PATRÓN DE DISEÑO

PROXY

CONTENIDOS

ÍNDICE

¿EN QUE CONSISTE?

El patrón de diseño estructural proxy, también conocido como surrogate (Sustituto o reemplazo).

Es un patrón el cual proporciona un sustituto o marcador de posición para otro objeto para controlar el acceso a él.

El cliente interactúa con el proxy como si fuera el objeto real, sin darse cuenta de que está usando un representante.

MOTIVACIÓN

Muchas veces, el objeto que queremos usar puede ser costoso de crear o manipular. Por ejemplo:

- Un documento con muchas imágenes grandes que tardan en cargar.
- Acceder a recursos en otra máquina (acceso remoto).
- Controlar quién puede hacer qué en un sistema (seguridad).

SOLUCIÓN

Si cargamos todos los objetos caros desde el inicio, la aplicación será lenta e ineficiente. Pero tampoco podemos ignorar esos objetos porque el usuario puede necesitarlos en cualquier momento.

El patrón Proxy resuelve esto permitiendo crear o acceder al objeto real solo cuando sea necesario, sin complicar el código cliente. Esto también ayuda a proteger el objeto real y manejar accesos remotos de forma transparente.

COMPONENTES PRINCIPALES

SUBJECT (SUJETO)

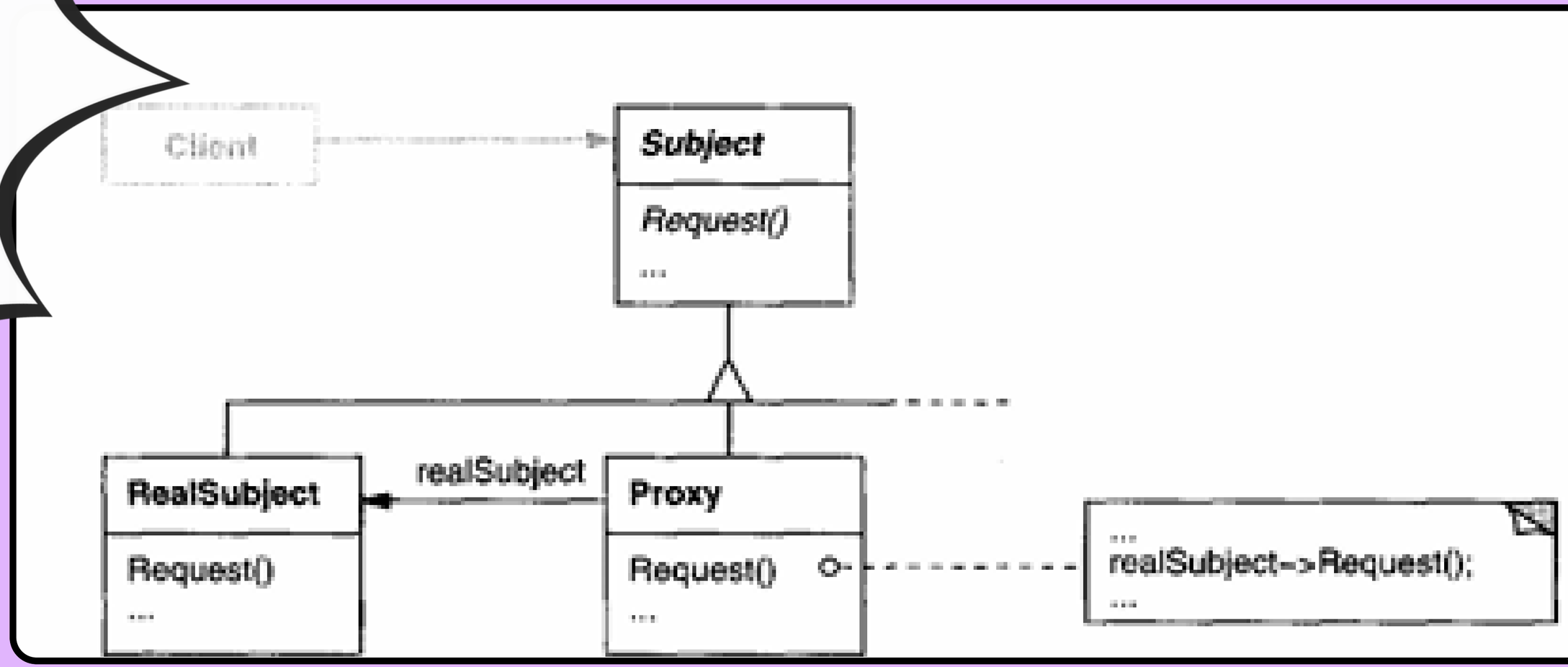
Una interfaz común que define las operaciones que pueden realizar tanto el objeto real como el proxy. Esto garantiza que el cliente pueda usar cualquiera sin diferenciar.

REALSUBJECT (OBJETO REAL)

El objeto que contiene la funcionalidad real, puede ser costoso de crear o acceder.

PROXY

El objeto sustituto que controla el acceso al RealSubject. Puede retrasar la creación del objeto real, verificar permisos, realizar caching, o gestionar acceso remoto.


Client
Subject
Request()
...
RealSubject
Request()
...
realSubject
Proxy
Request()
...
...
realSubject->Request();
...

TIPOS COMUNES DE PROXY

PROXY REMOTO

Proporciona un representante local para un objeto que reside en otra dirección de memoria o incluso en otra máquina.

PROXY VIRTUAL

Retrasa la creación de objetos costosos hasta que son realmente necesarios.

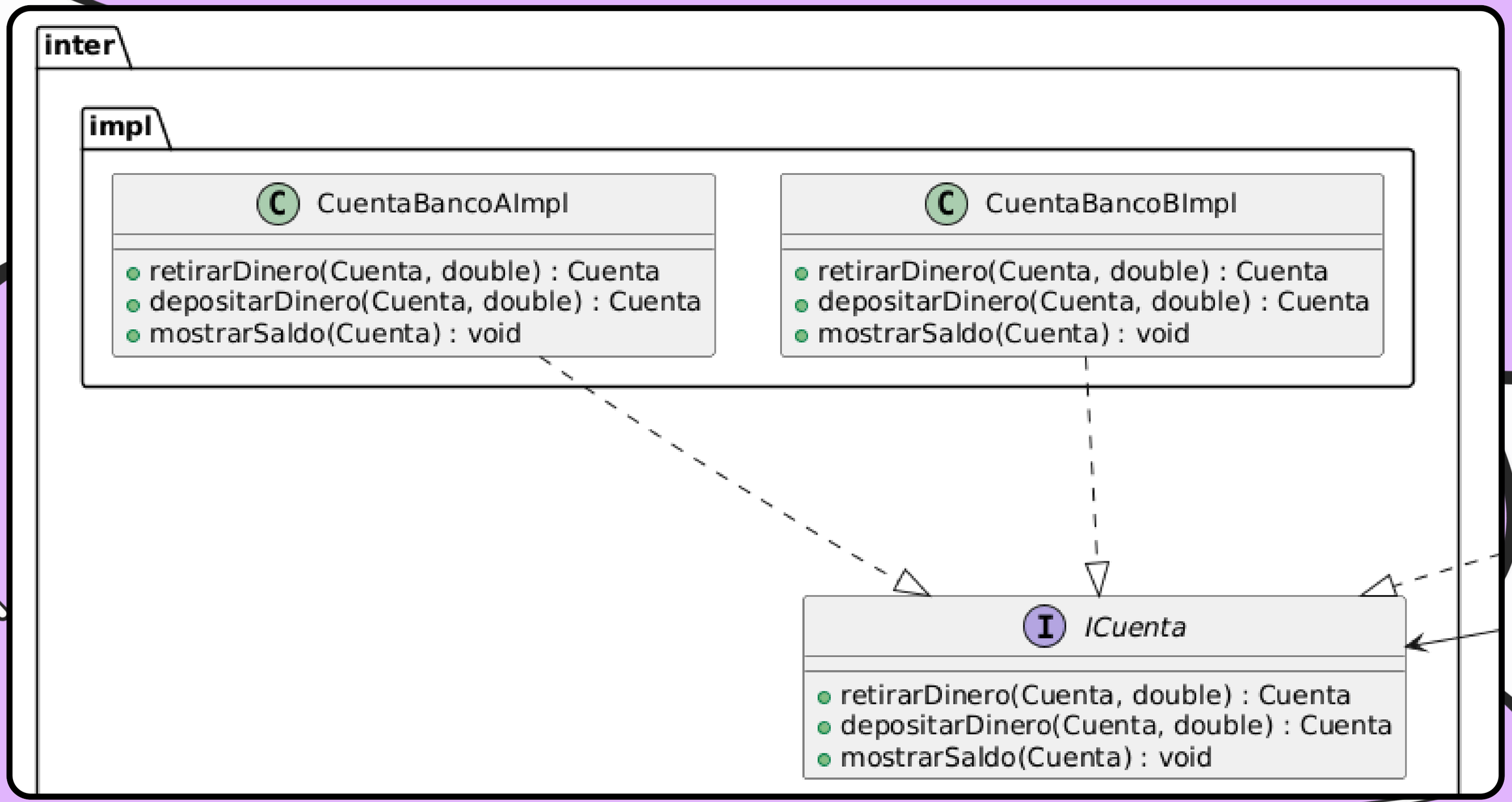
PROXY DE PROTECCIÓN

Controla el acceso al objeto real, permitiendo o denegando el acceso basado en permisos, roles, o condiciones específicas.

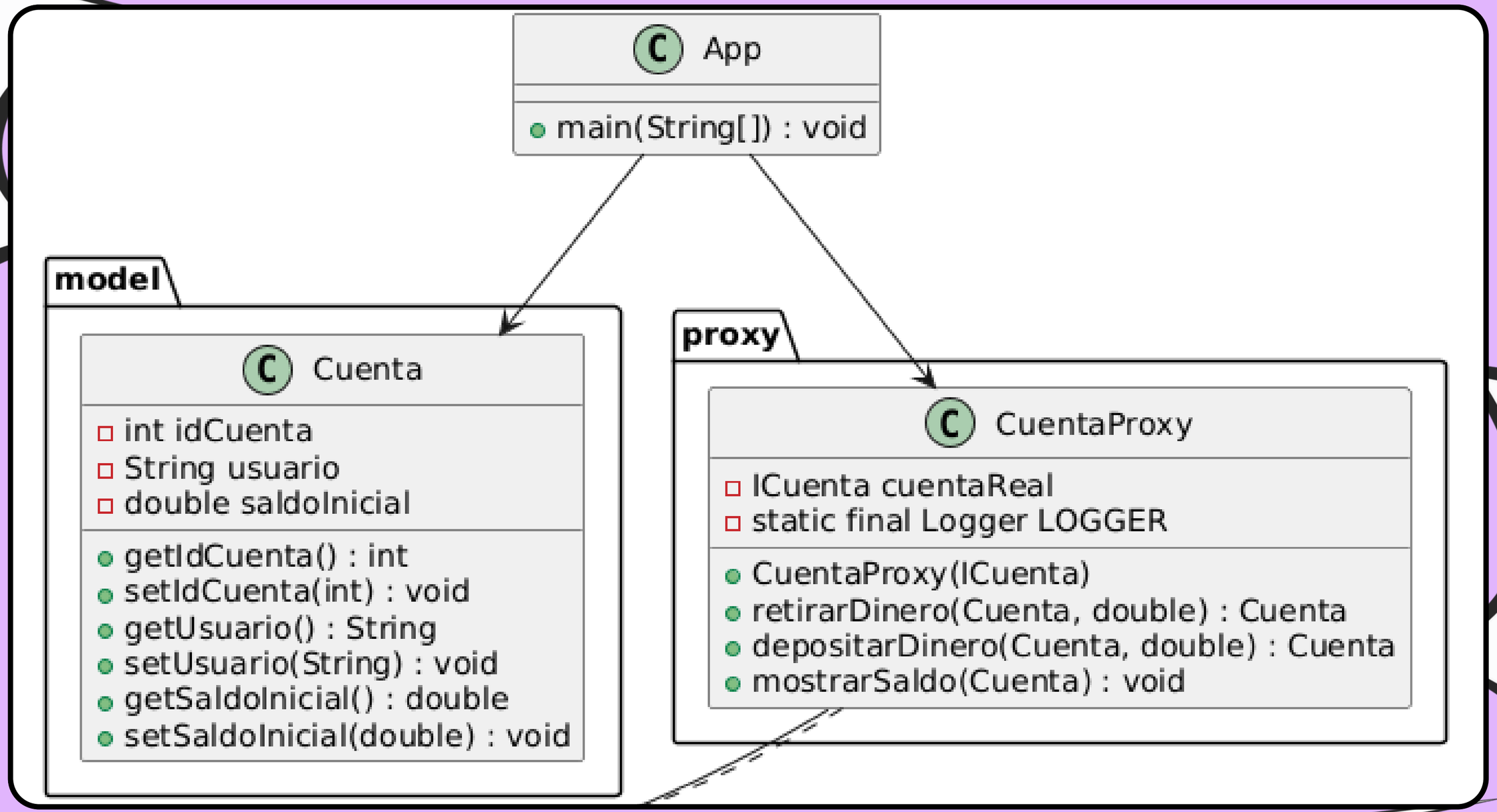
SMART PROXY

Realiza operaciones adicionales cuando se accede al objeto real.

MI IMPLEMENTACIÓN



MI IMPLEMENTACIÓN



VENTAJAS

- **Control de acceso:** Protege objetos reales con validaciones o permisos.
- **Optimización de recursos**: Retrasa creación o carga de objetos pesados solo cuando se usan.
- **Transparencia**: El cliente no necesita cambiar su código, solo usa la interfaz común.
- **Flexibilidad**: Se puede extender para añadir funcionalidades como caching o logging.

DESVENTAJAS

- **Complejidad adicional:** Se agrega una capa extra en la comunicación, lo que puede complicar el diseño y mantenimiento.
- **Posible impacto en rendimiento:** Si el proxy realiza muchas operaciones extra, puede hacer que la respuesta sea más lenta.
- **Difícil depuración:** La presencia del proxy puede ocultar la interacción real entre el cliente y el objeto, dificultando rastrear errores.

CONCLUSIÓN

El patrón Proxy es una herramienta poderosa para controlar el acceso y optimizar el uso de objetos en sistemas complejos. Su principal ventaja es permitir que el cliente interactúe con objetos de forma transparente, mientras el proxy decide cuándo y cómo crear o acceder al objeto real.

Este patrón es muy utilizado en frameworks y arquitecturas distribuidas, optimización de recursos y sistemas con altos requerimientos de seguridad.

¡GRACIAS POR LA ATENCIÓN!