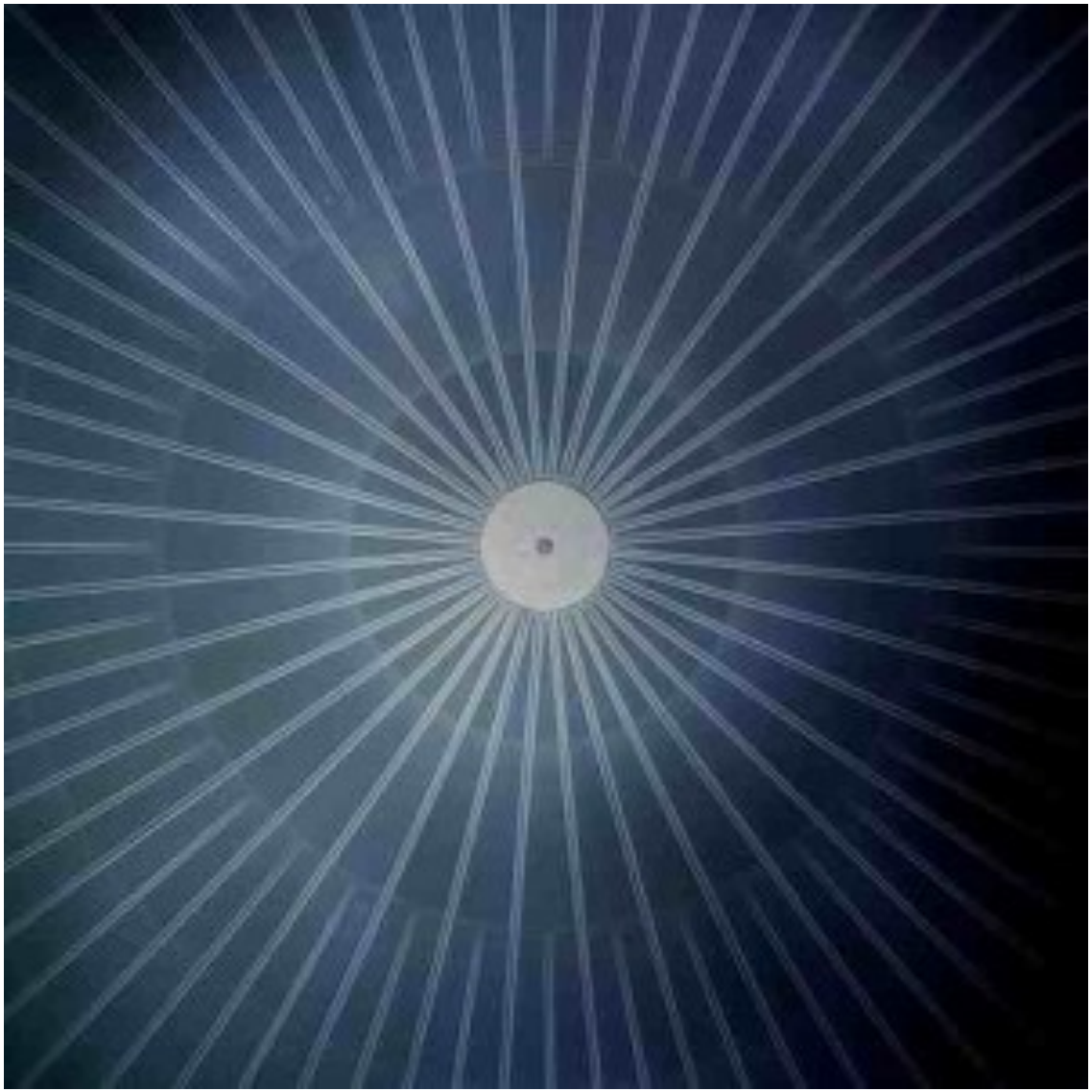
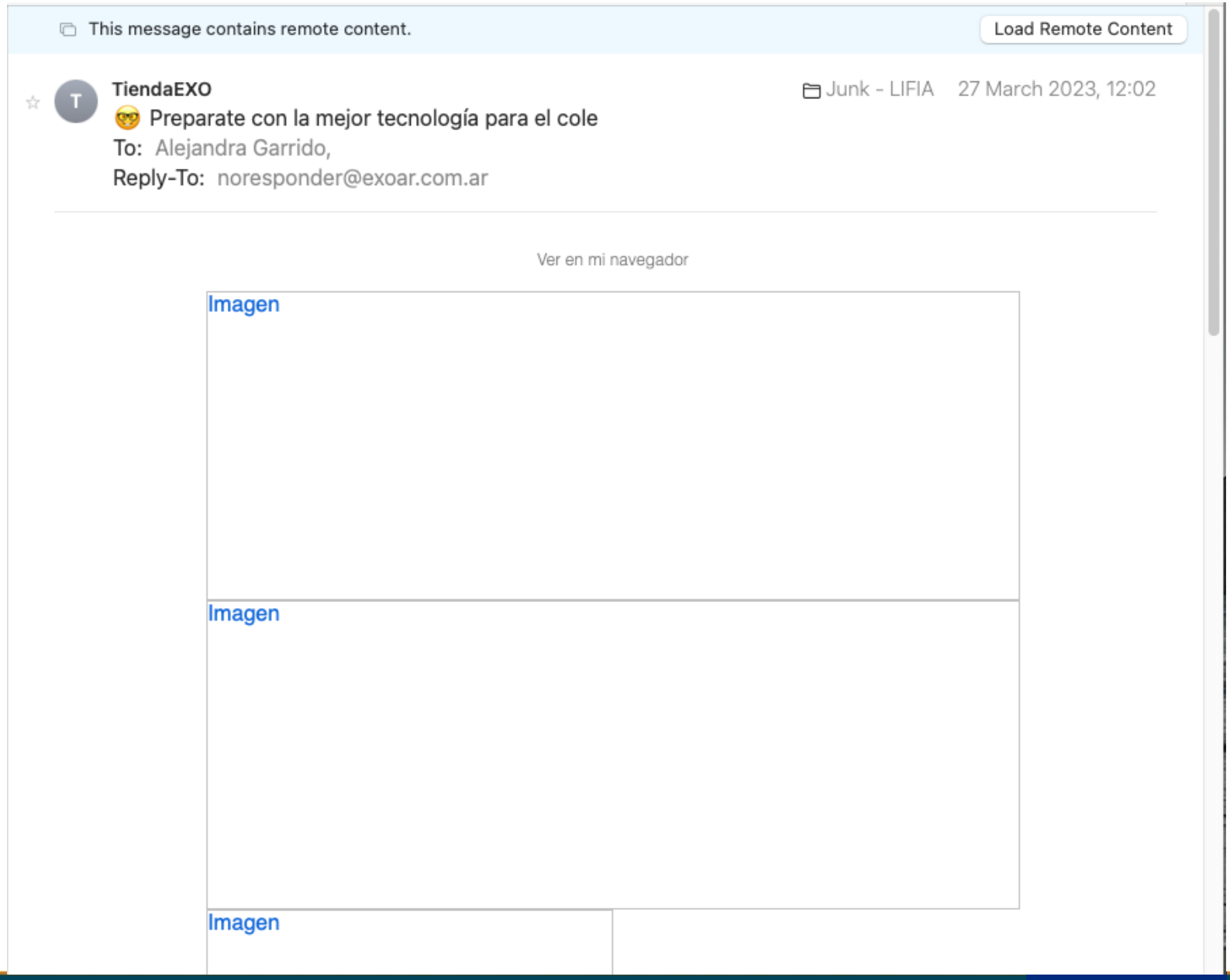
2do nuevo patrón



Ejemplo


This message contains remote content.
Load Remote Content
T
TiendaEXO
Junk - LIFIA
27 March 2023, 12:02
Preparate con la mejor tecnología para el cole
To: Alejandra Garrido,
Reply-To: noresponder@exoar.com.ar
Ver en mi navegador
Imagen
Imagen
Imagen

Carga bajo demanda – Fuerzas del problema

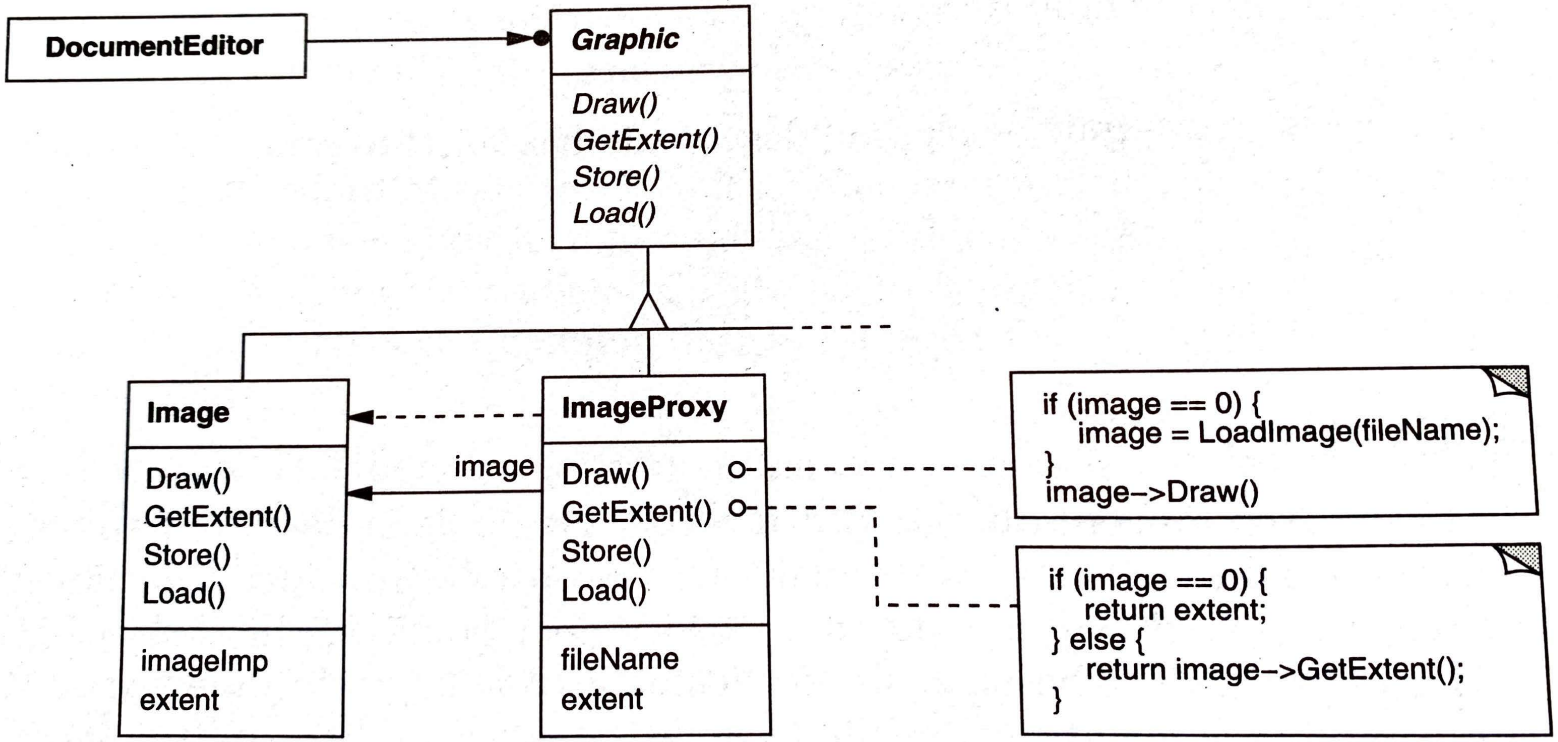
- En muchos casos un email puede tener muchas imágenes, siendo estas pesadas y lentas de cargar
- No queremos que la apertura de un email sea lenta.
- En algunos casos las imágenes ni siquiera serán vistas.
- Queremos evitar el costo de leer la imagen hasta tanto sea necesario mostrarla
- Igualmente necesitamos un « representante » de la imagen, de manera de darle al cliente un objeto que se vea y actúe como el cliente espera

Carga bajo demanda: Solución

- La idea es crear una imagen “falsa”, un impostor que
  - Debe responder a los mensajes de la imagen verdadera (mantiene el protocolo).
  - Sabe responder a algunos mensajes (tamaño de la imagen)
  - Cuando sea necesario mostrarla en pantalla, debe ir a buscar la imagen original al servidor, leerla y mostrarla.

Solución

- Cargar las imágenes bajo demanda, utilizando un objeto *proxy*. El proxy se comporta como una imagen normal y es el responsable de cargar la imagen bajo demanda



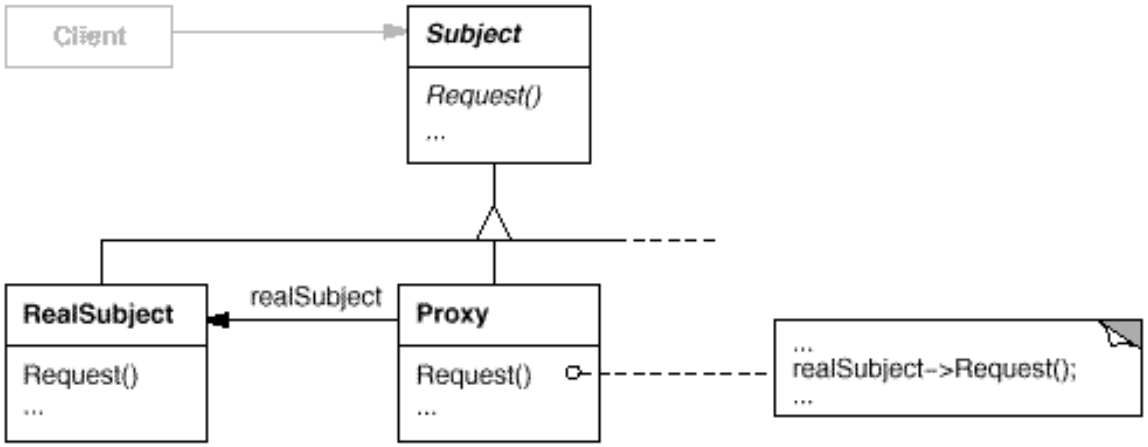
Patrón Proxy

- **Propósito:** proporcionar un intermediario de un objeto para controlar su acceso.

- **Aplicabilidad**: cuando se necesita una referencia a un objeto más flexible o sofisticada

Patrón *Proxy*. Solución

- Colocar un objeto intermedio que respete el protocolo del objeto que está reemplazando.
- Algunos mensajes se delegarán en el objeto original. En otros casos puede que el proxy colabore con el objeto original o que reemplace su comportamiento.



Patrón *Proxy*

- **Aplicaciones del proxy:**
  - **Virtual proxy:** demorar la construcción de un objeto hasta que sea realmente necesario, cuando sea poco eficiente acceder al objeto real.
  - **Protection proxy:** Restringir el acceso a un objeto por seguridad.
  - **Remote proxy:** representar un objeto remoto en el espacio de memoria local. Es la forma de implementar objetos distribuídos. Estos proxies se ocupan de la comunicación con el objeto remoto, y de serializar/deserializar los mensajes y resultados.

Proxy virtual

```
public class ConcreteImageViewer implements ImageViewer {
   private Image image; private Rectangle extent;
   public ConcreteImageViewer(String path, Rectangle rec) {
            image = Image.load(path);      // Costly operation
            extent = rec;  }
   public void displayImage() {
            image.display();      // Costly operation
   }}

public class ImageViewerProxy implements ImageViewer {
    private String iPath; private Rectangle extent; private ImageViewer viewer;

    public ImageViewerProxy(String path, Rectangle rec) {
            iPath = path;
            extent = rec; }

    public void displayImage() {
            if (viewer == null) {
                viewer = new ConcreteImageViewer(iPath, extent); }
            viewer.displayImage(); }}
```

Proxy de protección

```
public class RealBankAccount implements BankAccount {
      private float balance;

public float balance() {
      return balance;
}

public void deposit(float amount) {
      balance += amount;
}

public void withdraw(float amount) {
      balance -= amount;
}
}
```

Proxy de protección

```
public class BankAccountProxy  implements BankAccount {
    private BankAccount realAccount;
    public BankAccountProxy(BankAccount anAccount) {
        realAccount = anAccount; }

    public float balance() {
        if (! this.checkAccess()) {
            throw new RuntimeException("accesso denegado"); }
        return realAccount.balance();
    }
    public void deposit(float amount) {
        if (this.checkAccess)
                realAccount.deposit(amount); }
    public void withdraw(float amount) {
        if (this.checkAccess)
                realAccount.withdraw(amount);}

    private boolean checkAccess() ...
```

Proxy de acceso remoto

- Para acceder a objetos que se encuentran en otro espacio de memoria, en una arquitectura distribuida
- El proxy empaqueta el request, lo envía a través de la red al objeto real, espera la respuesta, desempaqueta la respuesta y retorna el resultado
- En este contexto el proxy suele utilizarse con otro objeto que se encarga de encontrar la ubicación del objeto real. Este objeto se denomina **Broker**, del patrón de su mismo nombre

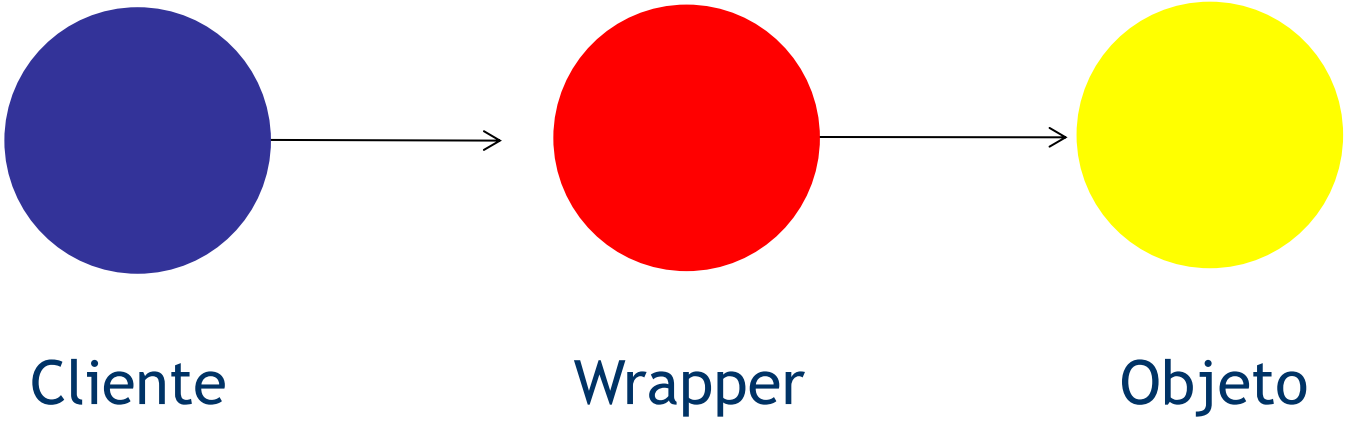
Ejemplos de Proxy

- https://java-design-patterns.com/patterns/proxy/
- https://stackabuse.com/the-proxy-design-pattern-in-java/
- ...

Adapter, Decorator, Proxy

- Todos patrones estructurales
- Todos con diagramas de objetos similares
- Distinto propósito

- A todos se los llama también “wrappers”



Proxy vs. Decorator vs. Adapter