Los patrones MVC (Modelo-Vista-Controlador) y DAO (Objeto de Acceso a Datos) son fundamentales en el desarrollo de sistemas que involucran una base de datos, como lo fue el caso de este sistema. Los patrones ofrecen una serie de beneficios y ventajas que mejoran la estructura y mantenibilidad del programa. Dichos patrones fueron importantes y relevantes en el desarrollo de este programa que represento una especie de banco con sus respectivos clientes y cuentas. Ambos patrones contribuyen a una estructura más sólida y mantenible del programa.

Importancia del patrón MVC:

- Separación de responsabilidades: Divide la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador. Esto permite asignar responsabilidades específicas a cada componente, lo que facilita el desarrollo y el mantenimiento del código. El Modelo (en este caso Client y Account) se encarga de manejar los datos y la lógica de negocio, la Vista se ocupa de la presentación de la información al usuario, y el Controlador gestiona las interacciones y eventos del usuario.
- Modularidad y reutilización de código: Al separar claramente las diferentes capas del sistema, el patrón MVC promueve la modularidad y la reutilización de código. Esto facilita la extensibilidad del sistema, ya que cada componente puede ser desarrollado y probado de forma independiente. Asimismo, al utilizar interfaces entre los componentes, se logra un acoplamiento bajo, lo que favorece la flexibilidad y el mantenimiento a largo plazo.
- Mejora de la interfaz de usuario: Permite una gestión más eficiente de la interfaz de usuario. La separación del controlador y la vista permite cambios en la presentación sin afectar la lógica subyacente, lo que facilita la evolución de la interfaz de usuario de forma independiente. De igual manera, al utilizar controladores para manejar las interacciones del usuario, se puede ofrecer una experiencia de usuario más interactiva y receptiva.

Importancia del patrón DAO:

- Abstracción de la capa de acceso a datos: Proporciona una capa de abstracción entre la lógica de negocio y la base de datos. Esto permite desacoplar la lógica de negocio de los detalles de implementación de la base de datos, lo que facilita la portabilidad y el mantenimiento del código. También, proporciona una interfaz clara y consistente para acceder y manipular los datos, lo que mejora la legibilidad y entendimiento del código. Solo el DAO puede acceder y tener una conexión a la base de datos.
- Mejor rendimiento y escalabilidad: Se pueden implementar estrategias eficientes de acceso a datos, como el uso de caché, la optimización de consultas y la gestión de transacciones. Estas técnicas pueden mejorar el rendimiento de la aplicación y permitir una mejor escalabilidad, especialmente cuando se trata de grandes volúmenes de datos. De igual forma, facilita la gestión de conexiones a la base de datos puesto que todo lo que tiene que ver con la base de datos se encuentra en una misma y no más, evitando la creación innecesaria de conexiones y optimizando los recursos.
- Facilidad de mantenimiento y pruebas: Al tener una capa de acceso a datos separada, se simplifica el mantenimiento y las pruebas del código. Los cambios en la estructura o tecnología de la base de datos se pueden abordar en el DAO sin afectar la lógica de negocio. Además, al utilizar interfaces para definir las operaciones de acceso a datos, se puede realizar pruebas unitarias de forma aislada, sin depender de una base de datos real.