1. Generalidades

Desarrollar una aplicación que será utilizada por los trabajadores de una entidad financiera. Dicha aplicación permitirá la administración de los clientes, lo cual incluye el registro de clientes, la actualización de sus datos y la eliminación de los mismos. También permitirá crear productos financieros a sus clientes. Finalmente, a estos productos financieros se le podrán realizar movimientos transaccionales y consultar los estados de cuentas.

2. Requerimientos funcionales

Se debe crear un CRUD de clientes productos y transacciones y exponer un servicio Rest que permita cumplir con las siguientes indicaciones:

Clientes:

• La aplicación debe permitir crear un cliente con los siguientes atributos como mínimo: id, Tipo de identificación, número de identificación, nombres, apellido, correo electrónico, fecha de nacimiento, fecha de creación, fecha de modificación.
• La aplicación debe permitir modificar la información del cliente. Cuando se realiza una modificación de información de un cliente, se debe calcular esta fecha de modificación automáticamente, tomando la última fecha que se realiza la modificación. • La aplicación debe permitir eliminar un cliente que ha sido creado.
• Un cliente no podrá ser creado, ni existir en base de datos si es menor de edad. • Un cliente no podrá ser eliminado si tiene productos vinculados.
• La fecha de creación debe ser calculada automáticamente al registrar un cliente.

• Requerimientos opcionales:
- El campo correo electrónico solo debe permitir ingresar valores que correspondan con un correo electrónico, es decir, debe ser de un formato xxxx@xxxxx.xxx - La extensión del nombre y el apellido NO puede ser menor a 2 caracteres.

Productos (Cuentas) :

• La aplicación debe permitir crear únicamente dos tipos de productos: cuenta corriente o cuenta de ahorros.
• Un producto financiero, sólo podrá existir en la base de datos si está vinculado a un cliente de la entidad financiera.
• Las cuentas corrientes o de ahorro se deben crear con los siguientes atributos como mínimo: id, tipo de cuenta, número de cuenta, estado (activa, inactiva, cancelada), saldo, exenta GMF, fecha de creación, fecha modificación, usuario al que pertenece la cuenta. • La cuenta de ahorros no puede tener un saldo menor a $0 (cero).
• Las cuentas corrientes y de ahorros se pueden activar o inactivar en cualquier momento. • El número de las cuentas corrientes y de ahorros deben ser únicos y generarse automáticamente, la extensión del número de cuenta debe ser de 10 dígitos numéricos. El

número de las cuentas ahorro debe iniciar en “53” y el número de las cuentas corriente debe iniciar en “33”.
• Al crear una cuenta de ahorro esta debe establecerse como activa de forma predeterminada. • Solo se podrán cancelar las cuentas que tengan un saldo igual a $0.
• La fecha de creación de cada producto debe ser calculada automáticamente al registrar el producto. • El saldo de la cuenta deberá actualizarse al realizar cualquier transacción exitosa.

Transacciones (movimientos financieros):

• La aplicación debe permitir crear únicamente las siguientes transacciones: Consignación, Retiro y Transferencia entre cuentas.
• La aplicación debe actualizar el saldo y el saldo disponible con cada transacción realizada. • Las transferencias sólo se podrán realizar entre cuentas existentes en el sistema. Al realizar una transferencia se deben generar los movimientos de crédito en la cuenta de recepción y el movimiento débito en la cuenta de envío.

Importante: Se debe implementar las estructuras de persistencia en la base de datos necesaria para que se pueda cumplir con todos los requerimientos funcionales obligatorios.

3. Requerimientos no funcionales:

• Se deben desarrollar como mínimo un proyecto backend
• El proyecto Backend deberá ser desarrollado utilizando JAVA.
• El proyecto debe utilizar una arquitectura hexagonal en lo posible o MVC (Modelo, Vista, Controlador)
• El proyecto debe utilizar una base de datos de las siguientes opciones SQL SERVER, Oracle, PostgreSQL o MySQL.
• El proyecto del back se debe generar por capas utilizando al menos las siguientes: entity, service, controller, repository.
Se debe hacer uso de patrones de diseño GOF que se adapten a la solución.
Implementar sistema de seguridad para las peticiones
Se debe dejar trazabilidad en un archivo de logs.

Test Unitarios

• La aplicación debe implementar test unitarios utilizando la librería Junit con una cobertura para las capas Service y Controller.

Control de Versiones:

• Se debe utilizar el motor de versionamiento de Git.
• Se debe crear un solo repositorio en GitHub que tenga el código fuente del proyecto a realizar y se debe evidenciar al avance de los proyectos mediante commits y push.