Tema

ESPECIFICACIÓN DE REQUERIMIENTOS DE SISTEMA DE EUREKABANK

Autores

BARRIONUEVO ORDÓÑEZ LINDSAY DOMENIQUE

RIVERA LÓPEZ JOEL ALESSANDRO

YARANGA SUQUILLO LEONARDO JAVIER

Tutor

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software PhD. (c) Seguridad Información

Fecha

10/12/2025

ESPECIFICACIÓN DE REQUERIMIENTOS DEL PROYECTO DE SISTEMA DE EUREKABANK

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

CASO DE ESTUDIO

Se desea desarrollar una aplicación para facilitar y comprender el proceso de una entidad financiera, sus movimientos y cómo trabaja. El "Eureka Bank" es una entidad financiera que requiere modernizar su plataforma tecnológica para centralizar y asegurar sus operaciones transaccionales. Actualmente, el banco maneja información crítica sobre Clientes, Cuentas de Ahorro, Movimientos Monetarios y Empleados, pero necesita una capa de servicios robusta que permita la interoperabilidad entre sus diferentes canales de atención como pueden ser cajeros, Ventanilla, Banca Web.

El funcionamiento operativo del banco que debe automatizar el sistema es el siguiente:

1. Gestión de Seguridad: El acceso a las operaciones sensibles está restringido. Todo empleado (Cajero, Gerente) debe autenticarse contra una base de datos centralizada para obtener permisos, dirigidos por el superadmin Monster.

2. Operatividad de Cuentas: Los clientes poseen cuentas bancarias asociadas a una sucursal específica. El sistema debe ser capaz de identificar no solo el saldo, sino también los datos del titular y el estado de la cuenta.

3. Transaccionalidad (Depósitos y Retiros):

- Cuando un cliente realiza un depósito, el sistema debe identificar la cuenta, registrar la entrada de dinero y actualizar el saldo acumulado de forma atómica.

- Cuando se solicita un retiro, el sistema debe validar estrictamente que el saldo disponible sea mayor o igual al monto solicitado. Si la validación es exitosa, se debita el monto y se genera un número de operación único.

4. Auditoría e Historial: Cada operación (movimiento) queda registrada con fecha, hora, empleado responsable y tipo de movimiento. Los clientes pueden solicitar un extracto (historial) para visualizar sus últimos movimientos ordenados cronológicamente.

El sistema será desarrollado con Microsoft Visual Studio 2022 como IDE, y SQL Server como base de datos relacional. Adicionalmente, el proyecto técnico implica desarrollar esta lógica en una arquitectura orientada a servicios utilizando API RESTful con el framework .NET (ASP.NET Core), desplegando la base de datos en una instancia de Cloud SQL para SQL Server en la nube de Google Cloud Platform para garantizar alta disponibilidad.

1. Introducción

Este documento de Especificación de Requerimientos de Software, ERS, ha sido elaborado siguiendo los lineamientos de la norma IEEE 830, adaptada para proyectos de integración de servicios. Constituye la documentación técnica fundamental previa al desarrollo e implementación de la capa de servicios del sistema Eureka Bank. El documento detalla la arquitectura, las interfaces y los comportamientos funcionales necesarios para construir una solución bancaria basada en el protocolo SOAP.

2. Propósito

El propósito principal de este documento es definir, listar y detallar exhaustivamente los requerimientos funcionales y no funcionales del sistema "Eureka Bank Service Layer". Este documento sirve como la fuente de verdad única para los desarrolladores Java, los arquitectos de base de datos y los gestores del proyecto. Su objetivo es eliminar la ambigüedad en la lógica de negocio bancaria, estableciendo claramente cómo deben comportarse los servicios web ante transacciones financieras críticas, asegurando que todas las partes interesadas, Stakeholders, tengan una visión unificada del producto final a desplegarse en Google Cloud. Este documento servirá como guía maestra para:

- Los desarrolladores del servicio SOAP en Java (Backend).
- Los desarrolladores de las interfaces de usuario en sus respectivas tecnologías (Java Swing, Web/HTML5, Consola, Android).
- Los arquitectos de infraestructura encargados del despliegue en Google Cloud.
- El equipo de QA para validar la correcta integración entre los clientes y el servicio.

Es principalmente preparado para establecer el escenario para la fase de diseño del proyecto. El entregable que está siendo elaborado es la primera versión del documento de visión para el proyecto enfocado en el sistema de “Eureka Bank”.

3. Alcance

El presente trabajo se centra en la construcción del sistema Transaccional de Eureka Bank, tomando en cuenta su desarrollo Backend y Frontend, es decir, su lógica Full Stack. El sistema automatizará la comunicación entre las interfaces de usuario (Frontend) y los datos almacenados en la nube.

La implementación del sistema, una vez finalizada tendrá como beneficios la gestión de los siguientes módulos:

1. CAPA DE SERVICIOS (Backend)
   1.1. Desarrollo de **Web API REST (ASP.NET Core)** que exponen la lógica de negocio mediante endpoints HTTP y formato JSON (Login, Cuentas, Movimientos).
2. CAPA DE DATOS (Cloud)
   2.1. Implementación y gestión de la base de datos SQL Server en una instancia de Google Cloud SQL.

3. CAPA DE CLIENTE
   3.1. Cliente Consola: Interfaz de línea de comandos (CLI) desarrollada en .NET para ejecución rápida.
   3.2. Cliente Escritorio: Aplicación .NET (Windows Forms / WPF) para uso intensivo por parte de los cajeros del banco.
   3.3. Cliente Web: Portal accesible vía navegador consumiendo la API REST.
   3.4. Cliente Móvil: App Android que consume los servicios JSON para consultar saldos.

El sistema permitirá realizar las siguientes funciones críticas en todas las plataformas: Autenticación de usuarios, Consulta de saldos y movimientos, Reportes de sucursal (Web/Escritorio) y Ejecución de transacciones monetarias (Escritorio/Consola).

4. Definiciones, Acrónimos y Abreviaturas

4.1 Definiciones

Tabla 1. Definiciones

TÉRMINO	DEFINICIÓN
Backend	Capa de acceso a datos y lógica de negocio que se ejecuta en el servidor. En este proyecto, corresponde a la Web API REST alojada en IIS (Internet Information Services) o Kestrel.
Cliente Pesado (Escritorio)	Aplicación desarrollada en .NET (C#) que se instala localmente en las estaciones de trabajo. Consume recursos del PC y se conecta a la API para operaciones críticas.
Cliente Ligero (Web)	Aplicación nativa (Android APK) diseñada para ejecutarse en smartphones, optimizada para pantallas táctiles y redes de datos móviles (4G/5G).
Cliente Móvil	Detalle de cuotas mensuales con capital, intereses y saldo.
Cliente Consola (CLI)	Interfaz de Línea de Comandos. Aplicación minimalista basada en texto, utilizada por administradores para pruebas de conectividad y ejecución de operaciones sin carga gráfica.
Deploy (Despliegue)	Proceso de trasladar el código compilado (archivo .war) al servidor de aplicaciones y la estructura de base de datos a la nube para que el sistema esté operativo.

Endpoint	Punto de acceso específico (URL) donde el servicio web SOAP escucha y atiende las peticiones. Ejemplo: `http://ip-servidor:8080/EurekaBank/EurekaService`.
Google Cloud SQL	Servicio de base de datos administrado por Google. Para este proyecto, se utilizará la versión compatible con Microsoft SQL Server.
JDBC (Java Database Connectivity)	API estándar de Java que permite a las aplicaciones conectarse a bases de datos relacionales, enviar consultas SQL y procesar resultados.
ADO.NET / Entity Framework	Tecnologías de acceso a datos en .NET que permiten a las aplicaciones conectarse a bases de datos SQL Server, mapear objetos y ejecutar consultas. (Reemplaza a JDBC).
ASP.NET Core	Framework de código abierto multiplataforma para crear aplicaciones modernas conectadas a Internet, como aplicaciones web y APIs REST. (Reemplaza a JAX-WS).
Latencia	Tiempo que tarda un paquete de datos en viajar desde el cliente (ej. Celular) hasta el servidor (Cloud) y regresar. Crítico para la experiencia de usuario.
Serialization (Serialización)	Proceso de conversión de objetos C# en formato JSON para su envío por la red, y viceversa (Deserialization) al recibir una respuesta. (Reemplaza a Marshalling).
Middleware	Software que actúa como puente. En .NET, se refiere también a la tubería (pipeline) de procesamiento de peticiones HTTP en el servidor API.
Pool de Conexiones	Mecanismo que mantiene un caché de conexiones a la base de datos abiertas y listas para ser reutilizadas, mejorando el rendimiento ante múltiples usuarios simultáneos.
Payload	Carga útil. Contenido real del mensaje HTTP, generalmente en formato JSON (ej. {"cuenta": "123", "monto": 50}).
Rollback	Operación de base de datos que deshace una transacción en curso (ej. Depósito fallido), revirtiendo los datos a su estado anterior para garantizar integridad.
REST (Representational State Transfer)	Estilo de arquitectura de software para sistemas hipermedia distribuidos como la World Wide Web. Utiliza verbos HTTP (GET, POST, PUT) y es más ligero que SOAP.
Swagger / OpenAPI	Herramienta y especificación para describir, producir, consumir y visualizar servicios web RESTful. (Reemplaza a WSDL).
Gestionar	Hacer las acciones o los trámites necesarios para conseguir o

	resolver una cosa.
Automatizar	Aplicar máquinas o procedimientos automáticos en la realización de un proceso o en una industria o empresa.
Control de procesos	El control es una de las principales actividades administrativas dentro de las organizaciones. El control es el proceso de verificar el desempeño de distintas áreas o funciones de una organización.
Integral	Comprende todos los aspectos o todas las partes necesarias para estar completo.
Sostenibilidad	Cualidad de sostenible, especialmente las características del desarrollo que asegura las necesidades del presente sin comprometer las necesidades de futuras generaciones.
Requisito	Condición necesaria para cumplir las expectativas de algo.
Base de datos	Una base de datos es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Las bases de datos tradicionales se organizan por campos, registros y archivos.
Query/Consulta	Un query en base de datos es una búsqueda o pedido de datos almacenados en una base de datos. En forma genérica, query también puede tratarse de una inserción, actualización, búsqueda y/o eliminación en una base de datos.
MYSQL	Sistema de gestión de bases de datos relacional desarrollado bajo licencia libre.
IDE	Aplicación informática que proporciona servicios integrales para facilitarle el desarrollo de software.
Visual Studio 2022	Es un entorno de desarrollo integrado (IDE) de Microsoft para crear software en Windows, que permite a los programadores escribir, depurar, compilar y publicar código para diversas plataformas como web, móvil y escritorio
Dotnet	Plataforma de desarrollo gratuita y de código abierto para crear aplicaciones web, de escritorio, móviles y para la nube, que se pueden ejecutar en Windows, Linux y macOS. Incluye lenguajes como C#, F# y Visual Basic.

4.2 Acrónimos y abreviaturas

Tabla 2. Acrónimo y abreviaturas

Acrónimo/Abreviatura	Significado
REST	Representational State Transfer (Transferencia de Estado Representacional)

RESTful	REST (Representational State Transfer) y API (Application Programming Interface).
APK	Android Package Kit (Formato de archivo de aplicaciones Android).
HTTP / HTTPS	Hypertext Transfer Protocol / Secure (Protocolo de transporte web).
CLI	Command Line Interface (Interfaz de Línea de Comandos).
SQL	Structured Query Language (Lenguaje de Consulta Estructurado).
DAO	Data Access Object (Patrón de diseño para aislar la capa de datos).
DTO	Data Transfer Object (Objeto simple para transferir datos entre procesos).
GCP	Google Cloud Platform.
Usuario	Persona o cliente que usará aplicación WEB.
ERS o SRS	Especificación de requisitos de software, Software Requirements Specification
API	Application Programming Interface
RF	Requisito funcional
RNF	Requisito no funcional.
DB	Database (Base de Datos)
CRUD	Operaciones de base de datos (Create, Read, Update, Delete – Crear, Leer, Modificar, Eliminar)
IDE	Integrated Development Environment
ECUD	Especificación de Casos de Uso Detallado
ROI	Retorno de la inversión
JSON	JavaScript Object Notation.

5. Referencias

Tabla 3. Referencias

Referencia	Titulo	Ruta	Fecha	Autor
1.	Standard IEEE 830-1998	Recommended Practice for Software Requirements Specifications. http://www.qualitatis.org	1988	IEEE Computer Society
2.	Video Tutorial Base	WEB SERVICE REST PARTE 01 (Base lógica del negocio) - https://www.youtube.com/watch?v=N6jRpOdR9lQ	2017	Desarrolla Software
3.	Introducción a Visual Studio - Microsoft	Introducción al IDE de Visual Studio https://visualstudio.microsoft.com/es/vs/getting-started/	s.f.	Microsoft
4.	Documentación Google Cloud	Cloud SQL for SQL SERVER / Networking in GCP.	2024	Google
5.	REST - Semantic Web Standards	REST. https://www.w3.org/2001/sw/wiki/REST	2025	W3C

6. Visión General del Documento

Este documento proporciona una descripción completa y detallada del sistema "Eureka Bank". Se estructura en tres módulos principales:

- Introducción y Contexto: Define el propósito, alcance y terminología (Secciones 1-5).
- Descripción General: Explica la arquitectura del sistema, los usuarios involucrados y el entorno operativo distribuido (Sección 7).
- Especificación Técnica: Detalla minuciosamente los requisitos funcionales (Casos de Uso), no funcionales (Rendimiento, Usabilidad) y restricciones de diseño para las cuatro plataformas cliente y el backend (Secciones 8-9).

7. Descripción General

El sistema "Eureka Bank" es una solución integral diseñada bajo el paradigma de

Arquitectura RESTful. Su núcleo es un servidor centralizado que expone la lógica de negocio bancaria a través de una Web API (Application Programming Interface) desarrollada en .NET. Esta arquitectura permite desacoplar la lógica de las interfaces, facilitando que múltiples canales (Web, Móvil, Escritorio, Consola) consuman los mismos recursos de datos y reglas de negocio mediante peticiones HTTP estándar, garantizando consistencia y seguridad en todas las transacciones.

Su función principal es actuar como un gestor inteligente de recursos bancarios, recibiendo solicitudes en formato JSON desde diversos puntos de contacto y procesándolas de manera segura contra una base de datos alojada en la nube. El sistema no se limita a ser un simple repositorio de datos, sino que incorpora reglas de negocio activas en sus controladores para garantizar la integridad financiera de cada operación antes de que esta sea confirmada. El despliegue en nube asegura disponibilidad 24/7.

7.1 Perspectiva del Producto

Desde una perspectiva arquitectónica, el sistema funciona como un Backend transaccional basado en recursos que interconecta la infraestructura de datos con las interfaces de usuario. A diferencia de sistemas monolíticos tradicionales, esta solución desacopla completamente la lógica financiera de la presentación visual. Esto significa que el "producto" es, en esencia, un motor de procesamiento bancario capaz de interpretar intenciones de negocio (como mover dinero o consultar un saldo) a través de endpoints (URLs) específicos, traduciéndolas en operaciones atómicas de base de datos.

El sistema opera en un entorno distribuido, donde el procesamiento de las reglas de negocio ocurre en un servidor de aplicaciones .NET (IIS/Kestrel), mientras que la persistencia de los datos reside en una instancia de alta disponibilidad en Google Cloud Platform. Esta separación garantiza que, independientemente del canal que utilice el cliente (Web, Móvil o Escritorio), las reglas de validación y seguridad sean idénticas y consistentes.

El sistema no es un producto aislado, sino un ecosistema compuesto por cinco nodos interconectados:

- Servidor de Aplicaciones (Backend): Nodo central que ejecuta el código C# (ASP.NET Core), procesa las solicitudes HTTP/JSON y gestiona las transacciones.
- Servidor de Datos (Cloud): Nodo de persistencia en Google Cloud Platform (SQL Server) que almacena la información financiera.
- Cliente Escritorio (Desktop): Aplicación Windows Forms/WPF en la red interna del banco, usada por cajeros para alta transaccionalidad.
- Cliente Web / Móvil: Aplicaciones cliente externas (Internet) que consumen la API REST para consultas de usuarios y gerentes.
- Cliente Consola: Herramienta de línea de comandos para administración y diagnóstico por parte del equipo de TI.

7.2 Funciones del Producto (software)

La funcionalidad del sistema gira en torno a la gestión segura y eficiente de los activos financieros de los clientes, estructurándose en cuatro grandes capacidades operativas que definen el comportamiento del software.

En primer lugar, el sistema incorpora un módulo estricto de Control de Acceso y Autenticación (Login). Antes de permitir cualquier operación, el software tiene la capacidad de verificar la identidad digital de los empleados y usuarios. Esta función no solo compara credenciales cifradas, sino que establece una sesión segura (token JWT) que acompañará al usuario durante su interacción, garantizando que cada transacción posterior pueda ser auditada y vinculada a un responsable específico.

En segundo lugar, el sistema provee una capacidad de Visibilidad Financiera Consolidada (Obtener Cuentas). Esta función permite a los gestores y gerentes obtener una radiografía completa de los productos bancarios asociados a una sucursal mediante peticiones HTTP GET. El software es capaz de recuperar no solo los números de cuenta y sus saldos actuales, sino que cruza esta información en tiempo real con los datos maestros de los clientes —como nombres, cédulas y estado— para presentar un reporte JSON integral. Es importante destacar que el sistema trata la información del cliente como datos de solo lectura; es decir, consume la información existente para darle contexto a las cuentas, pero no permite la modificación o creación de nuevos clientes como parte de este alcance funcional.

En tercer lugar, el software ofrece una función de Auditoría Transaccional Detallada (Obtener Movimientos). Esta funcionalidad permite reconstruir la historia financiera de una cuenta específica. Al invocar esta función a través de la API REST, el sistema recupera cronológicamente cada operación que ha afectado el saldo, presentando detalles críticos como la fecha, la hora exacta, el monto involucrado y el tipo de operación. Esta capacidad es fundamental para la conciliación bancaria y para brindar transparencia al cliente final sobre el uso de sus fondos.

Finalmente, la función más crítica del sistema es el Motor de Procesamiento de Transacciones (Registrar Movimiento). Esta capacidad habilita al software para alterar los saldos de las cuentas mediante tres operaciones fundamentales: Depósitos, Retiros y Transferencias. El sistema no solo registra el dato, sino que ejecuta validaciones complejas en tiempo real, como verificar la existencia de fondos suficientes antes de un retiro o asegurar que una transferencia debite de una cuenta y acredite en otra de manera simultánea e indivisible. Esta funcionalidad es la encargada de mantener la consistencia contable del banco, asegurando que el dinero nunca se pierda ni se duplique ante fallos técnicos, aprovechando el manejo de transacciones nativo de ADO.NET / Entity Framework.A continuación, casos de uso propuestos. Ilustración 1. Diagrama de casos de uso del sistema (incluye actores: Administrador, Cliente, resumidos como empleados; casos: Login, Obtener Cuentas, Obtener Movimientos, Registrar Movimiento).

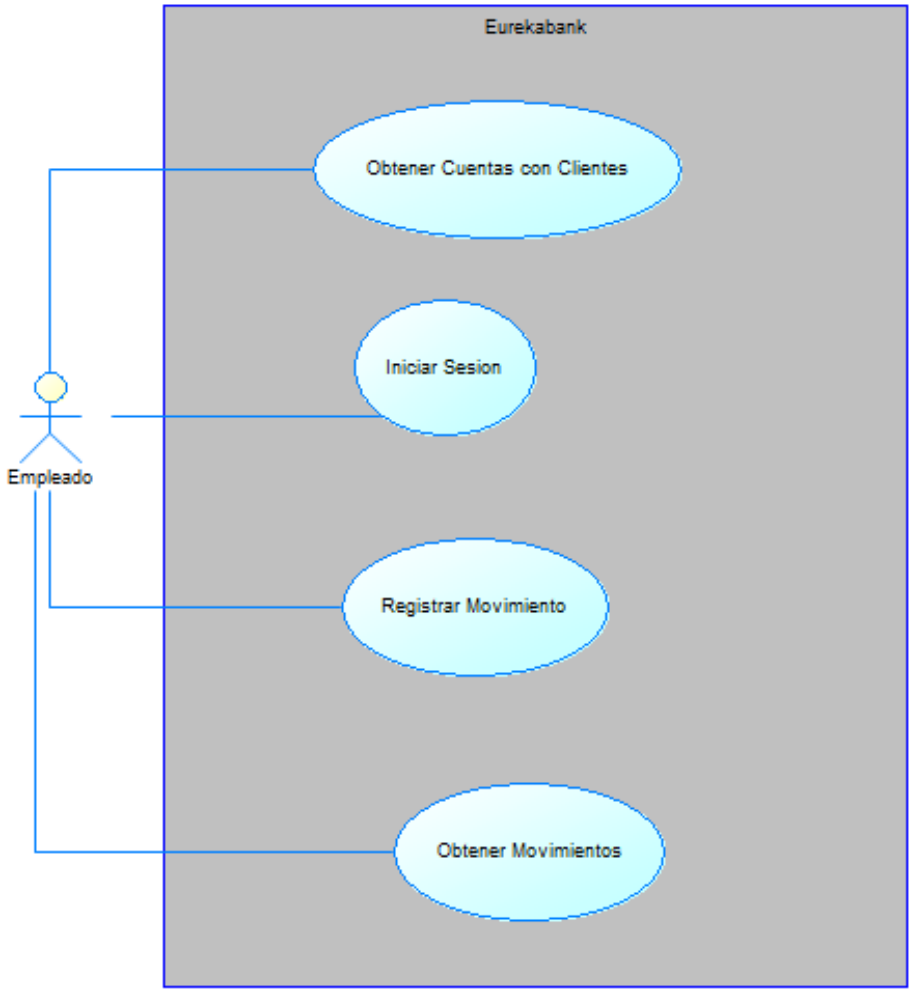

Ilustración 1. Diagrama de casos de uso del sistema

7.3 Condiciones del Entorno

El entorno operativo del sistema requiere una infraestructura de red robusta y estable. Dado que la base de datos SQL Server reside en la nube de Google, es condición indispensable que el servidor de aplicaciones mantenga una conexión a Internet de baja latencia y alta disponibilidad. Cualquier interrupción en la conectividad activará mecanismos de protección que revertirán las transacciones en curso para evitar inconsistencias. Asimismo, el entorno debe soportar protocolos de seguridad criptográfica (HTTPS/TLS) para proteger los datos sensibles (JSON) mientras viajan desde las terminales de los cajeros o dispositivos móviles hasta el núcleo de procesamiento en la nube.

7.4 Características de los Usuarios

Los usuarios interactúan con el sistema a través de las diferentes plataformas .NET disponibles (Web, Escritorio, Móvil, Consola), cada una adaptada a su perfil y responsabilidades dentro de la entidad bancaria. A continuación, se detallan las características de cada perfil.

Tabla 4. Usuario Administrador de Sistemas

Tipo de usuario	Administrador de Sistemas
Formación	Ingeniero de Sistemas, Tecnólogo en Informática o carreras afines.
Habilidades	Conocimiento avanzado en administración de servidores Windows Server/Linux, bases de datos SQL Server, redes y seguridad informática. Capacidad para ejecutar scripts de PowerShell y comandos en terminal.
Actividades	Responsable de la gestión técnica de la plataforma. Utiliza principalmente el Cliente Consola (.NET CLI) para verificar la conectividad con Google Cloud, monitorear los logs de transacciones del servidor de aplicaciones (IIS/Kestrel) y realizar mantenimientos de la base de datos. Gestiona los accesos de nivel superior.

Tabla 5. Usuario Gerente

Tipo de usuario	Gerente de Sucursal
Formación	Licenciatura en Administración de Empresas, Finanzas, Economía o Contabilidad.
Habilidades	Capacidad analítica para la interpretación de reportes financieros. Manejo intermedio de navegadores web y herramientas de ofimática. Visión estratégica del negocio.
Actividades	Utiliza el Cliente Web para supervisar la operatividad de la sucursal. Su función principal es consultar reportes consolidados de las cuentas activas y la información de los clientes asociados (Caso de Uso 1.1) consumiendo la API REST, con el fin de auditar el rendimiento y detectar anomalías en los saldos.

Tabla 6. Usuario Cajero

Tipo de usuario	Cajero de Ventanilla
Formación	Bachiller, Técnico Bancario o estudios universitarios en curso.
Habilidades	Alta velocidad y precisión en la digitación numérica. Conocimiento de procedimientos de caja, conteo de efectivo y validación de identidad. Manejo ágil de aplicaciones de escritorio Windows (Forms/WPF).
Actividades	Operador principal del sistema transaccional. Utiliza el Cliente de Escritorio (.NET) para procesar depósitos, retiros y transferencias en tiempo real (Caso de Uso 1.3). Es responsable de la entrada y salida física de dinero, validando los fondos en el sistema antes

	de entregar efectivo.

Tabla 7. Usuario Cliente

Tipo de usuario	Cliente Final
Formación	No requerida (Público en general).
Habilidades	Uso básico de dispositivos móviles inteligentes (Smartphones) y navegación en aplicaciones Android.
Actividades	Utiliza la Aplicación Móvil (Android) para realizar consultas de auto-servicio. Accede para verificar su saldo disponible y revisar el historial cronológico de sus últimos movimientos (Caso de Uso 1.2), consumiendo los servicios JSON de forma transparente.

7.5 Interfaces Externas

El sistema Eureka Bank actúa como un núcleo centralizado que expone interfaces modernas para la comunicación de datos. La interfaz principal es la API RESTful, que define un contrato flexible pero documentado (mediante Swagger/OpenAPI) para que las aplicaciones externas (Clientes) puedan solicitar operaciones utilizando verbos HTTP estándar (GET, POST, PUT).

Adicionalmente, el sistema mantiene una interfaz de persistencia crítica con el servicio de Google Cloud SQL para SQL Server, a través de la cual se realizan todas las operaciones de lectura y escritura. Esta comunicación se gestiona mediante el proveedor de datos Microsoft.Data.SqlClient o Entity Framework Core, asegurando una integración nativa y optimizada con el motor de base de datos. Estas interfaces son invisibles para el usuario final pero vitales para la arquitectura.

7.6 Restricciones

El desarrollo y operación del sistema están sujetos a las siguientes limitantes técnicas y de negocio:

- Hardware Cliente: Las PCs de los cajeros deben contar con al menos 4GB de RAM para ejecutar la JVM sin lentitud. Los móviles deben tener Android 8.0 o superior.
- Conectividad: Es mandatorio que el Servidor de Aplicaciones tenga salida a Internet estable para alcanzar la nube de Google. La latencia no debe superar los 200ms para garantizar la fluidez en caja.
- Tecnología Backend: El núcleo debe ser desarrollado exclusivamente en C# (.NET 6 o superior) utilizando ASP.NET Core para la creación de la API. No se permite el uso de servicios SOAP antiguos (WCF) a menos que sea estrictamente necesario por legado.

- Base de Datos: El motor debe ser Microsoft SQL Server (compatible con Cloud SQL) alojado en Google Cloud.
- Transaccionalidad: El sistema no debe permitir bajo ninguna circunstancia operaciones que generen inconsistencia (pérdida de dinero), obligando el uso de transacciones ACID.
- Protocolo: Comunicación basada en HTTP/HTTPS utilizando JSON como formato de intercambio de datos (Arquitectura REST).
- Hardware: Computador Core i5, 8GB de memoria RAM.
- Se trabaja con el sistema operativo Windows 11 tanto Home como Pro y Linux, distribuciones basadas en Debian.
- Navegador Web
- Se utilizará el lenguaje de programación Java.
- Para la elaboración del modelo conceptual, lógico y físico de la base de datos, se hará uso de la herramienta Power Designer.
- Los perfiles de los usuarios limitarán el uso de los módulos del sistema.
- La interfaz gráfica deberá ser amigable para el usuario y sencilla de utilizar.
- El sistema debe tener una conexión a internet estable que permita realizar transacciones sobre backend.

7.7 Suposiciones y Dependencias

En caso de que el sistema de “Eureka Bank” no cuente con el hardware detallado en los requisitos, existirá dos posibles soluciones:

- El personal del banco realiza la inversión para la adquisición del hardware necesario.
- Los requisitos cambiarán para ajustarse a la necesidad del cliente.

Si el cliente necesita realizar un cambio en algunos de los requisitos expuestos en el documento, se lo realizará con la anticipación de 7 días laborales; consecuentemente el requisito afectado tendrá un cambio tanto en el desarrollo como en el documento.

Se asume que la infraestructura de red del banco cuenta con las medidas de seguridad perimetral (Firewalls) adecuadas. Se depende directamente de la disponibilidad del servicio de Google Cloud Platform; una caída en la región de la nube afectará la operatividad total. Asimismo, se asume que los dispositivos móviles de los clientes cuentan con acceso a datos o WiFi para conectarse al servicio.

Otro factor importante por considerar es el sistema operativo con el que cuenta las computadoras donde se implementará el sistema, debido a que, si no es lo suficientemente bueno o estable, los requisitos sufrirán un cambio para posterior análisis de cómo solucionar esta problemática, o a su vez si los requisitos sufrirán un cambio y así poder acoplar al sistema operativo en funcionamiento. La conexión a internet es una parte esencial dentro del sistema a desarrollar, por ello es por lo que, si el cliente no cuenta con una conexión buena y estable, los requisitos sufrirán un cambio para ajustarse a lo que cuente la empresa.

- Infraestructura de Red: Se asume que la infraestructura de red del banco cuenta con las medidas de seguridad perimetral (Firewalls) adecuadas para permitir el tráfico saliente por el puerto 1433 (SQL Server) hacia Google Cloud.

- Disponibilidad de Nube: Se depende directamente de la disponibilidad del servicio de Google Cloud Platform; una caída en la región de la nube afectará la operatividad total, aunque SQL Server en Cloud ofrece alta disponibilidad.
- Conectividad Móvil: Se asume que los dispositivos móviles de los clientes cuentan con acceso a datos o WiFi para consumir la API REST.
- Dependencias de Software: El funcionamiento del sistema depende críticamente de la instalación del .NET Runtime (versión 6.0 o superior) en los equipos cliente de escritorio y en el servidor.
- Librerías: Se depende de paquetes NuGet específicos como Microsoft.EntityFrameworkCore.SqlServer para el acceso a datos y Newtonsoft.Json o System.Text.Json para el procesamiento de los mensajes.
- Entorno Operativo: Se asume que los equipos de los cajeros operan bajo Windows 10/11 para la ejecución óptima del cliente de escritorio .NET.

8. Especificación de Requerimientos

8.1 Requisitos comunes de las interfaces

8.1.1 Interfaces de usuario

El sistema, al ser multi-plataforma, presenta interfaces adaptadas a cada contexto. El entorno Web debe ser limpio y responsivo. El entorno de Escritorio debe priorizar la eficiencia con uso de teclado. El entorno Móvil debe ser táctil e intuitivo. El entorno de Consola debe ser textual y directo. Todas deben compartir la misma lógica de negocio provista por el backend.

8.1.2 Interfaces de hardware

Tabla 8. Requisitos de interfaces de hardware

Nombre	Detalle	Marca	Características	Descripción	Precio
Servidor de Aplicaciones	Nodo central de procesamiento (API)	Virtual (Google Cloud Compute Engine)	- CPU: 4 vCores - RAM: 8 GB - Disco: SSD 50GB	Ejecuta el servidor web Kestrel o IIS para alojar la Web API .NET. Procesa peticiones HTTP JSON y gestiona la lógica de negocio.	Variable (Según consumo Cloud)
Portátil	Principal dispositivo de salida (interfaz), que muestra datos o información. Útil en caso de desarrollo y de cliente	Genérico / Lenovo / HP	- CPU: Core i3 o sup. - RAM: 4GB min. - Teclado numérico	Hardware donde se ejecuta el Cliente Escritorio (Winforms). Debe permitir la digitación ágil de montos.	$500 - $900
Monitor	Principal dispositivo de	Acer	- 24"	El software deberá mostrar	$116

	salida (interfaz), que muestra datos o información. Es la Estación de trabajo cliente.		- Resolución: Full HD (1920 x 1080) - Relación de aspecto: 16:9 Tiempo de Respuesta: 5 ms - Frecuencia de actualización: 60 Hz. - Colores Admitidos: 16.7 millones - Relación de Contraste: 100,000,000:1 - Brillo: 250 cd/m²	información al usuario a través de la pantalla.	
Mouse	Dispositivo apuntador utilizado para facilitar el manejo de un entorno gráfico en una computadora.	Genius	Ratón óptico con sensor de 1000dpi.	El software debe interactuar con el movimiento del mouse y sus botones.	$6
Teclado	Dispositivo o periférico de entrada	Genius	Estándar de teclas de 104/105/106 Puerto USB: Si Soporte Sistema Windows10 /Windows®7/Vista/XP Soporte Interfaz USB	El software deberá interactuar con las pulsaciones del teclado. Con el teclado el usuario digitara las peticiones que requiera.	$10
CPU	La unidad central de procesamiento es el hardware dentro de un ordenador u otros dispositivos	Acer	Procesador: Intel Core I5 2da Gen o superior. compatibles con un sistema operativo como Windows 10, Linux (…), Memoria Ram: 8GB	En buen estado, el cual poseerá todos los controladores para poder manejar cada periférico de entrada y salida.	$399
Dispositivo Móvil	Terminal del cliente final	Genérico / Samsung / Xiaomi / Motorola	- Pantalla: 5"+ - RAM: 2GB -Conectividad 4G/WiFi	Dispositivo Android donde corre la App para consultas de saldo y movimientos.	$150 - $ 800
Servidor de Base de Datos	Instancia de Persistencia	Google Cloud Platform	-Conectividad 4G/WiFi - Tipo: e2-standard-2 - Almacenamiento: Persistente SSD	Instancia en la nube dedicada exclusivamente a ejecutar el motor SQL Server y almacenar la data financiera.	$ --

8.1.3 Interfaces de software

Tabla 9.Interfaces de software

	Detalles	Definición	Propósito
Sistema Operativo Servidor	Nombre: Windows Server / Linux N.º Especificación: x64 N.º Versión: 2019 / Ubuntu 20.04 Fuente: microsoft.com Distribución: Comercial / Open Source	Sistema operativo base para servidores. Al usar .NET Core, es posible desplegar tanto en entornos Windows (IIS) como Linux (Kestrel) en la nube.	Proveer la plataforma base para ejecutar la API REST y gestionar las conexiones de red seguras con Google Cloud Platform.
Sistema Operativo Cliente	Nombre: Microsoft Windows N.º Especificación: NT Kernel N.º Versión: 10 Pro / 11 Fuente: microsoft.com Distribución: Comercial Precio: ~$199 USD	Sistema operativo con interfaz gráfica desarrollado por Microsoft, estándar en entornos corporativos y estaciones de trabajo.	Servir de entorno para las PC de los cajeros, ejecutando el cliente de escritorio .NET y soportando los periféricos bancarios.
Sistema Operativo Móvil	Nombre: Android N.º Especificación: API Level 26+ N.º Versión: 8.0 a 14.0 Fuente: android.com Distribución: Open Source Precio: Gratuito	Sistema operativo móvil basado en el núcleo Linux, diseñado principalmente para dispositivos con pantalla táctil como teléfonos inteligentes.	Permitir la ejecución de la App móvil de Eureka Bank en los dispositivos personales de los clientes para consultas remotas.
Base de Datos	Nombre: Microsoft SQL Server N.º Especificación: SQL:2019 N.º Versión: 2019 / 2022 Fuente: microsoft.com/sql-server Distribución: Propietaria Precio: Licencia / Cloud Pricing	Sistema de gestión de bases de datos relacional desarrollado por Microsoft, robusto y nativo para entornos .NET, con soporte de transacciones ACID.	Almacenar de forma persistente y segura toda la información transaccional (cuentas, clientes, movimientos) en la nube.
Lenguaje de Programación	Nombre: C# (C Sharp) N.º Especificación: C# 10 / 11 N.º Versión: .NET 6.0 / 7.0 Fuente: dotnet.microsoft.com Distribución: Open Source (MIT) Precio: Gratuito	Lenguaje de programación moderno, orientado a objetos y seguro por tipos, desarrollado por Microsoft para la plataforma .NET.	Desarrollo unificado de la lógica del Backend (Web API), y de los clientes de Escritorio (Windows Forms) y Consola.
Entorno de Desarrollo	Nombre: Visual Studio 2022 N.º Especificación: IDE Platform N.º Versión: 17.x	Entorno de desarrollo integrado (IDE) completo de Microsoft, diseñado para optimizar el ciclo de vida del	Herramienta para escribir código, diseñar interfaces gráficas, depurar

	Fuente: visualstudio.microsoft.com Distribución: Community / Pro Precio: Gratuito (Community)	desarrollo en .NET.	errores y generar los ejecutables del proyecto.
Framework REST	Nombre: ASP.NET Core N.º Especificación: Web API N.º Versión: 6.0 / 7.0 Fuente: asp.net Distribución: Open Source Precio: Gratuito	Framework de alto rendimiento para construir servicios web modernos conectados a Internet y APIs RESTful.	Gestionar los controladores, el enrutamiento de URLs, la inyección de dependencias y la serialización de datos JSON.
Diseño de modelo de base de datos	Nombre: Power Designer Desarrollador: SAP Version:16.5 Distribución: Pagada Precio: 99.98 $	Es una herramienta para el análisis, diseño inteligente y construcción sólida de una base de datos y un desarrollo orientado a modelos de datos a nivel físico y conceptual, que da a los desarrolladores	El propósito de este programa en el proyecto es diseñar un modelo relacional de base de datos

8.1.4 Interfaces de comunicación

El sistema utilizará el protocolo HTTP/HTTPS sobre TCP/IP para el transporte de mensajes, siguiendo el estilo arquitectónico REST. A diferencia de los modelos basados en SOAP, la comunicación se basará en el intercambio de recursos en formato JSON (JavaScript Object Notation), lo que garantiza una transmisión de datos ligera y legible.

El servidor de aplicaciones (API) escuchará las peticiones en el puerto 443 (HTTPS) para asegurar la encriptación de datos en tránsito, o en el puerto 80/5000 para entornos de desarrollo internos.

La comunicación con la base de datos en la nube (Google Cloud SQL para SQL Server) se realizará a través del puerto estándar 1433, utilizando cadenas de conexión encriptadas (TLS) y autenticación robusta mediante Microsoft.Data.SqlClient.

Extracción de datos: Se obtiene la información mediante consultas optimizadas (LINQ / SQL Raw) desde la API hacia la base de datos. El sistema identifica los recursos solicitados (Cuentas, Clientes, Movimientos) y extrae la información relevante de las tablas en SQL Server.

Carga: Se escriben los datos en la base de datos transaccional. Esta fase ocurre cuando la API recibe una petición POST o PUT (ej. Registrar Movimiento), momento en el cual los datos JSON son deserializados, validados y persistidos en el sistema de destino en la nube (Cloud Platform), garantizando la integridad referencial.

8.2 Requisitos Funcionales

8.2.1. Objetivo General

El Sistema "Eureka Bank" permitirá la gestión centralizada y segura de las operaciones bancarias (depósitos, retiros, consultas) a través de múltiples canales de atención, garantizando la integridad de los datos en la nube.

8.2.2. Objetivos Específicos

A continuación, se muestran los objetivos específicos en el que se sustenta el proyecto de desarrollo de software.

Tabla 10. Autenticación y Seguridad (Login)

OBJ-001	Controlar el Acceso al Sistema
Descripción	Asegurar que solo personal autorizado (Empleados) y clientes validados puedan acceder a la información financiera, mediante un proceso de login único centralizado.
Importancia	Alta
Comentarios	

Tabla 11. Visibilidad Operativa (Cuentas)

OBJ-002	Centralizar Información de Cuentas
Descripción	Permitir a la gerencia obtener reportes inmediatos del estado de todas las cuentas de una sucursal, vinculando los datos financieros con la identidad del cliente.
Importancia	Alta
Comentarios	

Tabla 12. Auditoría (Movimientos)

OBJ-003	Proveer Historial Transaccional
Descripción	Garantizar que cada movimiento quede registrado y sea consultable cronológicamente, brindando transparencia total al cliente sobre el uso de su dinero.
Importancia	Alta
Comentarios	

Tabla 13. Integridad Transaccional (Registro)

OBJ-004	Ejecutar Transacciones Seguras
Descripción	Procesar depósitos y retiros aplicando reglas de validación de fondos y bloqueos de base de datos (ACID) para evitar pérdidas o inconsistencias financieras.
Importancia	Crítica
Comentarios	

8.2.3. Actores

Tabla 14. Actor Cajero (Escritorio)

ACT-001	Cajero
Descripción	Empleado del banco encargado de operar la aplicación de Escritorio para ejecutar transacciones monetarias físicas (ingresos y egresos).
Comentarios	

Tabla 15. Actor Gerente (Web)

ACT-002	Gerente de Sucursal
Descripción	Supervisor encargado de monitorear el rendimiento de la sucursal y consultar listados de clientes y cuentas vía Web.
Comentarios	

Tabla 16. Actor Administrador (Consola)

ACT-003	Administrador IT
Descripción	Personal técnico que utiliza la consola para verificar la salud del sistema y realizar pruebas de conectividad.
Comentarios	

Tabla 17. Actor Cliente (Móvil)

ACT-004	Cliente Final
Descripción	Usuario propietario de la cuenta que utiliza la App Móvil para consultar su saldo y movimientos personales.
Comentarios	

8.2.4. Lista de requisitos funcionales

A continuación, se presenta una lista con los requisitos funcionales del sistema “Eureka

Bank. Considerar que por el momento, se tendrá un superadmin que realice todos los movimientos.

Tabla 18. Lista de requisitos funcionales

Servicio de Eureka Bank

Lista de requerimientos funcionales			
Código	Requerimiento	Caso de uso	Actor
RF – 01	El sistema debe validar las credenciales de usuario (Usuario/Clave) contra la base de datos y retornar el perfil del empleado.	Login (CU 1.4)	Todos
RF – 02	El sistema debe permitir consultar todas las cuentas de una sucursal específica, mostrando saldo, moneda y nombre del cliente titular.	Obtener Cuentas (CU 1.1)	Administrador, Cliente
RF – 03	El sistema debe permitir consultar el historial detallado de movimientos de una cuenta, ordenados por fecha descendente.	Obtener Movimientos (CU 1.2)	Cliente / Cajero / Administrador
RF – 04	El sistema debe permitir registrar depósitos, retiros y transferencias, validando saldo suficiente y ejecutando la operación de forma atómica.	Registrar Movimiento (CU 1.3)	Cajero / Cliente

8.2.5 Requisitos Funcionales

Para el Sistema “Eureka Bank se ha identificado los siguientes requerimientos funcionales.

Tabla 19. Gestionar Login

RQF-001	**Gestionar Login**
Descripción	El sistema debe permitir recibir un usuario y contraseña desde cualquier cliente (Web, Móvil, Escritorio). Debe conectarse a la BD SQL Server, buscar en la tabla Empleado y verificar la coincidencia. Si es exitoso, retorna el objeto Empleado con su código de sucursal.
Objetivo	OBJ-001

Importancia	Alta
Estado	Aprobado
Estabilidad	Alta
Comentarios	

Tabla 20. Obtener Cuentas con sus Clientes

RQF-002	**Obtener Cuentas con Clientes**
Descripción	El sistema debe recibir un código de sucursal. Debe ejecutar una consulta SQL con INNER JOIN entre Cuenta y Cliente. Debe retornar una lista de objetos que contengan: Código de Cuenta, Saldo, Estado, Código de Cliente, Nombre y Apellido del Cliente.
Objetivo	OBJ-002
Importancia	Alta
Estado	Aprobado
Estabilidad	Alta
Comentarios	Solo lectura. No permite modificar datos del cliente.

Tabla 21. Obtener Movimientos de una Cuenta

RQF-003	**Obtener Movimientos**
Descripción	El sistema debe recibir un número de cuenta. Primero valida su existencia. Luego consulta la tabla Movimiento filtrando por la cuenta y ordenando por fecha reciente. Retorna una lista con: Número de Movimiento, Fecha, Tipo (Depósito/Retiro), Monto y Empleado responsable.
Objetivo	OBJ-003
Importancia	Alta
Estado	Aprobado
Estabilidad	Alta
Comentarios	

Tabla 22. Registrar Movimiento

RQF-004	**Registrar Movimiento (Depósito/Retiro)**
Descripción	El sistema debe gestionar la información del movimiento realizado, mostrar saldo actual, que tipo de movimiento será. El sistema recibe Cuenta, Monto y Tipo. 1. Inicia Transacción BD. 2. Bloquea registro (SELECT FOR UPDATE). 3. Valida Estado Activo. 4. Si es Retiro, valida Saldo >= Monto. 5. Actualiza Saldo (UPDATE). 6. Inserta Movimiento (INSERT). 7. COMMIT. Si algo falla, hace ROLLBACK.

Objetivo	OBJ-004
Importancia	Crítica
Estado	Aprobado
Estabilidad	Alta
Comentarios	Manejo estricto de excepciones ACID.

8.3 Requisitos No Funcionales

8.3.1 Objetivo General

Para el Sistema de “Eureka Bank” se ha identificado los siguientes requerimientos no funcionales en relación con la tecnología de la información, siendo el objetivo principal garantizar que el sistema Eureka Bank opere con altos estándares de calidad técnica.

Tabla 23. Requerimiento no funcional Tiempo de respuesta

RNF-001	**Tiempo de Respuesta**
Descripción	El sistema debe procesar transacciones críticas (Depósitos/Retiros) en menos de 3 segundos, incluyendo la latencia de red a Google Cloud. Las consultas deben responder en menos de 1 segundo.
Importancia	Alta
Estado	Aprobado

Tabla 24. Requerimiento no funcional utilización de colores

RNF-002	**Utilización de Colores**
Descripción	Las interfaces gráficas (Web, Móvil, Escritorio) deben usar una paleta consistente que no supere una gama de más de tres colores o que sean monocromáticos o del círculo cromático. Los montos negativos o de retiro deben resaltarse en Rojo.
Importancia	Media
Estado	Aprobado

Tabla 25. Requerimiento no funcional ícono de operaciones

RNF-003	**Íconos e Identidad**
Descripción	Uso de íconos estándar para acciones comunes (Lupa para buscar, Disco para guardar) en todas las plataformas para facilitar la usabilidad.
Importancia	Media

Estado	Aprobado

Tabla 26. Requerimiento no funcional métodos de acceso

RNF-004	**Métodos de Acceso**
Descripción	El acceso al sistema requiere obligatoriamente autenticación por usuario y contraseña. No se permite acceso anónimo a ningún servicio.
Importancia	Alta
Estado	Aprobado

Tabla 27. Requerimiento no funcional plataforma

RNF-005	**Plataforma Estándar**
Descripción	El backend debe ejecutarse sobre el entorno de tiempo de ejecución .NET (Common Language Runtime - CLR) versión 6.0 o superior, y la base de datos debe ser un sistema relacional compatible con Transact-SQL (Microsoft SQL Server).
Importancia	Alta
Estado	Aprobado

Tabla 28. Requerimiento no funciona accesibilidad

RNF-006	**Accesibilidad**
Descripción	Las interfaces deben ser claras y permitir la navegación intuitiva. La App Móvil debe soportar fuentes escalables.
Importancia	Media
Estado	Aprobado

Tabla 29. Requerimiento no funcional mantenimiento

RNF-007	**Documentación y Mantenimiento**
Descripción	El sistema deberá tener un manual de usuario para facilitar el mantenimiento futuro. El código fuente debe estar documentado utilizando el estándar de comentarios XML de .NET. Además, se debe entregar el manual de despliegue de la API y el script SQL para la generación de la base de datos.
Importancia	Alta

- El sistema contará con un nivel de acceso, basado en autenticación usuario y contraseña para un superadministrador.
- El sistema contará con un nombre de usuario y contraseña personal
- El sistema tendrá mecanismos de respaldo de información automáticos, los cuales aumentarán la integridad de estos al usar una base de datos de Cloud.
- Emitirá mensajes de alertas cuando el usuario realice un procedimiento erróneo en la ejecución del sistema.
- Emitirá advertencias y mensajes de confirmación en el momento de registrar, modificar y eliminar detalles de los movimientos.
- Las Búsquedas se realizarán clasificándolas según: Fecha, cliente o movimiento.
- El administrador podrá consultar el estado de las cuentas con montos disponibles de cada cliente.
- El administrador puede buscar datos de todas las cuentas.

8.4 Otros Requerimientos

8.4.1 Restricciones de Diseño

Las restricciones de diseño son las siguientes:

- No sobrecargar con muchas imágenes la parte principal del usuario, limitarla a logos, e imágenes llamativas de cada tipo de movimiento, por lo demás se trabajará solamente con íconos para mantener la semiótica.
- Las imágenes deben ser en formato png de acuerdo al tipo de espacio, así que variarán los pixeles, pero no deben superar los 250 x 250px.
- Se mantenga el orden del menú principal, submenú y pantalla de operación como container del sistema.
- Formato minimalista, colores bajos que no canse la visión del usuario.
- Arquitectura REST pura: lógica en el servidor, presentación en el cliente.
- Diseño Responsivo en cada cliente a excepción del de consola, es decir, para móvil, escritorio y web.

8.4.2 Restricciones de Hardware

- El servidor debe tener redundancia de disco para evitar pérdida de datos en caso de fallo físico (manejado por GCP).
- Los equipos disponibles deben tener las siguientes características:
    - Procesador Intel Core i5 o superior,
    - 8 GB de RAM o superior,
    - Disco de 250 GB o superior,

- Sistema Operativo Windows 10 u 11 versión Home o Pro, además puede ser Linux con distribuciones Debian.

Por este motivo es necesario procurar que el sistema se adapte a dichas capacidades, condicionado a una mejora inmediata de los mismos.

8.4.3 Atributos de calidad

Fiabilidad. - El sistema, ofrecerá garantía de funcionar correctamente y cumplir con los requisitos presentados previamente. Integridad de datos garantizada al 100%.

Portabilidad. - Al tratarse de una aplicación orientada a distintos clientes, se puede ejecutar en cualquier dispositivo electrónico que posea un navegador y una conexión a internet.

Seguridad. - El sistema tendrá mecanismos que brindan la posibilidad de que los datos sean resguardados satisfactoriamente. Se usarán los siguientes mecanismos de acceso a los datos:

- Cada usuario tiene una cuenta, pero la cuenta general la dirige el super administrador.
- Para utilizar el sistema, es necesario introducir un usuario y una contraseña, ambos previamente creados y ejecutados por un script SQL.

Disponibilidad. - Los usuarios podrán acceder al sistema durante las 24 horas del día, respaldado por su base de datos en la nube.

9. Requisitos de Rendimiento

9.1. Requisitos de Interfaces Externas

9.1.1. Interfaces de Usuario

Los formularios deben validar tipos de datos (números vs letras) antes de enviar la petición RESTful para reducir carga al servidor.

Las interfaces que se elaboran en las plataformas web, escritorio y móvil incluyen:

- Botones representativos para elegir las opciones de tipo de movimiento.
- Pestañas específicas para cada movimiento a manera de modales.
- Mensajes informativos que mantengan al tanto de si una operación se realizó exitosamente.
- Mensajes de error representativos en caso de que fallen los procesos, que permitan saber al usuario lo que pasó.
- Formularios para el ingreso, modificación, actualización y búsqueda de datos, para completar las operaciones CRUD.
- Considerar colores que lleven a la semiótica de las acciones, como gris o rojo para

cancelar, el color principal para operaciones de éxito.

- Utilización de Hovers e ícono de cursores para dar un feed visual al usuario.

9.1.2. Interfaces de Hardware

La pantalla del monitor/ teléfono: El software muestra información al usuario a través de la pantalla del monitor o del móvil, adaptando la resolución y disposición de los elementos (Responsive Design) según el dispositivo.

Ratón: El software interactúa con el movimiento y los botones del ratón. El ratón activa las zonas de entrada de datos, botones de comando y selección de las opciones de los menús en las interfaces Web y Escritorio.

Teclado: El software interactúa con el sistema capturando la entrada de datos alfanuméricos en el momento de realizar búsquedas o transacciones.

Servidor de aplicaciones: Debe gestionar eficientemente la memoria (Managed Heap del CLR) para soportar la serialización y deserialización de objetos JSON grandes durante la generación de reportes masivos, evitando fugas de memoria.

9.1.3. Interfaces de Software

- Sistema Operativo: Compatible con Windows 10/11 (Clientes PC), Android 8-14 (Móvil), Linux (Servidor).
- Navegador Web: Chrome, Mozilla, Safari, Opera.

9.1.4. Interfaces de Comunicación

Los servidores y clientes se comunicarán mediante una arquitectura REST sobre el protocolo HTTP/S.

- Formato: Se utilizará JSON para el intercambio de datos, lo que garantiza una comunicación ligera.
- Puertos: Se utilizarán puertos estándar de internet (80, 443) para facilitar el paso a través de firewalls corporativos y asegurar la encriptación del tráfico (HTTPS).

9.1.5. Base de Datos

El sistema dispondrá de comunicación persistente y segura con la base de datos "EurekaBank" (SQL Server) alojada en Google Cloud.

- Conectividad: Se utilizará el proveedor de datos ADO.NET o el ORM Entity Framework Core, configurado con un pool de conexiones optimizado para soportar múltiples hilos de ejecución simultáneos y gestionar la concurrencia de transacciones de manera eficiente.

RECOMENDACIONES

- Se recomienda implementar un certificado SSL/TLS en el servidor de aplicaciones para cifrar el tráfico JSON y proteger los datos sensibles de los clientes durante la transmisión.
- Es importante realizar pruebas de carga (Stress Testing) sobre los endpoints de la API REST antes del lanzamiento para calibrar el pool de conexiones a la base de datos en la nube y optimizar el rendimiento de Kestrel/IIS.
- Se sugiere desarrollar librerías cliente (SDKs o DTOs compartidos) para los proyectos de Consola y Escritorio, utilizando HttpClientFactory, para evitar la duplicidad de código en la generación de peticiones HTTP.

CONCLUSIONES

- El modelo del sistema Eureka Bank cumple con los requerimientos de modernización al integrar múltiples canales (Web, Móvil, Escritorio) bajo una arquitectura centralizada y robusta basada en servicios.
- La especificación detallada de los requisitos funcionales transaccionales (Login, Movimientos) asegura que el equipo de desarrollo tiene una hoja de ruta clara para implementar la lógica de negocio crítica en C# (.NET).
- El uso de estándares como REST, JSON y SQL en la nube garantiza la interoperabilidad, escalabilidad y seguridad de la solución a largo plazo, facilitando la integración con futuras plataformas.

BIBLIOGRAFÍA

IEEE Computer Society. (1998). Standard IEEE 830: Recommended Practice for Software Requirements Specifications. https://ieeexplore.ieee.org/document/720574

Desarrolla Software. (2017). Tutorial Web Services y Base de Datos (Base lógica del negocio). YouTube. https://www.youtube.com/watch?v=N6jRpOdR9lQ

Google Cloud. (2024). Cloud SQL para SQL Server. https://cloud.google.com/sql/docs/sqlserver

Microsoft. (2024). Documentación de ASP.NET Core. https://learn.microsoft.com/es-es/aspnet/core/

Microsoft. (2024). Documentación técnica de SQL Server. https://learn.microsoft.com/es-es/sql/sql-server/