sophos solutions

Reto Desarrollo Bootcamp Sophos Fullstack Portales

1. Generalidades

Desarrollar una aplicación Web 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 crearle 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

2.1. 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, usuario creación, fecha de modificación y usuario modificación. back hecho
- La aplicación debe permitir modificar la información del cliente. Cuando se realiza una modificación de información de un cliente, se debe calcular esta fecha automáticamente, tomando la última fecha que se realiza la modificación. back hecho
- La aplicación debe permitir eliminar un cliente que ha sido creado. Back hecho/
- 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 que no se encuentren cancelados. condicional
- Se debe poder visualizar en una sola pantalla todos los clientes que existen en la entidad. fROM EN PROCESO
- La fecha de creación debe ser calculada automáticamente al registrar un cliente. De manera predeterminada el usuario de creación y modificación deberá ser admin.
- **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.
  - Si se cuenta con la parte de seguridad implementada, el usuario de creación y modificación deberá asignarse de acuerdo al usuario que ha iniciado sesión en la aplicación.

sophos solutions

2.2. Productos (Cuentas)

- La aplicación debe permitir crear únicamente dos tipos de productos: cuenta corriente o cuenta de ahorros.
- 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, saldo disponible, exenta GMF, fecha de creación, usuario creación, fecha modificación, usuario modificación.
- La cuenta de ahorros no puede tener un saldo menor a $0 (cero).
- La cuenta corriente puede sobregirarse hasta un máximo de $3.000.000.
- 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 “46” y el número de las cuentas corriente debe iniciar en “23”
- Al crear una cuenta de ahorro esta debe establecerse como activa de forma predeterminada.
- Solo se podrán cancelar las cuentas que tengan un saldo inferior a $1, pero que no cuente con deudas con la entidad financiera para esa cuenta.
- Se debe poder visualizar en una sola pantalla los productos por cliente con su tipo de cuenta, número de cuenta, saldo y saldo disponible.
- Un producto financiero, solo podrá existir en la base de datos si está vinculado a un cliente de la entidad financiera.
- La fecha de creación de cada producto debe ser calculada automáticamente al registrar el producto.
- El saldo y saldo disponible de la cuenta deberá actualizarse al realizar cualquier transacción exitosa.
- **Requerimientos opcionales:**
    - Solo se podrá marcar una cuenta por cliente como exenta GMF
    - El saldo disponible deberá calcularse como el saldo menos el valor GMF de ese saldo, asumiendo que ese valor en algún momento se va a debitar. Tener presente que si una cuenta está marcada como exenta el valor GMF será igual a cero 0.
    - En la pantalla donde se visualizan los productos del cliente se deben ordenar los productos de mayor a menor saldo, Estado(mostrando primero las cuentas activas, después las inactivas y al final las cuentas canceladas).
    - En la pantalla donde se visualizan los productos del cliente debe si una clienta se encuentra marcada como exenta GMF, mostrarla de un color diferente a las demás.

- Si se cuenta con la parte de seguridad implementada, el usuario de creación y modificación deberá asignarse de acuerdo al usuario que ha iniciado sesión en la aplicación.

2.3. Transacciones (movimientos financieros):

- La aplicación debe permitir crear únicamente las siguientes transacciones: Consignación, Retiro y Transferencia entre cuentas.
- Se debe poder acceder a una pantalla llamada Estado de cuenta en la cual se verifique el resumen de todas las transacciones que ha tenido cada producto. El estado de cuenta como mínimo debe contener la siguiente información fecha del movimiento, tipo de transacción, descripción, valor, tipo de movimiento (débito o crédito), saldo y saldo disponible.
- La aplicación debe actualizar el saldo y el saldo disponible con cada transacción realizada.
- Las transferencias solo 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 debito en la cuenta de envío.
- Las cuentas inactivas permitirán generar movimientos crédito, pero no permitirán movimientos débito.
- **Requerimientos opcionales:**
    - La aplicación debe calcular el GMF (Gravamen al Movimiento Financiero: 4x1000) sobre cada movimiento débito siempre y cuando la cuenta no se encuentra marcada como exenta GMF.
    - Solo se podrá marcar una cuenta por cliente como exenta GMF
    - El saldo disponible deberá calcularse como el saldo total menos el valor GMF de ese saldo, asumiendo que ese valor en algún momento se va a debitar. Tener presente que si una cuenta está marcada como exenta, el valor GMF será igual a cero 0. Por ejemplo, si una cuenta tiene un saldo total de $2.000, el saldo disponible será $1.992.

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. Se debe contar con una página de navegación de inicio desde donde se pueda navegar a las diferentes funcionalidades de la aplicación y con una página donde se muestre la información de la persona que desarrollo el reto. En donde se incluya al menos la siguiente información: Nombre, correo electrónico, profesión (si cuenta con ella), dirección repositorio Github, perfil profesional.
Nota: Un tipo de movimiento débito se entiende como un movimiento de salida de dinero de la cuenta, un tipo de movimiento crédito se entiende como un movimiento de ingreso de dinero de la cuenta.

sophos solutions

3. Requerimientos no funcionales:

3.1. Arquitectura:

- Se deben desarrollar como mínimo dos proyectos: uno para el Backend y otro para Frontend.
- El proyecto Backend deberá ser desarrollado utilizando JAVA + Springboot.
- El proyecto Frontend deberá ser desarrollado utilizando Angular.
- El proyecto debe utilizar una base de datos de las siguientes opciones Oracle, PostgreSQL o MySQL.
- De forma opcional se podrán utilizar librerías adicionales como Bootstrap.
- El proyecto del back se debe generar por capas utilizando al menos las siguientes: entity, service, controller, repository.
- El tipo de respuestas del controller deben ser de tipo ResponseEntity, dándole un manejo adecuado a las excepciones con tipos de respuestas HTTP al menos exitosos y de fallo.

3.2. Seguridad (Opcional)

- La aplicación debe implementar las buenas prácticas de desarrollo.
- La aplicación debe tener control de acceso en el Frontend mediante usuario y contraseña.
- La aplicación debe tener control de acceso en el Backend mediante el uso de JWT y utilizando la librería de SpringSecurity.
- Si se realiza la parte de seguridad, se debe generar una nueva clase tipo usuario para darle el manejo de varios usuarios y la contraseña en base de datos debe guardarse codificada.

3.3. Test Unitarios (Opcional)

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

3.4. Idioma de desarrollo

Todo el desarrollo deberá realizarse en idioma inglés, es decir, el nombramiento de variables, clases, métodos, etc. Por ejemplo para la clase clientes usar Clients.

3.5. 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 de los dos proyectos a realizar (Backend y Frontend) y se debe evidenciar al avance de los proyectos mediante commits y push.

4. Criterios de aceptación

- Los proyectos se deben desplegar exitosamente en ambiente local para su respectiva sustentación.
- El código de fuente final de los proyectos debe estar versionado y alojado en GitHub.
- La UI del proyecto debe ofrecer una buena experiencia de usuario.
- Se evaluará la implementación de todos los requerimientos funcionales y no funcionales obligatorios, es decir, que no estén señalados como opcionales.