio) y Android SDK (móvil).

- Impacto: La app no funciona enficienteme en las versiones de escritorio y móvil.
- Mitigación:
    - Optimización del código para cada plataforma y pruebas en dispositivos reales.

Riesgo 6: Complejidad al compartir lógica de negocio entre desktop y móvil debido a diferencias en APIs nativas.

- Impacto: Lógica compleja o spaguetti.
- Mitigación:
    - Modularización del código con capas compartidas (ej.: JavaBeans) y adaptadores específicos por plataforma.

Riesgo 7: Vulnerabilidades de seguridad en bibliotecas Java.

- Impacto: Datos sensibles expuestos a atacantes.
- Mitigación:

- Actualizaciones periódicas de dependencias y uso de herramientas como OWASP Dependency-Check para auditorías.

6. Requisitos no funcionales (NFR)

6.1 Convención de numeración (p. ej., NFR-001, NFR-002…)

NFR-00X

6.2 Rendimiento y capacidad (latencia, throughput) (NFR-001)

- Métrica: Tiempo máximo de respuesta para operaciones críticas (ej.: inicio de sesión, carga de transacciones).
- Umbral: ≤ 2 segundos en el 95% de los casos.
- Método de verificación: Pruebas de carga con herramientas como JMeter o LoadRunner, simulando 100 usuarios concurrentes.
- Vinculado a: HU-001 (Inicio de sesión), ADR-002 (Selección de API para conexión bancaria).

6.3 Seguridad (authZ/authN, cifrado, logs) (NFR-002)

- Verificación: Pruebas de autenticación OAuth2 en entorno de prueba.
- Ubral: 100% de éxito en pruebas de acceso autorizado.
- Método de verificación: Uso de herramientas estándar de Java (javax.security) y logging con SLF4J.
- Vinculado a: HU-002 (Gestión de cuentas), ADR-001 (Implementación de OAuth 2.0).

6.4 Accesibilidad y UX (pautas mínimas) (NFR-003)

- Verificación: Pruebas de contrastes cromáticos y navegación táctil en dispositivos móviles (Java ME).
- Umbral: 95% de cumplimiento con WCAG 2.1.
- Método de verificación: Herramientas de validación como AXE para JavaFX y Android Accessibility Suite.

6.5 Calidad del código y mantenibilidad (estándares, lint, pruebas) (NFR-004)

- Métrica: Cobertura de pruebas unitarias y de integración.
- Umbral: ≥ 80% cobertura de código crítico.
- Método de verificación: Informes de SonarQube o similar, revisados por el equipo técnico.

6.6 Observabilidad (logs, métricas) (NFR-004)

- Métrica: Tiempo máximo para detectar errores críticos en logs.
- Umbral: ≤ 5 minutos desde la ocurrencia del error.
- Método de verificación: Monitoreo en tiempo real con herramientas como Prometheus/Grafana.
- Vinculado a: ADR-002 (Selección de API para conexión bancaria).

6.7 Verificación de NFR (métricas y cómo medirlas)

Declaradas en cada NFR

7. Prototipo Figma y Anexos

7.1 Prototipo navegable (lista de pantallas/flows y estados)

7.2 Guía de diseño (tokens: color, tipografía, espaciado; componentes y variantes)

8. Anexos

8.3.1 Plan de pruebas, trazabilidad y evidencias (capturas, vídeos, enlaces)

- Historia de usario objetivo:

  HU-015

- Criterios de aceptación:

  Recorrer hasta el menu y recibir el feedback a partir de la señal de la API y su respuesta.

- Videos/Imagenes:

  - Videos
  - Imagenes

8.3.2 KPIs iniciales (aceptación HU, defectos, lead time)

8.3.3 Enlaces y referencia de artefactos (repo, tablero, ADR, NFR)

8.3.4 Changelog de E1 (tabla de cambios relevantes)