SISTEMA BANCARIO

MANUAL TECNICO

Presentación

El siguiente manual se ha desarrollado con la finalidad de dar a conocer la información necesaria para realizar mantenimiento, instalación y exploración del software para un Sistema Bancario, el cual consta de diferentes funcionalidades para poder tener una correcta gestión de Tarjetas dentro del Sistema. El manual ofrece la información necesaria de ¿cómo está realizado el software? para que la persona (Desarrollador en JAVA) que quiera editar el software lo haga de una manera apropiada, dando a conocer la estructura del desarrollo del aplicativo.

Resumen

El manual detalla los aspectos técnicos e informáticos del software para el Sistema Bancario con la finalidad de explicar la estructura del aplicativo al personal que quiera administrarlo, editarlo o configurarlo. La siguiente guía se encuentra dividida en las herramientas que se usaron para la creación del software con una breve explicación paso a paso, El aplicativo maneja diferentes funcionalidades los cuales se explicará que funcionamiento realiza cada uno de ellos, dando sugerencias para el debido uso del sistema de información.

Introducción

El manual se realiza con el fin de detallar el software para el Sistema Bancario en términos técnicos para que la persona que vaya a administrar, editar o configurar el aplicativo lo haga de una manera apropiada. El documento se encuentra dividido en las siguientes secciones:

- ASPECTOS TEORICOS: Se darán a conocer conceptos, definiciones y explicaciones de los componentes del aplicativo desde un punto de vista teórico para mayor entendimiento por parte del lector sobre el funcionamiento del sistema de información y herramientas.
- DIAGRAMAS DE MODELAMIENTO: Se compone por diagramas e ilustraciones alusivos al funcionamiento del aplicativo.

ASPECTOS TECNICOS

El aplicativo del Sistema Bancario tiene la finalidad de administrar Tarjetas que pertenezcan a determinados Clientes. Se recomienda que el siguiente manual sea manipulado únicamente por la persona que quiera administrar, editar o configurar el software para el Sistema Bancario para velar por la seguridad de los datos que se almacenan.

1. Herramientas utilizadas para el desarrollo: En ésta sección se procede a explicar las herramientas informáticas empleadas para el desarrollo del aplicativo:
   1.1. Apache NetBeans: NetBeans es un entorno de desarrollo integrado (IDE) para Java. NetBeans permite desarrollar aplicaciones a partir de un conjunto de componentes de software modulares llamados módulos. NetBeans se ejecuta en Windows, macOS, Linux y Solaris. Además del desarrollo en Java, cuenta con extensiones para otros lenguajes como

PHP, C, C++, HTML5 y JavaScript. Las aplicaciones basadas en NetBeans, incluido NetBeans IDE, pueden ser ampliadas por desarrolladores externos.

NetBeans IDE es un entorno de desarrollo integrado de código abierto. NetBeans IDE admite el desarrollo de todos los tipos de aplicaciones Java (Java SE (incluido JavaFX), Java ME, web, EJB y aplicaciones móviles) listas para usar. Entre otras características se encuentran un sistema de proyectos basado en Ant, soporte Maven, refactorizaciones, control de versiones (compatible con CVS, Subversion, Git, Mercurial y Clearcase).

La plataforma Apache NetBeans es un marco genérico para aplicaciones Swing. Proporciona la "plomería" que, antes, cada desarrollador tenía que escribir por sí mismo: guardar el estado, conectar acciones a elementos del menú, elementos de la barra de herramientas y atajos de teclado; gestión de ventanas, etc.

1.2. MySQL Workbench: Workbench es una herramienta gráfica para trabajar con servidores y bases de datos MySQL. MySQL Workbench es totalmente compatible con MySQL Server 8.0. Las versiones obsoletas de MySQL Server pueden ser incompatibles con MySQL Workbench y deben actualizarse antes de intentar realizar una conexión. Las versiones posteriores a la 8.0, como la 8.4, pueden conectarse, pero es posible que algunas funciones no sean compatibles. La funcionalidad de MySQL Workbench cubre cinco temas principales:

1.2.1. Desarrollo de SQL: le permite crear y administrar conexiones a servidores de bases de datos. Además de permitirle configurar parámetros de conexión, MySQL Workbench brinda la capacidad de ejecutar consultas SQL en las conexiones de bases de datos mediante el Editor SQL integrado.

1.2.2. Modelado de Datos (diseño): le permite crear modelos de su esquema de base de datos de forma gráfica, realizar ingeniería inversa y directa entre un esquema y una base de datos activa, y editar todos los aspectos de su base de datos mediante el completo Editor de tablas. El Editor de tablas proporciona funciones fáciles de usar para editar tablas, columnas, índices, activadores, particiones, opciones, inserciones y privilegios, rutinas y vistas.

1.2.3. Administracion de Servidor: le permite administrar instancias del servidor MySQL administrando usuarios, realizando copias de seguridad y recuperaciones, inspeccionando datos de auditoría, visualizando el estado de la base de datos y monitoreando el rendimiento del servidor MySQL.

1.2.4. Migracion de Datos: permite migrar desde Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostgreSQL y otras tablas, objetos y datos RDBMS a MySQL. La migración también permite migrar desde versiones anteriores de MySQL a las versiones más recientes.

1.2.5. Soporte para MySQL Enterprise: Soporte para productos empresariales como MySQL Enterprise Backup, MySQL Firewall y MySQL Audit.

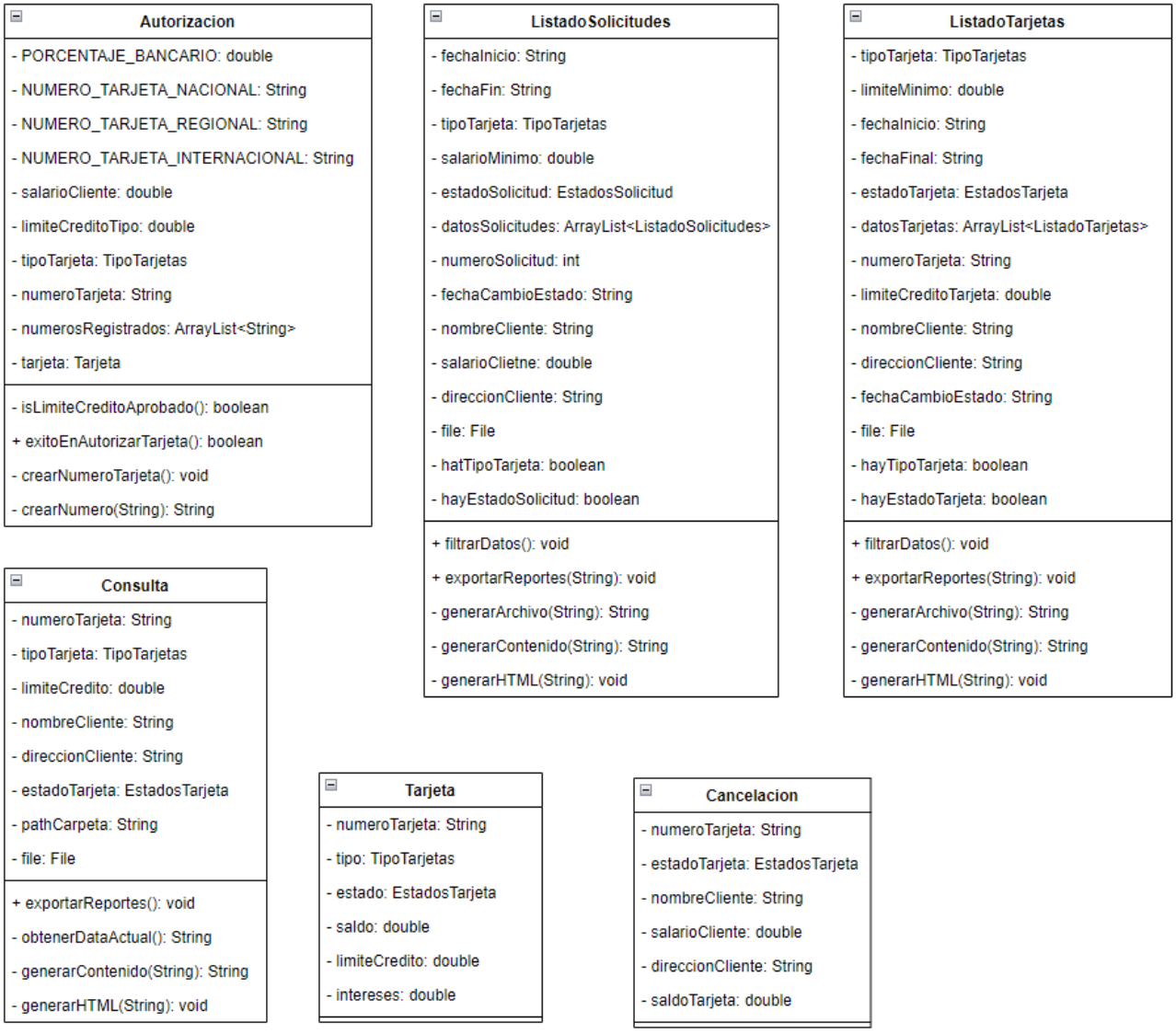
DIAGRAMAS DE MODELAMIENTO

1. DIAGRAMA DE CLASES

El diagrama de clases está compuesto de las entidades, métodos y atributos que se crearon para el funcionamiento del software.

BackEnd

En las siguientes imágenes se detalla cada una de las entidades (clases) usadas para el funcionamiento lógico del aplicativo, en el cual cada una de ellas realiza las funciones posteriormente descritas:

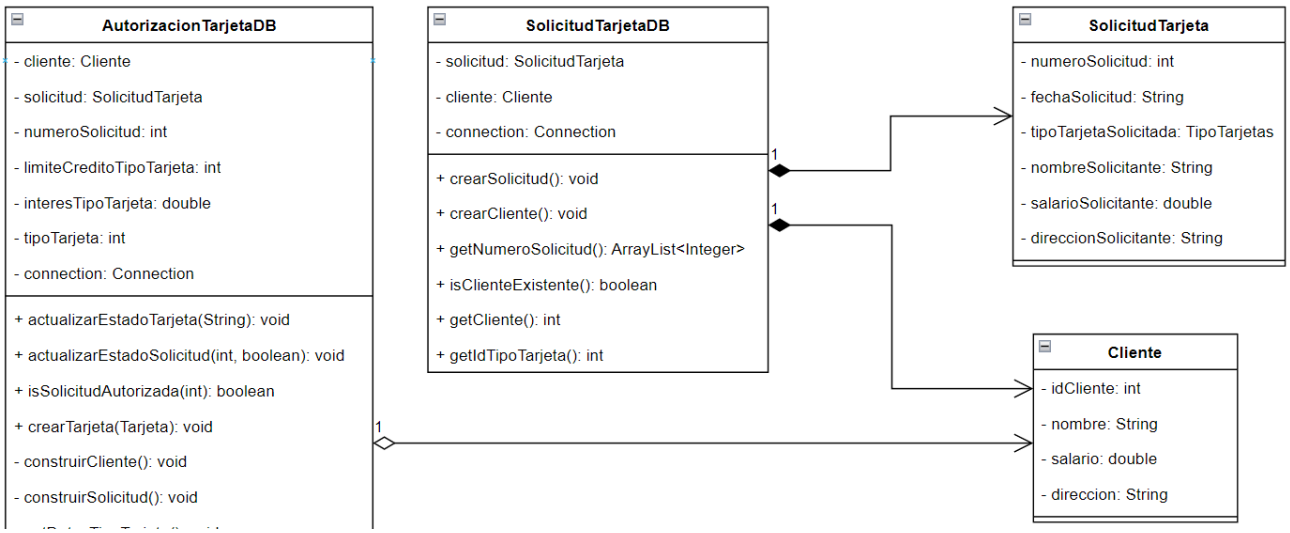


- **Autorizacion:** Entidad que almacena los datos para determinar la autorización de una solicitud y así generar una nueva tarjeta dentro del sistema

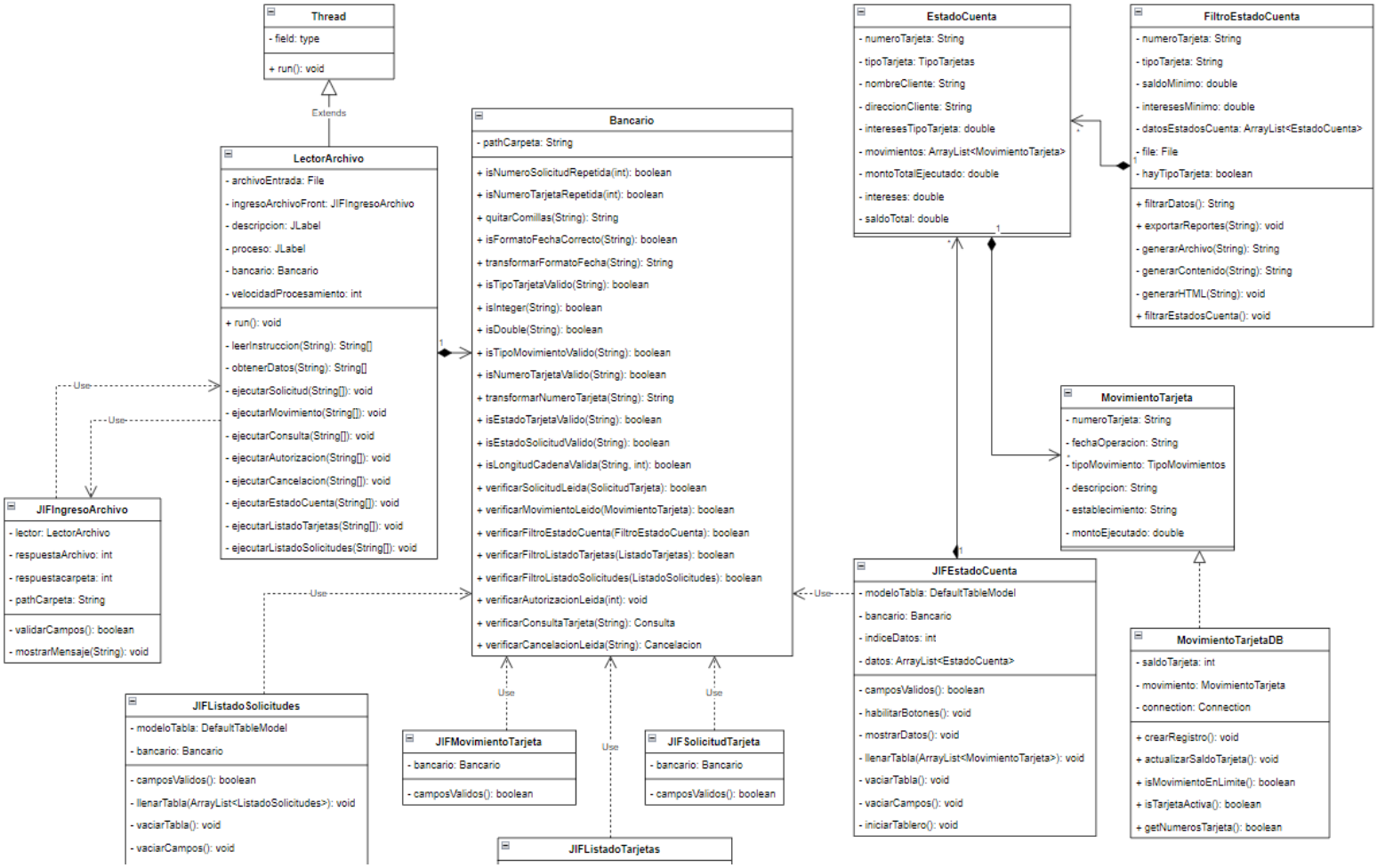
- isLimiteCreditoAutorizado: Metodo que multiplica el *salarioCliente* por el *PORCENTAJE_BANCARIO*, y el resultado los evalua con el *limiteCreditoTipo* para poder determinar si es aprobado o no el limite de credito que se tendra en la nueva tarjeta, de ser aprobado retorna *true* de lo contrario retorna *false*.
- exitoEnAutorizacionTarjeta: Método que llama a *isLimiteCreditoAprobado* para determinar si se procede o no a la creación de una nueva tarjeta, de ser aprobado el límite entonces se procede a crear el nuevo número para la tarjeta y así hacer una instancia de *tarjeta* y retornar *true* si todo sale bien.
- crearNumeroTarjeta: Método que evalúa *tipoTarjeta* para saber cuál será el número clave con el que empezara el nuevo número de la tarjeta autorizada, y llama a *craerNumero* para que se obtenga el número de la nueva tarjeta.
- crearNumero: Método que recibe como parámetro el número clave con el que empezara el nuevo número de tarjeta. El método consulta por medio de *AutorizacionTarjetaDB* los números registrados hasta el momento dentro del sistema para de esa manera generar el numero siguiente al último registro

- **ListadoSolicitudes:** Entidad que almacena la información para generar el reporte para el listado de solicitudes que se tienen en el sistema
  - filtrarDatos: Método que genera la parte de una condición de consulta SQL en base a la validación de los diferentes atributos que se tiene para filtrar la consulta en la Base de Datos.
  - exportarReportes: Método que exporta el reporte de listado de solicitados en formato HTML guardándolo en el path de carpeta ingresado como parámetro.
  - generarArchivo: Método que verifica si existe el reporte con el mismo nombre, de ser así los elimina y actualiza con la nueva información, de lo contrario genera el encabezado de etiquetas HTML y las retorna.
  - generarContenido: Método que genera todas las etiquetas HTML con los datos para expresar el reporte de listado de solicitudes que se mostraran en la web.
  - generarHTML: Método que recibe el contenido HTML anteriormente generado y le agrega las etiquetas de cierre HTML para establecer el documento que se mostrar como reporte de listado de solicitudes.
- **ListadoTarjetas:** Entidad que almacena la información para generar el reporte para el listado de tarjetas que se tienen en el sistema
  - filtrarDatos: Método que genera la parte de una condición de consulta SQL en base a la validación de los diferentes atributos que se tiene para filtrar la consulta en la Base de Datos.
  - exportarReportes: Método que exporta el reporte de listado de tarjetas en formato HTML guardándolo en el path de carpeta ingresado como parámetro.
  - generarArchivo: Método que verifica si existe el reporte con el mismo nombre, de ser así los elimina y actualiza con la nueva información, de lo contrario genera el encabezado de etiquetas HTML y las retorna.
  - generarContenido: Método que genera todas las etiquetas HTML con los datos para expresar el reporte de listado de tarjetas que se mostraran en la web.

    - generarHTML: Método que recibe el contenido HTML anteriormente generado y le agrega las etiquetas de cierre HTML para establecer el documento que se mostrar como reporte de listado de tarjetas.
- **Consulta:** Entidad que almacena la información para generar el reporte para la consulta de la tarjeta respectiva
    - exportarReportes: Método que exporta el reporte de listado de tarjetas en formato HTML guardándolo en el path de carpeta ingresado como parámetro.
    - obtenerDataActual: Método que verifica si existe el reporte con el mismo nombre, de ser así los elimina y actualiza con la nueva información, de lo contrario genera el encabezado de etiquetas HTML y las retorna.
    - generarContenido: Método que genera todas las etiquetas HTML con los datos para expresar el reporte de listado de tarjetas que se mostraran en la web.
    - generarHTML: Método que recibe el contenido HTML anteriormente generado y le agrega las etiquetas de cierre HTML para establecer el documento que se mostrar como reporte de listado de tarjetas.
- **Tarjeta:** Entidad que tiene los atributos para representar una tarjeta.
- **Cancelación:** Entidad que tiene los atributos para representar una Cancelación para una determinada tarjeta.



- **SolicitudTarjeta:** Entidad que tiene los atributos para representa a una solicitud para una nueva tarjeta
- **Cliente:** Entidad que tiene los atributos para representar a un cliente.



- **LectorArchivo:** Entidad que maneja la lectura de los archivos de texto que se ingresan para poder alimentar a la aplicación
    - leerInstruccion: Metodo que recibe la linea leida del archivo de texto y la descompone en dos, en donde la primer parte representa la accion que se realizara y la segunda parte representa los valores o datos que se emplearan para realizar la accion respectiva.
    - obtenerDatos: Metodo que obtiene los datos o valores de una determinada accion pero de una manera en que no se puede identificar de una manera clara, en el metodo se descompone la cadena recibida como parametro para que cada valor o dato se identifique de una manera mas clara
    - ejecutarSolicitud: Metodo que verifica que los datos recibidos como parametro esten bien validados y de esa forma poder realizar la accion de hacer la Solicitud para una nueva tarjeta
    - ejecutarMovimiento: Metodo que verifica que los datos recibidos como parametro esten bien validados y de esa forma poder realizar la accion de hacer el Movimiento de una tarjeta

- ejecutarConsulta: Metodo que verifica que los datos recibidos como parametro esten bien validados y de esa forma poder realizar la accion de hacer la Consulta para una determinada tarjeta
- ejecutarAutorizacion: Metodo que verifica que los datos recibidos como parametro esten bien validados y de esa forma poder realizar la accion de hacer la Autorizacion para una nueva tarjeta
- ejecutarCancelacion: Metodo que verifica que los datos recibidos como parametro esten bien validados y de esa forma poder realizar la accion de hacer la Cancelacion de una tarjeta
- ejecutarEstadoCuenta: Metodo que verifica que los datos recibidos como parametro esten bien validados y de esa forma poder realizar la accion de hacer el reporte que contiene los estados de cuenta respectivos a las tarjetas que cumplan con cierto filtro de busqueda
- ejecutarListadoTarjetas: Metodo que verifica que los datos recibidos como parametro esten bien validados y de esa forma poder realizar la accion de hacer el reporte que contiene el listado de las tarjetas registradas en el sistema que cumplan con cierto filtro de busqueda
- ejecutarListadoSolicitudes: Metodo que verifica que los datos recibidos como parametro esten bien validados y de esa forma poder realizar la accion de hacer el reporte que contiene el listado de solicitudes registradas en el sistema que cumplan con cierto filtro de busqueda

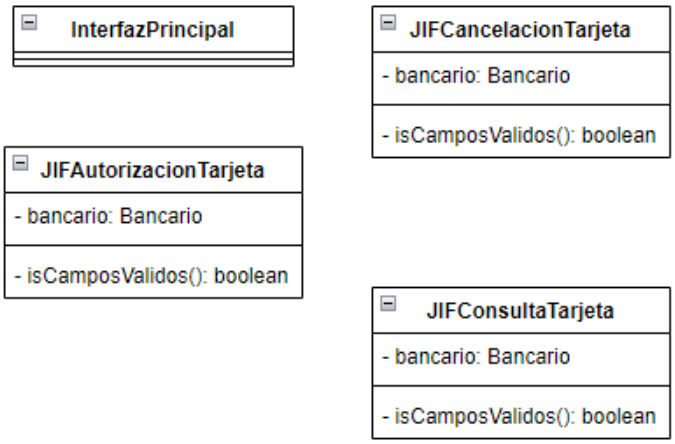
- **Bancario:** Entidad que se encarga de las verificaciones y validaciones de datos que se van ingresando para las diferentes operaciones y así poder ejecutarlas o hacerlas de manera correcta
  - isNumeroSolicitudRepetida: Método que obtienen por medio de la clase *SolicitudTarjetaDB* todas las solicitudes registradas en el sistema, y evalúa si el número de solicitud recibida como parámetro coincide con alguno de los números registrados en el sistema, de haber coincidencia devuelve *true*, de lo contrario devuelve *false.*
  - isNumeroTarjetaRepetida: Método que obtienen por medio de la clase *MovimientoTarjetaDB* todos los números de tarjetas registradas en el sistema, y evalúa si el número de tarjeta recibida como parámetro coincide con alguno de los números registrados en el sistema, de haber coincidencia devuelve *true*, de los contrario devuelve *false.*
  - quitarComillas: Método que verifica si la cadena recibida como parámetro contiene comillas, de ser así se las quita
  - isFormatoFechaCorrecto: Método que evalúa la fecha ingresada como parámetro para saber si tiene el formato de fecha de ingreso correcta, de ser así también evalúa los rangos de días y meses para garantizar coherencia, si se cumple lo anterior se retorna *true*, de los contrario se retorna *false.*
  - transformarFormatoFecha: Método que transforma la fecha ingresada como parámetro al formato en que se va a trabajar dentro de las funcionalidades como también el formato que usa la Base de Datos.

- isTipoTarjetaValido: Método que evalúa si la cadena ingresada como parámetro coincide entre los tipos de tarjetas validas dentro del sistema, de coincidir e retorna *true*, de lo contrario se retorna *false*.
- isInteger: Método que evalúa si la cadena ingresada como parámetro es válida para que sea un número entero positivo de ser así retorna *true*, de lo contrario retorna *false*.
- isDouble: Método que evalúa si la cadena ingresada como parámetro es válida para que sea un número decimal positivo de ser así retorna *true*, de lo contrario retorna *false*.
- isTipoMovimientoValido: Método que evalúa si la cadena ingresada como parámetro coincide entre los tipos de movimientos validas dentro del sistema, de coincidir se retorna *true*, de lo contrario se retorna *false*.
- isNumeroTarjetaValido: Método que verifica si la cadena ingresada como parámetro cumple con el formato correcto para considerarse un número de tarjeta valida, si se cumple con lo anterior se retorna *true*, de lo contrario se retorna *false*.
- transformarNumeroTarjeta: Método que transforma la cadena recibida como parámetro en el formato correcto para considerarse un número de tarjeta valido para los futuros procesamientos.
- isEstadoTarjetaValido: Método que evalúa si la cadena ingresada como parámetro coincide entre los estados de tarjetas validas dentro del sistema, de coincidir se retorna *true*, de lo contrario se retorna *false*.
- isEstadoSolicitudValido: Método que evalúa si la cadena ingresada como parámetro coincide entre los estados de solicitud validas dentro del sistema, de coincidir se retorna *true*, de lo contrario se retorna *false*.
- isLongitudCadenaValida: Método que verifica que la longitud de la cadena ingresada como parámetro sea menor al número recibido como parámetro
- verificarSolicitudLeida: Método que evalúa la Solicitud de Tarjeta recibida como parámetro para determinar que todos sus datos sean válidos para su posterior ejecución en el sistema
- verificarMovimientoLeido: Método que evalúa el Movimiento de Tarjeta recibida como parámetro para determinar que todos sus datos sean válidos para su posterior ejecución en el sistema
- verificarFiltroEstadoCuenta: Método que evalúa el filtro para generar el reporte con los Estados de Cuenta que se recibió como parámetro para determinar que todos sus datos sean válidos para su posterior ejecución en el sistema
- verificarFiltroListadoTarjetas: Método que evalúa el filtro para generar el reporte con el listado de Tarjetas recibida como parámetro para determinar que todos sus datos sean válidos para su posterior ejecución en el sistema
- verificarFiltroListadoSolicitudes: Método que evalúa el filtro para generar el reporte con el listado de Solicitudes recibida como parámetro para determinar que todos sus datos sean válidos para su posterior ejecución en el sistema

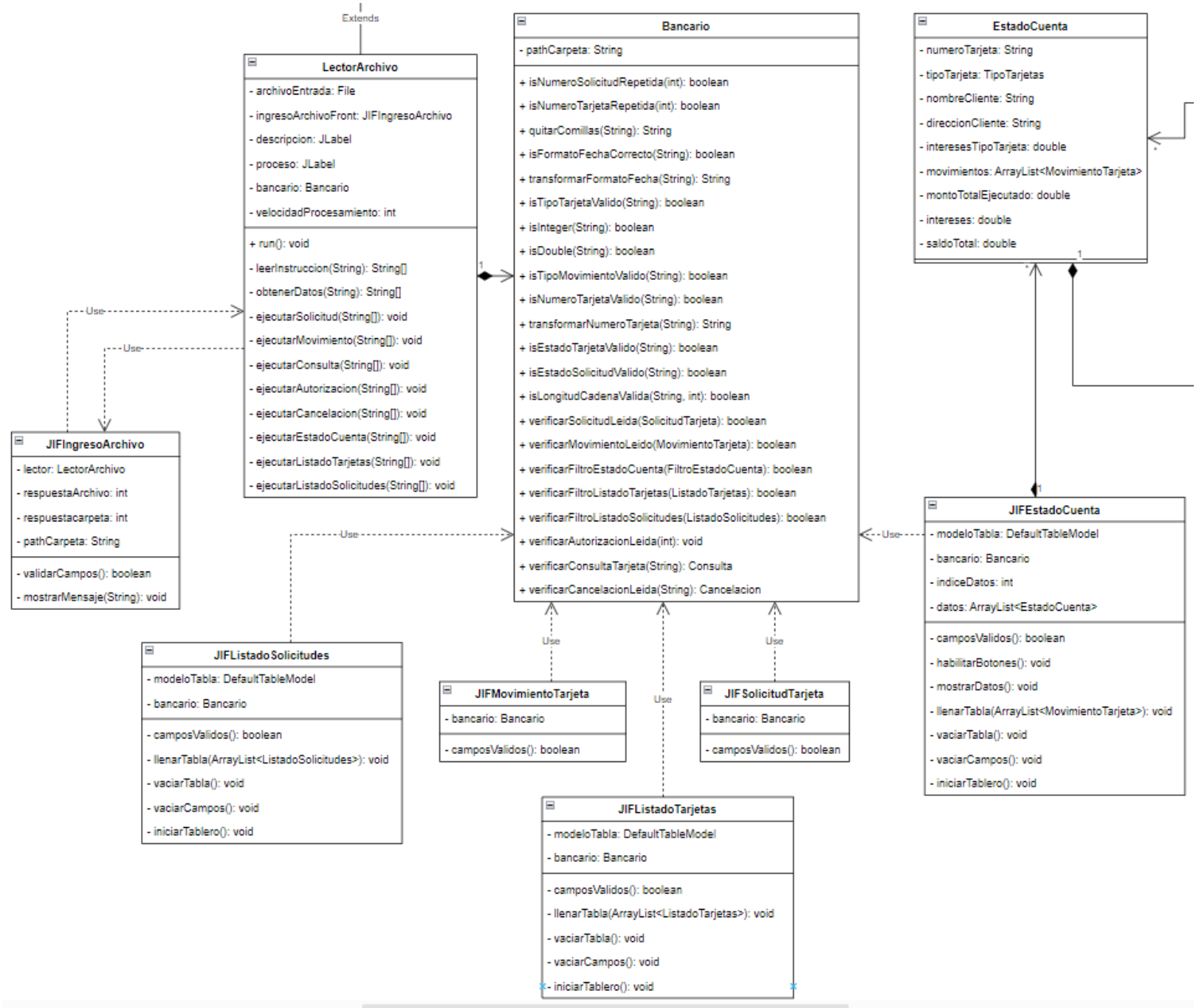
    - verificarAutorizacionLeida: Método que evalúa la Autorización de Tarjeta recibida como parámetro para determinar que sus datos sean válidos y así realizar su posterior ejecución en el sistema
    - verificarConsultaLeida: Método que evalúa la Consulta de una determinada tarjeta recibida como parámetro, y así verificar que todos sus datos sean válidos para su posterior ejecución en el sistema
    - verificarCancelacionLeida: Método que evalúa la Cancelación de una determinada tarjeta recibida como parámetro, y así verificar que todos sus datos sean válidos para su posterior ejecución en el sistema
- **EstadoCuenta:** Entidad que tiene los atributos para poder representar un estado de cuenta de una determinada tarjeta.
- **FiltroEstadoCuenta:** Entidad que almacena los datos que se utilizan para filtrar los estados de cuenta de las diferentes tarjetas registradas en el sistema
    - filtrarDatos: Método que genera la parte de una condición de consulta SQL en base a la validación de los diferentes atributos que se tiene para filtrar la consulta en la Base de Datos.
    - exportarReportes: Método que exporta el reporte de listado de tarjetas en formato HTML guardándolo en el path de carpeta ingresado como parámetro.
    - generarArchivo: Método que verifica si existe el reporte con el mismo nombre, de ser así los elimina y actualiza con la nueva información, de lo contrario genera el encabezado de etiquetas HTML y las retorna.
    - generarContenido: Método que genera todas las etiquetas HTML con los datos para expresar el reporte de listado de tarjetas que se mostraran en la web.
    - generarHTML: Método que recibe el contenido HTML anteriormente generado y le agrega las etiquetas de cierre HTML para establecer el documento que se mostrar como reporte de listado de tarjetas.
    - filtrarEstadosCuenta: Método que filtra los estados de cuenta ya obtenidos en base a los intereses si los hay o sino en base a un saldo mínimo si lo hay, para de esa forma solo mostrar o importar los estados de cuentas que cumplan con el filtro de búsqueda seleccionado.
- **MovimientoTarjeta:** Entidad que tiene los atributos para poder representar un movimiento para una determinada tarjeta.

FrontEnd

En las siguientes imágenes se detalla cada una de las entidades (clases) usadas para el funcionamiento lógico del aplicativo, en el cual cada una de ellas realiza las funciones posteriormente descritas:



- **InterfazPrincipal:** Entidad que contiene el diseño de la Interfaz Principal de Usuario para la aplicación.
- **JIFCancelacionTarjeta:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere realizar una Cancelación de una determinada Tarjeta
    - isCamposValidos: Método que valida que cada campo dentro del formulario de Interfaz para la cancelación este rellenado de forma correcta, si es de esa forma entonces se retorna *true*, de lo contrario se retorna *false*.
- **JIFAutorizacionTarjeta:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere realizar una Autorización de una determinada Tarjeta
    - isCamposValidos: Método que valida que cada campo dentro del formulario de Interfaz para la autorización este rellenado de forma correcta, si es de esa forma entonces se retorna *true*, de lo contrario se retorna *false*.
- **JIFConsultaTarjeta:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere realizar una Consulta de una determinada Tarjeta
    - isCamposValidos: Método que valida que cada campo dentro del formulario de Interfaz para la consulta este rellenado de forma correcta, si es de esa forma entonces se retorna *true*, de lo contrario se retorna *false*.



- **JIFIngresoArchivo:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere alimentar a la aplicación por medio de un archivo de entrada de texto plano
    - validarCampos: Método que valida que cada campo dentro del formulario de Interfaz para el ingreso del archivo de alimentación de texto este rellenado de forma correcta, si es de esa forma entonces se retorna *true*, de lo contrario se retorna *false*.
    - mostrarMensaje: Método que muestra un el mensaje obtenido como parámetro en una ventana emergente

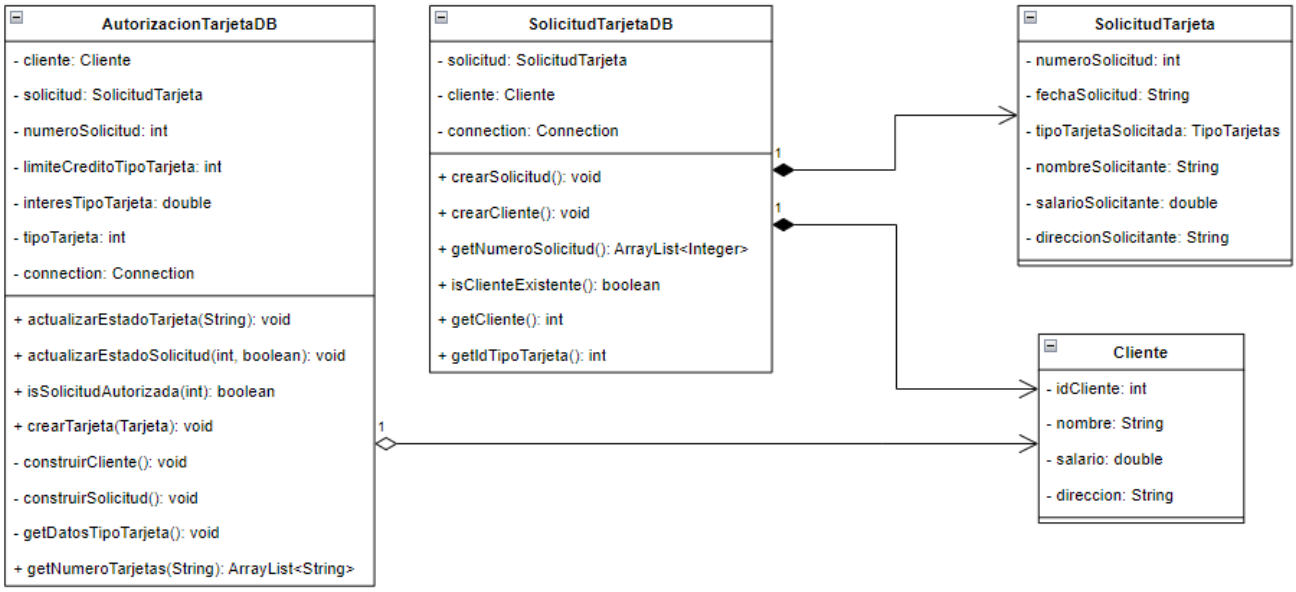
- **JIFListadoSolicitudes:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere consultar el reporte del listado de solicitudes registradas dentro del sistema
    - camposValidos: Método que valida que cada campo dentro del formulario de Interfaz para la consulta este rellenado de forma correcta, si es de esa forma entonces se retorna *true,* de lo contrario se retorna *false.*
    - llenarTabla: Método que rellena la tabla de la interfaz con los datos del arreglo que contiene todas las solicitudes que cumplen con el filtro de datos recibidos del formulario
    - vaciarTabla: Método que elimina los datos que estén existentes en la tabla de la interfaz para que no haya problemas con colapsos de datos
    - vaciarCampos: Método que limpia los campos del formulario que se utilizaron para filtrar la consulta
    - iniciarTablero: Método que le da un modelo especifico a la tabla de la interfaz para que se puedan mostrar los datos de forma correcta
- **JIFMovimientoTarjeta:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere realizar un Movimiento para una determinada Tarjeta
    - camposValidos: Método que valida que cada campo dentro del formulario de Interfaz para el movimiento de tarjeta este rellenado de forma correcta, si es de esa forma entonces se retorna *true,* de lo contrario se retorna *false.*
- **JIFListadoTarjetas:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere consultar el reporte del listado de tarjetas registradas dentro del sistema
    - camposValidos: Método que valida que cada campo dentro del formulario de Interfaz para la consulta este rellenado de forma correcta, si es de esa forma entonces se retorna *true,* de lo contrario se retorna *false.*
    - llenarTabla: Método que rellena la tabla de la interfaz con los datos del arreglo que contiene todas las tarjetas que cumplen con el filtro de datos recibidos del formulario
    - vaciarTabla: Método que elimina los datos que estén existentes en la tabla de la interfaz para que no haya problemas con colapsos de datos
    - vaciarCampos: Método que limpia los campos del formulario que se utilizaron para filtrar la consulta
    - iniciarTablero: Método que le da un modelo especifico a la tabla de la interfaz para que se puedan mostrar los datos de forma correcta
- **JIFSolicitudTarjeta:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere realizar una Solicitud para una nueva tarjeta.
    - camposValidos: Método que valida que cada campo dentro del formulario de Interfaz para la solicitud de una nueva tarjeta este rellenado de forma correcta, si es de esa forma entonces se retorna *true,* de lo contrario se retorna *false.*

- **JIFEstadoCuenta:** Entidad que contiene el diseño de la Interfaz de Usuario cuando se quiere consultar el reporte del estado de cuenta de cada tarjeta que cumpla con ciertos filtros de datos.
    - camposValidos: Método que valida que cada campo dentro del formulario de Interfaz para la consulta este rellenado de forma correcta, si es de esa forma entonces se retorna *true*, de lo contrario se retorna *false*.
    - habilitarBotones: Método que habilita el botón de Siguiente siempre que haya más de un estado de cuenta por mostrar en la interfaz de usuario
    - mostararDatos: Método que muestra los datos en los espacios correspondientes para que el usuario pueda visualizar los datos de una determinada tarjeta.
    - llenarTabla: Método que rellena la tabla de la interfaz con los datos del arreglo que contiene todos los estados de cuenta de las tarjetas que cumplen con el filtro de datos recibidos del formulario
    - vaciarTabla: Método que elimina los datos que estén existentes en la tabla de la interfaz para que no haya problemas con colapsos de datos
    - vaciarCampos: Método que limpia los campos del formulario que se utilizaron para filtrar la consulta
    - iniciarTablero: Método que le da un modelo especifico a la tabla de la interfaz para que se puedan mostrar los datos de forma correcta

Data Base

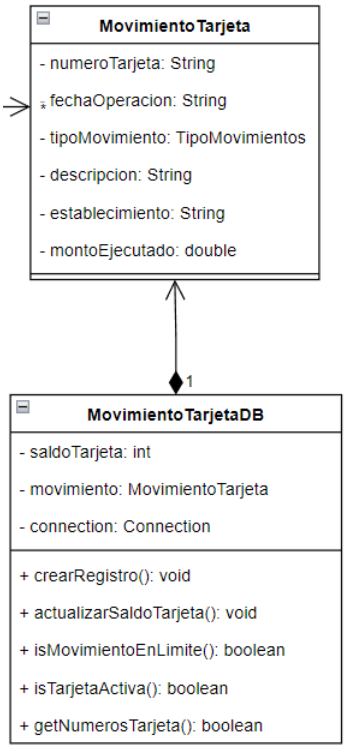
En las siguientes imágenes se detalla cada una de las entidades (clases) usadas para el funcionamiento lógico del aplicativo, en el cual cada una de ellas realiza las funciones posteriormente descritas:

- **CancelacionTarjetaDB:** Entidad que se encarga de concectarse con la Base de Datos para poder hacer las consultas y actualizacione necesarias en base a la cancelacion de una tarjeta
    - getCancelacaion: Metodo que hace un SELECT a la Base de Datos para poder obtener los datos e informacion necesaria para poder instatnciar un Objeto del Tipo Cancelacion
    - actualizarEstadoTarjeta: Metodo que hace un UPDATE a la Base de Datos para poder actualizar el estado de una determinada tarjeta.
    - getNumeroTarjetas: Metodo que hace un SELECT a la Base de Datos para poder obtener todos los numeros de tarjetas registradas en el sistema, y los retorna en una Array de *String*.
- **EstadoCuentaDB:** Entidad que se encarga de concectarse con la Base de Datos para poder hacer las consultas para poder generar los estados de cuenta por cada tarjeta
    - getEstadoCuenta: Metodo que hace un SELECT a la Base de Datos para apoder obtener el estado de cuenta por cada tarjeta que sumpla con un filtro previo de busqueda, el metodo retorna un Array de *EstadoCuenta.*
    - getMovimientoTarjeta: Metodo que hace un SELECT a la Base de Datos para poder obtener todos los movimientos que ha tenido cada tarjeta que cumpla un filtro previo de busqueda, el metodo retorna un Array de *MovimientoTarjeta*
- **ConexionDB:** Entidad que se encarga de concectarse con la Base de Datos como tal con la URL, USER y PASSWORD
- **ListadoSolicitudesDB:** Entidad que se encarga de concectarse con la Base de Datos para poder hacer las consultas para poder generar el listado de Solicitudes dentro del sistema
    - getListadoTarjetas: Metodo que hace un SELECT a la Base de Datos para poder obtener todas las solicitudes que cumpla un filtro previo de busqueda, el metodo retorna un Array de *ListadoSolicitudes*
- **ConsultaTarjetaDB:** Entidad que se encarga de concectarse con la Base de Datos para poder hacer las consultas y actualizacione necesarias en base a la cancelacion de una tarjeta
    - getConsulta: Metodo que hace un SELECT a la Base de Datos para poder obtener los datos e informacion necesaria para poder instatnciar un Objeto del Tipo Consulta
    - getNumeroTarjetas: Metodo que hace un SELECT a la Base de Datos para poder obtener todos los numeros de tarjetas registradas en el sistema, y los retorna en una Array de *String*.
- **ListadoTarjetasDB:** Entidad que se encarga de concectarse con la Base de Datos para poder hacer las consultas para poder generar el listado de Tarjetas dentro del sistema
    - getListadoTarjetas: Metodo que hace un SELECT a la Base de Datos para poder obtener todas las tarjetas que cumpla un filtro previo de busqueda, el metodo retorna un Array de *ListadoTarjetas*



- **AutorizacionTarjetaDB:** Entidad que se encarga de concectarse con la Base de Datos para poder hacer consultas, actualizacione e inserciones necesarias en base a la autorizacion de una tarjeta
    - actualizarEstadoSolicitud: Metodo que hace un UPDATE a la Base de Datos para poder actualizar el estado de una solicitud
    - isSolicitudAutorizada: Método que hace un SELECT a la Base de Datos para poder obtener el estado actual de una determinada solicitud que coincida con el número de solicitud recibida como parámetro
    - crearTarjeta: Método que hace un INSERT INTO a la Base de Datos para poder registrar una nueva tarjeta que acaba de ser autorizada.
    - construirCliente: Metodo que hace un SELECT a la Base de Datos para poder obtener los datos e informacion necesaria para poder instatnciar un Objeto del Tipo Cliente.
    - construirSolicitud: Metodo que hace un SELECT a la Base de Datos para poder obtener los datos e informacion necesaria para poder instatnciar un Objeto del Tipo Solicitud
    - getDatosTipoTarjeta: Metodo que hace un SELECT a la Base de Datos para poder obtener los datos e informacion necesaria que se vinculan al tipo de tarjeta que hay en sistema bancario.
    - getNumeroTarjetas: Metodo que hace un SELECT a la Base de Datos para poder obtener todos los numeros de tarjetas registradas en el sistema, y los retorna en una Array de *String*.
    - getNumeroTarjeta: Metodo que hace un SELECT a la Base de Datos para poder obtener el numero de tarjeta que este relacionada con el numero de solicitud recibido como parametro

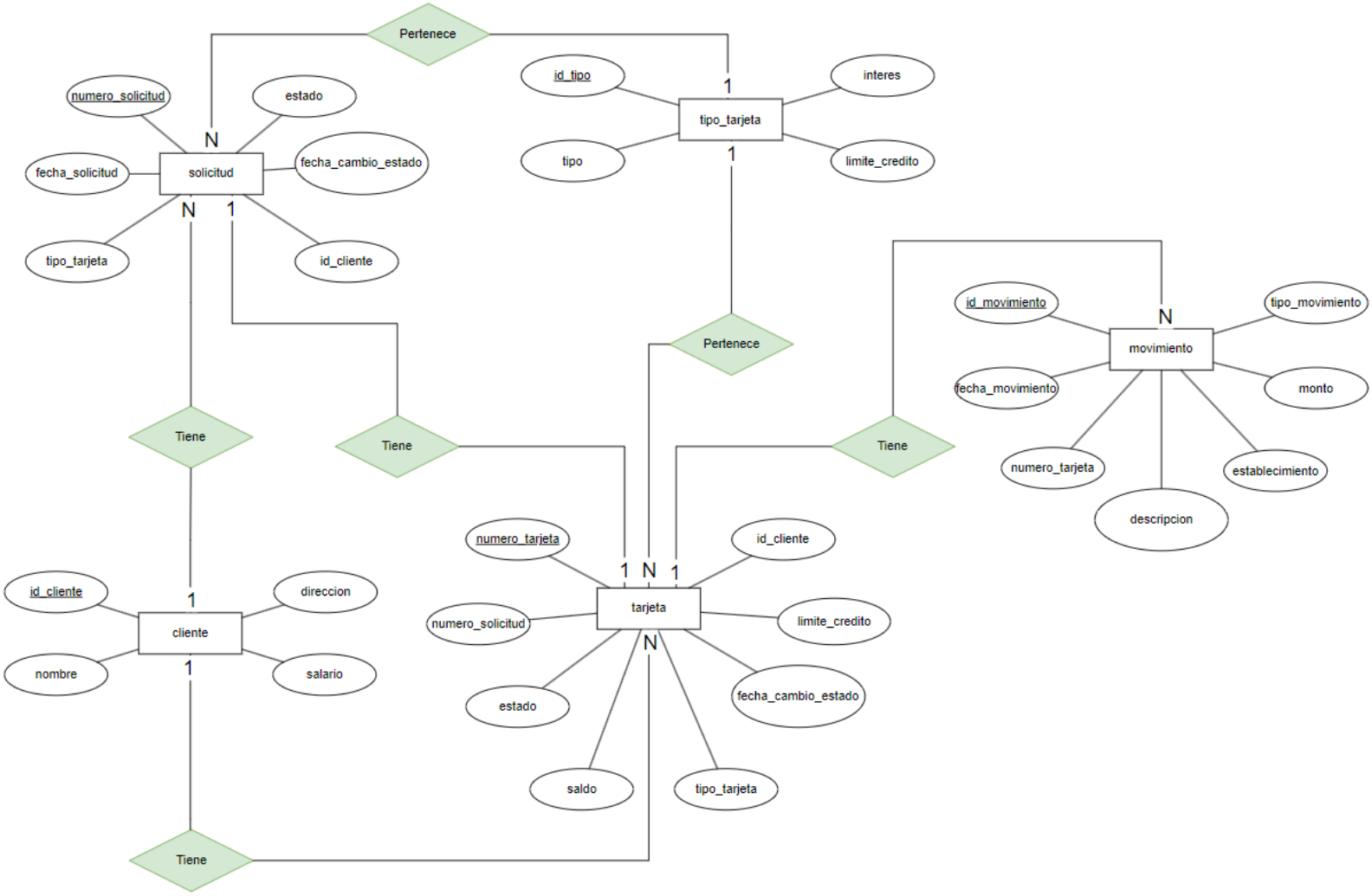
- **SolicitudTarjetaDB:** Entidad que se encarga de concectarse con la Base de Datos para poder hacer las consultas e inserciones necesarias en base a la solicitud de una nueva tarjeta
    - crearSolicitud: Método que hace un INSERT INTO a la Base de Datos para poder registrar una nueva solicitud de tarjeta
    - crearCliente: Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo cliente
    - getNumeroSolicitud: Metodo que hace un SELECT a la Base de Datos para poder obtener todos los numeros de solicitudes registradas en el sistema, y los retorna en una Array de *Integer*.
    - isClienteExistente: Método que hace un SLEECT a la Base de Datos para poder comparar datos de un supuesto cliente existente, si todo los datos son encontrados en el sistema quiere decir que el cliente ya existe y por lo tanto el método retorna *true*, de los contario se retorna *false*
    - getIdCliente: Método que hace un SELECT a la Base de Datos para poder obtener el id de un cliente en donde los datos pasados a la query coincidan con el que se encuentra registrado en sistema para de esa forma retornar el número de id.
    - getIdTipoTarjeta: Método que hace un SELECT a la Base de Datos para poder obtener el id de un determinado tipo de tarjeta en donde los datos pasados a la query coincidan con el que se encuentra registrado en sistema para de esa forma retornar el número de id.



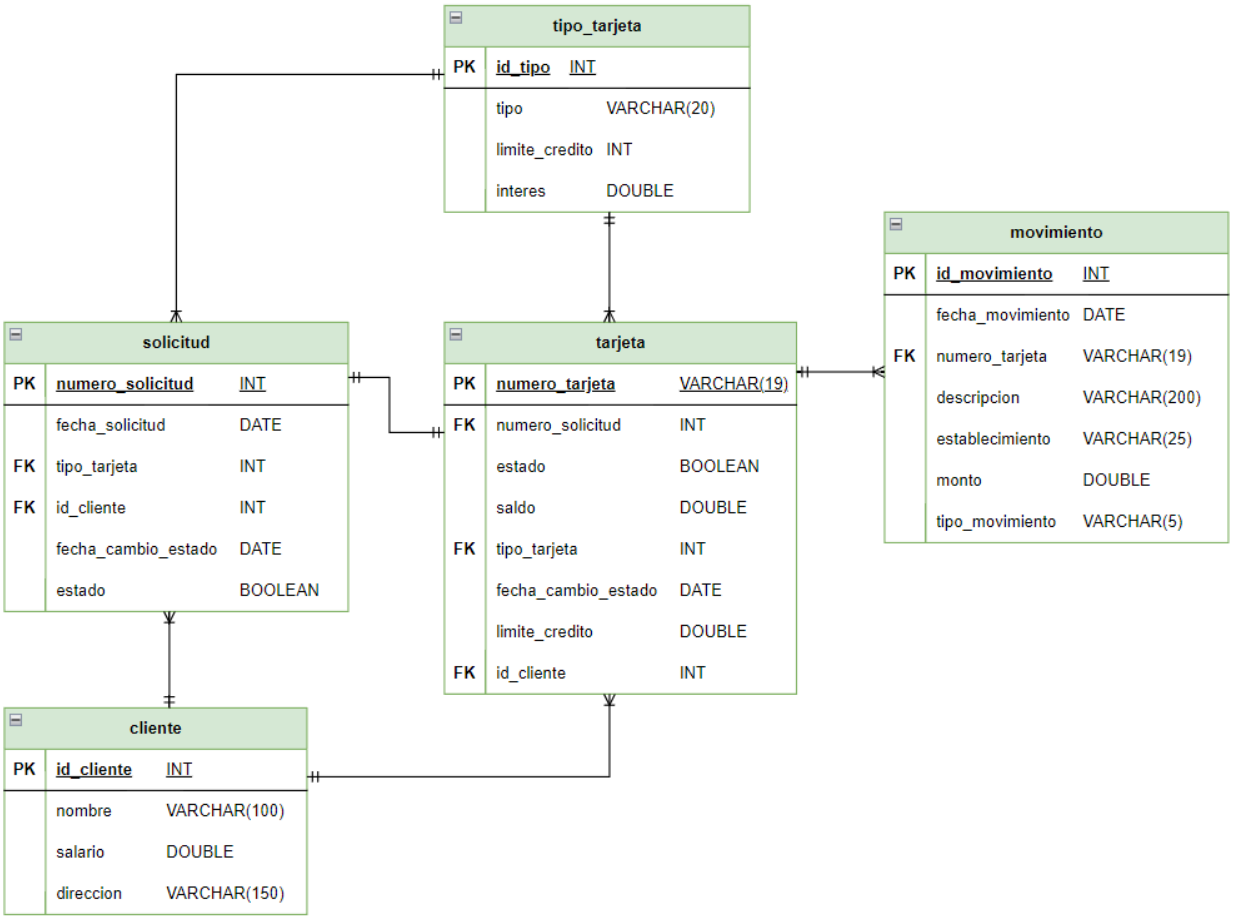
- **MovimientoTarjetaDB:** Entidad que se encarga de concectarse con la Base de Datos para poder hacer las consultas, inserciones y actualizaciones necesarias en base al movimiento de una tarjeta

- crearRegistro: Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo movimiento de una determinada tarjeta
- actualizarSaldoTarjeta: Metodo que hace un UPDATE a la Base de Datos para poder actualizar el saldo de una determinada tarjeta
- isMovimientoEnLimite: Método que hace un SELECT a la Base de Datos para obtener el saldo y el límite de crédito de una tarjeta, y evalúa si el monto ejecutado en el movimiento más el saldo actual de la tarjeta no sobrepasa el límite de crédito de la tarjeta, si la operación no sobrepasa el límite ser retorna *true*, de lo contrario se retorna *false*.
- isTarjetaActiva: Método que hace un SELECT a la Base de Datos para obtener el estado actual de la tarjeta y devolverlo
- getNumerosTarjeta: Metodo que hace un SELECT a la Base de Datos para poder obtener todos los numeros de tarjetas que estan registradas en el sistema

2. DIAGRAMA ENTIDAD RELACION E/R



3. DIAGRAMA DE TABLAS



4. MAPEO FISICO DE LA BASE DE DATOS

```
CREATE SCHEMA bd_banco;


USE bd_banco;


CREATE TABLE tipo_tarjeta (
        id_tipo INT NOT NULL AUTO_INCREMENT,
        tipo VARCHAR(20) NOT NULL,
        limite_credito INT NOT NULL,
        interes DOUBLE NOT NULL,
```

```
    CONSTRAINT PK_TIPO_TARJETA PRIMARY KEY(id_tipo)
);

CREATE TABLE cliente (
    id_cliente INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    salario DOUBLE NOT NULL,
    direccion VARCHAR(150) NOT NULL,
    CONSTRAINT PK_CLIENTE PRIMARY KEY(id_cliente)
);

CREATE TABLE solicitud (
    numero_solicitud INT NOT NULL,
    fecha_solicitud DATE NOT NULL,
    tipo_tarjeta INT NOT NULL,
    id_cliente INT NOT NULL,
    fecha_cambio_estado DATE NULL,
    estado BOOLEAN NULL,
    CONSTRAINT PK_SOLICITUD PRIMARY KEY(numero_solicitud),
    CONSTRAINT FK_TIPO_TARJETA_IN_TIPO_TARJETA FOREIGN KEY(tipo_tarjeta) REFERENCES
    tipo_tarjeta(id_tipo),
    CONSTRAINT FK_CLIENTE_IN_ID_CLIENTE FOREIGN KEY(id_cliente) REFERENCES
    cliente(id_cliente)
);

CREATE TABLE tarjeta (
    numero_tarjeta VARCHAR(19) NOT NULL,
    numero_solicitud INT NOT NULL,
    estado BOOLEAN NOT NULL,
    saldo DOUBLE NOT NULL,
```

```
    tipo_tarjeta INT NOT NULL,
    fecha_cambio_estado DATE NOT NULL,
    limite_credito DOUBLE NOT NULL,
    id_cliente INT NOT NULL,
    CONSTRAINT PK_TARJETA PRIMARY KEY(numero_tarjeta),
    CONSTRAINT FK_SOLICITUD_IN_NUMERO_SOLICITUD FOREIGN KEY(numero_solicitud)
    REFERENCES solicitud(numero_solicitud),
    CONSTRAINT FK_TIPO_TARJETA_IN_TIPO_TARJETA_IN_TARJETA FOREIGN KEY(tipo_tarjeta)
    REFERENCES tipo_tarjeta(id_tipo),
    CONSTRAINT FK_CLIENTE_IN_ID_CLIENTE_IN_TARJETA FOREIGN KEY(id_cliente) REFERENCES
    cliente(id_cliente)
);

CREATE TABLE movimiento (
    id_movimiento INT NOT NULL AUTO_INCREMENT,
    fecha_movimiento DATE NOT NULL,
    numero_tarjeta VARCHAR(19) NOT NULL,
    descripcion VARCHAR(200) NOT NULL,
    establecimiento VARCHAR(25) NOT NULL,
    monto DOUBLE NOT NULL,
    tipo_movimiento VARCHAR(5) NOT NULL,
    CONSTRAINT PK_MOVIMIENTO PRIMARY KEY(id_movimiento),
    CONSTRAINT FK_TARJETA_IN_NUMERO_TARJETA FOREIGN KEY(numero_tarjeta) REFERENCES
    tarjeta(numero_tarjeta)
);

INSERT INTO tipo_tarjeta (tipo, limite_credito, interes) VALUES ('NACIONAL', '5000', '0.012');
INSERT INTO tipo_tarjeta (tipo, limite_credito, interes) VALUES ('REGIONAL', '10000', '0.023');
INSERT INTO tipo_tarjeta (tipo, limite_credito, interes) VALUES ('INTERNACIONAL', '20000', '0.0375');
```