REPORTE FINAL

API WEB – BANK MANAGEMENT

Proyecto Final III BIM

H.E - Software Development

Centro Educativo Técnico Laboral KINAL

🏫 **IN6BM – 2025**

🪪 **Anthony Josue Escobar Ponce - 2020229**

🪪 **Aníbal Guillermo Herrera Ortiz – 2020324**

Sistema de gestión bancaria

“El objetivo principal es crear y desarrollar una aplicación web de sistema bancario moderno, funcional y seguro. Lograr que permita la administración de usuarios y cuentas bancarias a través de un panel administrativo y de trabajador, además de una interfaz de cliente. Este sistema debe cumplir con el hecho de realizar gestiones seguras y dinámicas de usuarios, cuentas, movimientos financieros, y servicios exclusivos. contemplando autenticación por tokens, gestión de movimientos y productos, y soporte para divisas externas.”

PLANIFICACIÓN

SCRUM TEAM:

Ambos integrantes formaremos parte del equipo de desarrollo, colaborando activamente en la construcción de la aplicación.

ROLES PRINCIPALES:

Anthony Escobar:

Product Owner, guiando la visión del producto y priorizando el backlog.

Scrum Master, quien facilitará las ceremonias SCRUM y ayudará a resolver obstáculos.

Aníbal Herrera:

Desarrollador Principal, Desarrollador principal de Frontend de nuestra API web y developer encargado de estructuración final.

SPRINTS:

SEMANA 1

- **Definición de tecnologías base (stack MERN: MongoDB, Express, React, Node.js).**
- **Investigación de APIs bancarias de referencia para definir funcionalidades clave.**
- **Diseño de la estructura base del proyecto (carpetas /client para React y /server para Node.js).**
- **Definición de vistas principales: Login, Registro, Dashboard de cliente y Panel de administrador.**
- **Establecimiento de estilos con TailwindCSS (reemplazo de Bootstrap) y arquitectura inicial.**
- **Configuración de dependencias críticas: helmet, morgan, dotenv, argon2, mongoose, express-validator.**

SEMANA 2

Implementación del backend: Creación de modelos con Mongoose:

- **User: Validación de DPI único, ingresos mínimos (Q100), roles (CLIENT, EMPLOYEE, ADMIN).**

- **Account: Generación automática de número de cuenta y saldo.**
- **Movement: Lógica para depósitos, transferencias y reversión (1 minuto).**
- **Product: Productos bancarios con condiciones personalizadas.**
- **Controladores, middlewares (checkDailyLimit), validadores y rutas protegidas por roles.**
- **Autenticación JWT y middleware de autorización (hasRole).**
- **Documentación técnica con Swagger para endpoints principales (/auth, /account, /movement).**

SEMANA 3

Implementación de lógica de negocio:

Transferencias: Límite de Q2,000 por transacción y Q10,000 diarios.

Validación de saldo suficiente y cuentas existentes.

Módulo de favoritos (alias para cuentas frecuentes).

Integración de API de divisas (ExchangeRate-API de IBM) con caché en MongoDB.

Endpoint adicional /currency/history para consultar conversiones recientes.

Pruebas en Postman: Validación de reglas por roles y transacciones concurrentes (uso de transacciones ACID).

SEMANA 4

Conexión frontend-backend con Axios:

1. **Interceptors para manejo automático de tokens JWT.**
2. **Persistencia de sesión con localStorage y AuthContext.**

Desarrollo de vistas:

Panel de cliente:

1. **Visualización de saldo (con conversión a USD/EUR).**
2. **Historial de movimientos y sección de "Favoritos".**

Panel de administrador:

1. **Filtros para ordenar usuarios por movimientos (asc/desc).**
2. **Modal para revertir depósitos (ventana de 1 minuto).**

SEMANA 5-6

Refinamiento de frontend:

1. **Componentes reutilizables para formularios (transferencias, depósitos).**
2. **Validación de formularios con feedback visual.**

Pruebas finales:

1. **Postman: Cobertura del 100% de endpoints (incluyendo edge cases).**
2. **Frontend: Depuración de errores en consola y validación de rutas protegidas.**

Despliegue:

1. **Backend en Vercel: Configuración de vercel.json para redireccionamientos y CORS.**
2. **Frontend en Firebase: Reglas de hosting para SPAs (rewrite en firebase.json).**

Revisión y entrega de avances cada 4 dias para corrección de errores y validación de requerimientos a cumplir a continuación de manera previa a cada SPRINT.

REPORTE DETALLADO POR SEMANA

SEMANA 1

- Selección del stack MERN. Investigación de plataformas bancarias para funcionalidades clave.
- Diseño visual con TailwindCSS (colores oscuros, detalles en vino y blanco).
- Configuración inicial de repositorio GitHub y dependencias.

SEMANA 2

- Modelos implementados: User, Account, Movement, Product.
- Autenticación JWT y middleware de roles. Documentación Swagger para endpoints críticos.

SEMANA 3

- Reglas de negocio: Límites de transferencia, reversión de depósitos.
- Módulo de favoritos y API de divisas con caché.

SEMANA 4-5

- Frontend:
    - AuthContext para gestión global de sesión (retraso de 2 días por rediseño).
    - Dashboards funcionales para cliente y administrador.

SEMANA 6-7

- Despliegue:

- Problemas con CORS en Vercel y reglas de Firebase para SPAs. Soluciones implementadas.

CAMBIOS VS. PLANIFICACIÓN

- **Frontend:** Reemplazo de Bootstrap por TailwindCSS.
- **Backend:** Eliminación del modelo Bill (generación de PDF en Movement).
- **API de divisas:** Migración de Google a ExchangeRate-API (IBM) por limitaciones.
- **Demoras:**
  - 3 días en conexión frontend-backend (AuthContext).
  - 1.5 días en lógica de transacciones y movimientos con historial por cuentas

RESULTADOS FINALES

- **Backend:** 100% de endpoints probados. Validación de esquemas reduce errores 400 en 90%.
- **Frontend:** 90% libre de errores con interfaz didáctica e intuitiva de usuario con base a otros sistemas bancarios (como BI en línea).