PROYECTO DE CONSTRUCCIÓN Y EVOLUCIÓN DE SOFTWARE (ISWD633)

TEMA: ESTANDARES

GRUPO: CODE FORGE

Integrantes:

- **ANGEL VLADIMIR CABEZAS JACOME**
- **WILLIAM ESTEBAN ENRIQUEZ RECALDE**
- **ATIK AMILCAR TUQUERREZ FLORES**
- **DARIO ANDRES PALMA MERA**

Carrera: Ingeniería de Software

Grupo: GR2SW

Fecha de entrega: 01/12/2024

Paquetes:

- BD para la interacción con la base de datos.
- Modelo para las clases que representan la lógica y entidades del sistema.
- Controlador para la gestión de la interacción entre las vistas y el modelo.
- Vista para las interfaces gráficas.

- **Clases:**
  - Los nombres de las clases (Accion, AccionBD, Conexion) cumplen con la convención PascalCase.
  - Sin embargo, el nombre BD podría reemplazarse por algo más descriptivo, como repository o data.

- Métodos:
  - Siguen el formato camelCase y son descriptivos. Ejemplo: registrarCompra, mostrarCompras.

- Variables:
  - Las variables cumplen la convención de camelCase, pero algunas carecen de suficiente descriptividad. Por ejemplo, podría ser conexion.

Estructura del Código

Uso de Capas

- Modelo:
  - Las clases Accion y Usuario están bien estructuradas y encapsulan datos y validaciones.
  - Sin embargo, los métodos de validación como esFechaValida podrían moverse a una

clase utilitaria para un mejor desacoplamiento.

- Datos:
  - Las clases AccionBD y UsuarioBD manejan las operaciones con la base de datos, pero se mezclan con lógica que debería estar en la capa de servicio o controlador.

- Interfaz Gráfica:
  - El controlador UsuarioController conecta correctamente las vistas y modelos, pero podría beneficiarse de un desacoplamiento mayor entre las capas.

Manejo de Errores

- Actualmente, el manejo de errores es limitado y dependiente de JOptionPane. Esto puede dificultar la reutilización del código en entornos que no sean GUI.

Uso de Spring Boot

El proyecto actualmente no usa Spring Boot, pero sería ideal migrar a este framework para mejorar.

Funcionalidades y Validación

Validación de Datos

- Fortalezas:
  - Las validaciones en la clase Accion son robustas, verificando fechas, valores y cantidades.
  - El método gananciaPerdidaPorcentaje es útil y bien diseñado.
- Áreas de Mejora:
  - Implementar más validaciones en el lado del servidor para evitar errores en la base de datos.
  - Validar datos en la interfaz gráfica antes de enviarlos.

Integración con API Externa

- La clase AccionAPI utiliza correctamente la API de Alpha Vantage para obtener precios y verificar empresas.
- Mejoras Sugeridas
  - Manejar errores de red con mensajes más descriptivos para los usuarios.
  - Cachear respuestas frecuentes para reducir la dependencia de la API en tiempo real.

Base de Datos

- Esquema:
  - La base de datos está bien estructurada, con claves primarias y foráneas.
  - Se utiliza el tipo de dato adecuado para cada columna.

Interfaz de Usuario

- La interfaz parece funcional, pero podría beneficiarse de mejoras:
  - Uso de **Swing** limita la experiencia de usuario. Considerar una migración a **JavaFX** o una interfaz web.
  - Agregar validaciones visuales en los formularios (p. ej., marcar campos obligatorios).