Documentación del Diagrama de Clases

SISTEMA BROKER

Contexto:

ARGBroker es una app desarrollada por estudiantes del primer año de la Tecnicatura Superior en Desarrollo de Software del Instituto Superior Politécnico Córdoba. Es el Proyecto Final para el módulo Programador que está compuesto por las materias: Programación I, Base de Datos y Ética y Deontología.

Objetivo:

El desarrollo de ARGBroker tiene como objetivo principal permitir que los estudiantes puedan integrar y aplicar de manera práctica los conocimientos y habilidades adquiridas en el semestre.

Cada una de ellas contribuye de manera crucial al desarrollo y éxito del proyecto, aportando un conjunto específico de competencias. En el caso de Programación I, se aprendieron conceptos básico de la programación,

como así también algunas técnicas de desarrollo de software y lógica de programación.

A través de este desarrollo, los estudiantes tiene la oportunidad de:

- Demostrar competencias y habilidades técnicas en programación.
- Trabajar en un proyecto real que simula un entorno profesional.
- Desarrollar las capacidades del trabajo colaborativo y en equipo, habilidades comunicacionales, pensamiento crítico y resolución de problemas.

ARGBroker representa una preparación esencial para el futuro desempeño profesional como Desarrolladores de Software y para enfrentar desafíos del mundo real.

Alcance del diagrama:

El diagrama de clases actual incluye las siguientes clases:

- **UsuarioLogin**: Autenticación de usuarios.
- **Inversor**: Gestión de la información del inversor.
- **Portafolio**: Gestión del portafolio de inversiones del usuario.
- **AccionComprada**: Información sobre las acciones compradas.
- **Empresa**: Información sobre las empresas cuyas acciones están disponibles.
- **Transaccion**: Información y operaciones relacionadas con las transacciones de compra y venta de acciones.

Relaciones entre clases

UsuarioLogin -> **Inversor**: Un usuario se autentica y se convierte en inversor.
Inversor -> **Portafolio**: Un inversor tiene un portafolio.
Portafolio -> **AccionComprada**: Un portafolio puede tener múltiples acciones compradas.
Inversor -> **Transaccion**: Un inversor puede realizar múltiples transacciones.
Transaccion -> **AccionComprada**: Cada transacción involucra una acción específica.
AccionComprada -> **Empresa**: Cada acción comprada está asociada a una empresa.

Documentación del Diagrama clases

1- UsuarioLogin: Para usar la app, el usuario debe ingresar al sistema informático. Este acceso se valida teniendo en cuenta si el usuario está registrado o si es nuevo. Cada usuario debe tener un nombre único que lo identifique.

- autenticarUsuario: Este un método que verifica que las credenciales del usuario sean correctas (nombre de usuario y contraseña).

- validacionCuenta: Comprueba si el nombre de usuario existe ya en el sistema.
- validacionIdentificadorFiscal: se valida el usuario con su respectivo CUIL/CUIT que es el código con el que la AFIP identifica a trabajadores autónomos, comercios y empresas en Argentina.

2- Inversor: Una vez ingresado en el sistema, el usuario toma el rol de inversor y se trae su información desde la base de datos.

- cerrarSesion: Permite al inversor cerrar completamente la sesión de la app.
- cambiarContrasenia: Permite al inversor cambiar su contraseña.
- getPortafolio: Devuelve el portafolio del inversor.
- comprar: Realiza la compra de una acción.

3- **Portafolio:** Cada usuario tiene un portafolio el cual es único, este contiene toda la información actualizada de sus acciones.

- verSaldoCuenta: Devuelve el saldo de la cuenta demo.
- actualizarGananciaPerdida: Calcula y actualiza la ganancia o pérdida del portafolio.
- verTotalInvertido: Devuelve el total invertido en el portafolio
- vender: Realiza la venta de una acción.
- verValorComprometido: Devuelve el valor comprometido en inversiones.
- getAccionComprada: Devuelve la información de una acción comprada específica.

4- **acciónComprada**: Esta clase trae la información sobre acciones compradas desde la base de datos, teniendo en cuenta la empresa a la cual fue comprada, el símbolo y el precio actual de ella.

- getPrecioActual: Devuelve el precio actual de la acción.
- getValorTotal: Calcula y devuelve el valor total de la cantidad de acciones compradas.

5- **Empresa:** Es un panel que muestra datos relacionados a todas las acciones disponibles de otras empresas.

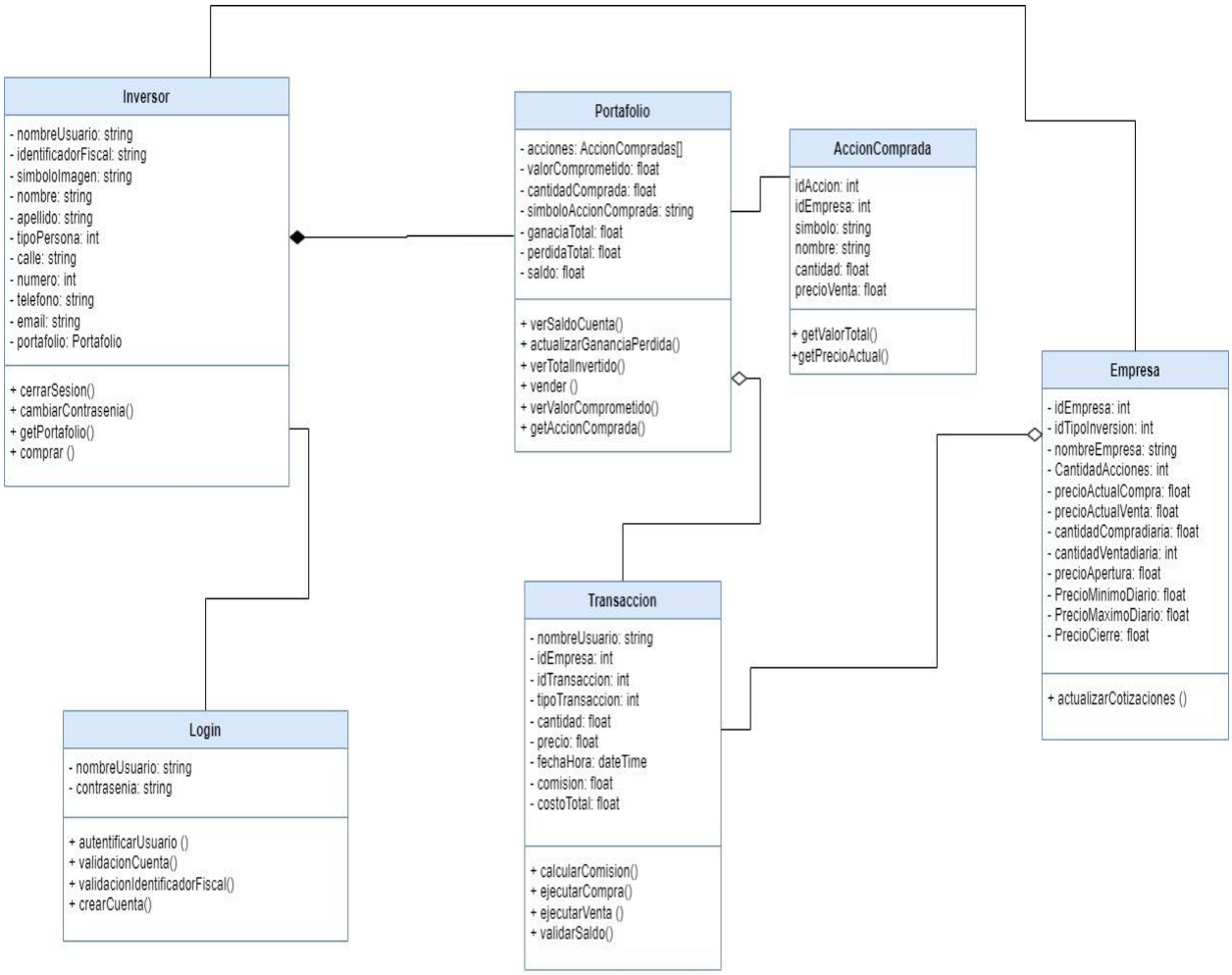
- actualizarCotizaciones: Actualiza el precio de acciones al valor del momento

4- **Transacción:** Nos muestra la operación efectuada a la hora de la compra o venta de acciones, nos brinda información respecto de la operación efectuada teniendo en cuenta el cobro de la comisión que se queda el broker.

- calcularComision: Calcula la comisión de 1,5 % sobre compra o venta de acciones
- ejecutarCompra: Realiza la compra de una acción.
- ejecutarVenta: Realiza la venta de una acción.
- validarSaldo: Corrobora que haya saldo suficiente para ejecutar la transacción.

Disidencias:

- Podría utilizarse una superclase llamada “usuario”, y de ella derivarse dos clases: “usuarioAdminstrador” y “usuarioInversor”. El usuarioInversor sería aquel que compra o vende acciones en el panel de la app mientras que usuarioAdministrador es aquel que carga las acciones en el panel.
- Se podría poner una clase que muestre o contenga el saldo (por ejemplo, el saldo inicial) y los movimientos de ese saldo con cada compra o venta (como si fuera una billetera virtual).
- Tenemos pensado en poner en la clase Transacción un atributo idTransaccion por si lo necesitamos en un futuro.


Inversor
- nombreUsuario: string
- identificadorFiscal: string
- simboloImagen: string
- nombre: string
- apellido: string
- tipoPersona: int
- calle: string
- numero: int
- telefono: string
- email: string
- portafolio: Portafolio
+ cerrarSesion()
+ cambiarContrasenia()
+ getPortafolio()
+ comprar ()
Portafolio
- acciones: AccionCompradas[]
- valorComprometido: float
- cantidadComprada: float
- simboloAccionComprada: string
- ganaciaTotal: float
- perdidaTotal: float
- saldo: float
+ verSaldoCuenta()
+ actualizarGananciaPerdida()
+ verTotalInvertido()
+ vender ()
+ verValorComprometido()
+ getAccionComprada()
AccionComprada
idAccion: int
idEmpresa: int
simbolo: string
nombre: string
cantidad: float
precioVenta: float
+ getValorTotal()
+getPrecioActual()
Empresa
- idEmpresa: int
- idTipoInversion: int
- nombreEmpresa: string
- CantidadAcciones: int
- precioActualCompra: float
- precioActualVenta: float
- cantidadCompradiaria: float
- cantidadVentadiaria: int
- precioApertura: float
- PrecioMinimoDiario: float
- PrecioMaximoDiario: float
- PrecioCierre: float
+ actualizarCotizaciones ()
Transaccion
- nombreUsuario: string
- idEmpresa: int
- idTransaccion: int
- tipoTransaccion: int
- cantidad: float
- precio: float
- fechaHora: dateTime
- comision: float
- costoTotal: float
+ calcularComision()
+ ejecutarCompra()
+ ejecutarVenta ()
+ validarSaldo()
Login
- nombreUsuario: string
- contrasenia: string
+ autentificarUsuario ()
+ validacionCuenta()
+ validacionIdentificadorFiscal()
+ crearCuenta()