Aplicación Billetera Virtual

Historial de cambios

Fecha	Versión	Descripción
03/11/2024	1.0	Documento inicial.
05/11/2024	1.1	Se agregó en el punto 3 el ítem #7 para los grupos de 3 personas.

Tabla de Contenido

1. Introducción

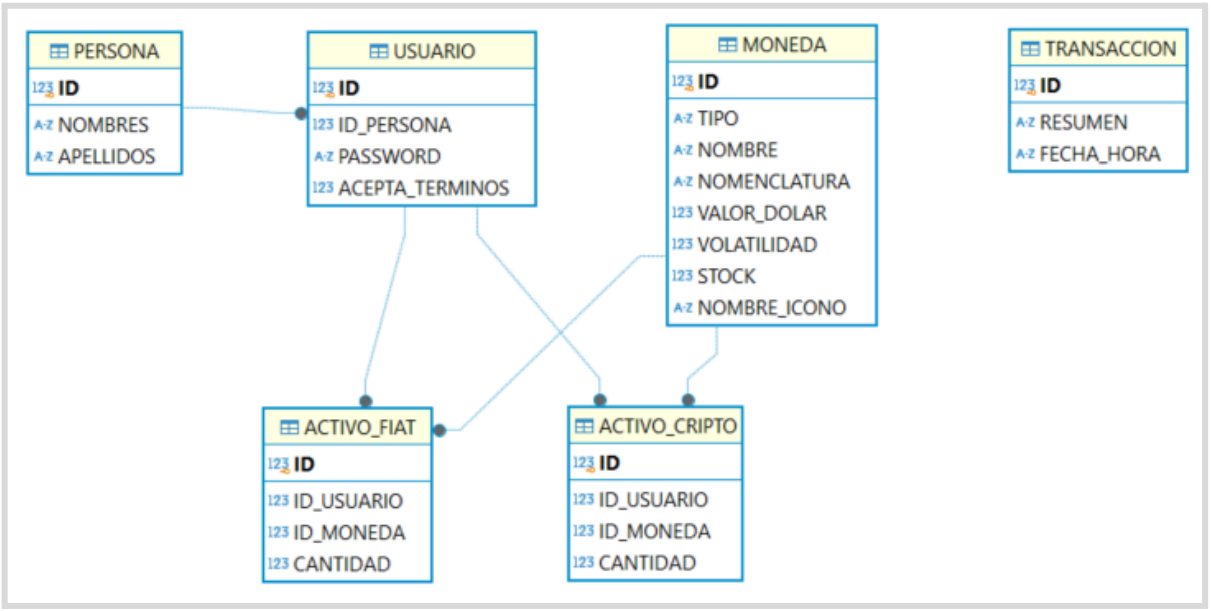
En el **Entregable 1** se trabajó en el modelo de clases de una Billetera Virtual. En el **Entregable 2** se trabajó en una POC, donde ya se podía empezar a interactuar con parte de la aplicación "de punta a punta", es decir, desde el ingreso de datos, siguiendo con la persistencia, y devolviendo un feedback al usuario, a la vez que se aplicaron algunos conceptos de patrones de diseño como el Singleton y el patrón DAO.

El objetivo del **Entregable 3** es una nueva iteración donde incorporaremos los elementos que completarán nuestra aplicación: Interfaz Gráfica (GUI), manejo de Excepciones, Entrada/Salida de archivos y manejo de Concurrencia (threads). Eventualmente ampliaremos o ajustaremos las clases del modelo y estructura de base de datos de los Entregables 1 y 2 para asociar un usuario con sus activos y nuevas funcionalidades.

2. Base de Datos

Realizaremos algunas modificaciones en la estructura de la Base de Datos. El diagrama y código que se presenta a continuación es una propuesta, que puede modificarse para adaptarlo a su proyecto. Note que no aparece la tabla BILLETERA, esto ocurre para simplificar la estructura, ya que podemos asociar los activos al usuario. En caso que prefiera modelar la BILLETERA, puede agregarla.

Diagrama Entidad-Relación (ER) de la Base de Datos:



Código SQL:

```
/**
* Este método se encarga de la creación de las tablas.
*
* @param connection objeto conexion a la base de datos SQLite
* @throws SQLException
*/
private static void creaciónDeTablasEnBD(Connection connection) throws
SQLException {
	Statement stmt;
	stmt = connection.createStatement();
	String sql = "CREATE TABLE  IF NOT EXISTS PERSONA "
			+ "("
			+ " ID	INTEGER	PRIMARY KEY AUTOINCREMENT NOT NULL ,
"
			+ " NOMBRES	VARCHAR(50)	NOT NULL, "
			+ " APELLIDOS	VARCHAR(50)	NOT NULL "
			+ ")";
	stmt.executeUpdate(sql);
	sql = "CREATE TABLE  IF NOT EXISTS USUARIO " + "(" + " ID	INTEGER
PRIMARY KEY AUTOINCREMENT NOT NULL , "
			+ " ID_PERSONA	INTEGER	NOT NULL, "
			+ " PASSWORD	VARCHAR(50)	NOT NULL, "
			+ " ACEPTA_TERMINOS	BOOLEAN	NOT NULL, "
			+ " FOREIGN KEY(ID_PERSONA) REFERENCES PERSONA(ID)"
			+ ")";
	stmt.executeUpdate(sql);

	sql = "CREATE TABLE  IF NOT EXISTS MONEDA "
			+ "("
			+ " ID	INTEGER	PRIMARY KEY AUTOINCREMENT NOT NULL ,
"
			+ " TIPO	VARCHAR(1)	NOT NULL, "
			+ " NOMBRE	VARCHAR(50)	NOT NULL, "
			+ " NOMENCLATURA VARCHAR(10)  NOT NULL, "
			+ " VALOR_DOLAR	REAL	NOT NULL, "
			+ " VOLATILIDAD	REAL	NULL, "
			+ " STOCK	REAL	NULL, "
			+ " NOMBRE_ICONO	VARCHAR(50)	NOT NULL "
			+ ")";
	stmt.executeUpdate(sql);
	sql = "CREATE TABLE  IF NOT EXISTS ACTIVO_CRIPTO"
			+ "("
			+ " ID	INTEGER	PRIMARY KEY AUTOINCREMENT NOT NULL ,
"
			+ " ID_USUARIO INTEGER	NOT NULL, "
```

```
                    + " ID_MONEDA INTEGER    NOT NULL, "
                    + " CANTIDAD REAL    NOT NULL, "
                    + " FOREIGN KEY(ID_USUARIO) REFERENCES USUARIO(ID),"
                    + " FOREIGN KEY(ID_MONEDA) REFERENCES MONEDA(ID) "
                    + ")";
        stmt.executeUpdate(sql);
        sql = "CREATE TABLE  IF NOT EXISTS ACTIVO_FIAT"
                    + "("
                    + " ID       INTEGER   PRIMARY KEY AUTOINCREMENT NOT NULL ,
"
                    + " ID_USUARIO INTEGER    NOT NULL, "
                    + " ID_MONEDA INTEGER    NOT NULL, "
                    + " CANTIDAD REAL    NOT NULL, "
                    + " FOREIGN KEY(ID_USUARIO) REFERENCES USUARIO(ID),"
                    + " FOREIGN KEY(ID_MONEDA) REFERENCES MONEDA(ID)"
                    + ")";
        stmt.executeUpdate(sql);
        sql = "CREATE TABLE  IF NOT EXISTS TRANSACCION"
                    + "("
                    + " ID     INTEGER   PRIMARY KEY AUTOINCREMENT NOT NULL , "
                    + " RESUMEN VARCHAR(1000)   NOT NULL, "
                    + " FECHA_HORA            DATETIME  NOT NULL "
                    + " ID_USUARIO INTEGER    NOT NULL, "
                    + " FOREIGN KEY(ID_USUARIO) REFERENCES USUARIO(ID)"
                    + ")";
        stmt.executeUpdate(sql);

        stmt.close();
}
```

3. Funcionalidades del Sistema

En líneas generales estas son las funcionalidades principales. El detalle se brinda en la sección 4. Prototipo:

1. **Login de usuario** usando simplemente su e-mail y password.
2. **Registración de un usuario.** Los datos solicitados son los mínimos necesarios. No se contempla verificaciones a excepción de la cuenta de email, que no puede estar asociada a otro usuario, y la aceptación de términos y condiciones.
3. **Visualización del Balance y Mis Activos,** con posibilidad de exportación en archivo con formato “valores separado por coma” (CSV).
4. **Visualización de las Transacciones** realizadas en el sistema.

5. **Visualización de Cotizaciones de las Criptomonedas**, cuyo valor se actualiza en segundo plano.
6. **Compra de Criptomonedas.**
7. **Swap de Criptomonedas** (adicional para grupos de 3 personas)

4. Prototipo

Aquí tiene disponible un ejemplo de un boceto de la aplicación en la que es posible observar el flujo entre las diferentes pantallas. Tenga en cuenta que a excepción de lo remarcado como comentarios, ud. puede realizar ajustes, siempre que amplíen y no reduzcan el alcance de la entrega. Lea todos los comentarios incluidos en el diagrama.
Prototipo de la aplicación

5. De la Implementación del Prototipo

A. Se espera que ud. implemente una aplicación siguiendo el patrón MVC.
B. Debe crear 3 excepciones propias y manejarlas dentro de su sistema.
C. Puede implementar cualquiera de las opciones vistas en clase para manejo de concurrencia.
D. Respecto de la obtención de la cotización actual de las monedas, puede adaptar el siguiente código. Tenga en cuenta que para usarlo necesitará incorporar la librería JSON a su aplicación.

   IMPORTANTE: la obtención de estas cotizaciones no puede bloquear el funcionamiento de la aplicación.

```
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import org.json.JSONObject; // Necesita agregar la librería org.json para
trabajar con JSON
public class ConsultarPrecioCripto {
   private static final String URL_API =
"https://api.coingecko.com/api/v3/simple/price?ids=bitcoin,ethereum,usd-coin,te
ther,dogecoin&vs_currencies=usd";
   public static void main(String[] args) {
       HttpClient cliente = HttpClient.newHttpClient();
       HttpRequest solicitud = HttpRequest.newBuilder()
```

```
                .uri(URI.create(URL_API))
                .GET()
                .build();
        try {
            HttpResponse<String> respuesta = cliente.send(solicitud,
HttpResponse.BodyHandlers.ofString());
            if (respuesta.statusCode() == 200) {
                parsearYMostrarPrecios(respuesta.body());
            } else {
                System.out.println("Error: " + respuesta.statusCode());
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
    private static void parsearYMostrarPrecios(String cuerpoRespuesta) {
        JSONObject json = new JSONObject(cuerpoRespuesta);
        System.out.println("Precios de Criptomonedas (en USD):");
        double precioBTC = json.getJSONObject("bitcoin").getDouble("usd");
        System.out.println("BTC: $" + precioBTC);
        double precioETH = json.getJSONObject("ethereum").getDouble("usd");
        System.out.println("ETH: $" + precioETH);
        double precioUSDC = json.getJSONObject("usd-coin").getDouble("usd");
        System.out.println("USDC: $" + precioUSDC);
        double precioUSDT = json.getJSONObject("tether").getDouble("usd");
        System.out.println("USDT: $" + precioUSDT);
        double precioDOGE = json.getJSONObject("dogecoin").getDouble("usd");
        System.out.println("DOGE: $" + precioDOGE);
    }
}
```

6. Contenido de la Entrega

La entrega incluye un archivo ZIP que contiene lo siguiente:

A. Un archivo .zip con el proyecto (código fuente)
B. Un archivo .jar ejecutable con la aplicación
C. Un archivo readme con cualquier particularidad o información que considere necesario tener en cuenta para ejecutar su aplicación ó respecto de la implementación.