Documento de Decisiones Técnicas

Descripción

Esta aplicación web permite a los operadores bancarios gestionar cuentas de ahorro para clientes existentes, realizar transacciones (depósitos y retiros), consultar saldos y visualizar el historial de transacciones.

Tecnologías Utilizadas

- Frontend: Blazor Web Server App && Blazor Bootstrap
- Backend: .NET 8 Web API (o superior)
- Base de datos: SQL Server

Justificación:

Decidí utilizar las tecnologías ya mencionadas porque, aunque eran tecnologías nuevas para mí, quería demostrar mi capacidad para aprender y adaptarme rápidamente a diferentes entornos. Elegí Blazor porque me permitió desarrollar una interfaz dinámica y moderna con C# en lugar de JavaScript, lo que hizo más eficiente el desarrollo al compartir lógica entre el frontend y el backend. Por otro lado, .NET 8 me brindó un alto rendimiento para la API, mientras que SQL Server aseguró un manejo robusto de los datos.

Características Implementadas

Frontend (Blazor Web Server App)

- Componente Razor para cada funcionalidad
- Formularios con validaciónes utilizando Toast Sevices.
- Interacción con API: Servicio HttpClient, Interfaz IRepository<T> para operaciones CRUD y Serialización/Deserialización JSON automática
- Implementación de Blazor Bootstrap.
- Manejo de errores en la interfaz de usuario.

Backend (Web API)

- Controladores RESTful siguiendo convenciones de .NET
- Entity Framework Core para acceso a datos.
- Manejo centralizado de excepciones.
- DTO para no dañar la integridad de los modelos de datos.

Patrones de Diseño Implementados

Arquitectura Base

- **MVC**: En backend para estructura API RESTful
- **Component-Based**: Arquitectura en frontend con Blazor

Justificacion

En el backend, seguí el patrón MVC para estructurar la API RESTful, separando responsabilidades (controladores para rutas, modelos para datos y servicios para lógica), lo que mejoró la mantenibilidad y el orden del proyecto.

Para el frontend, aproveché la arquitectura basada en componentes de Blazor, que me permitió reutilizar código, mantener una interfaz consistente y desarrollar de manera más ágil mediante la encapsulación de funcionalidades en piezas independientes.

Diagrama ER