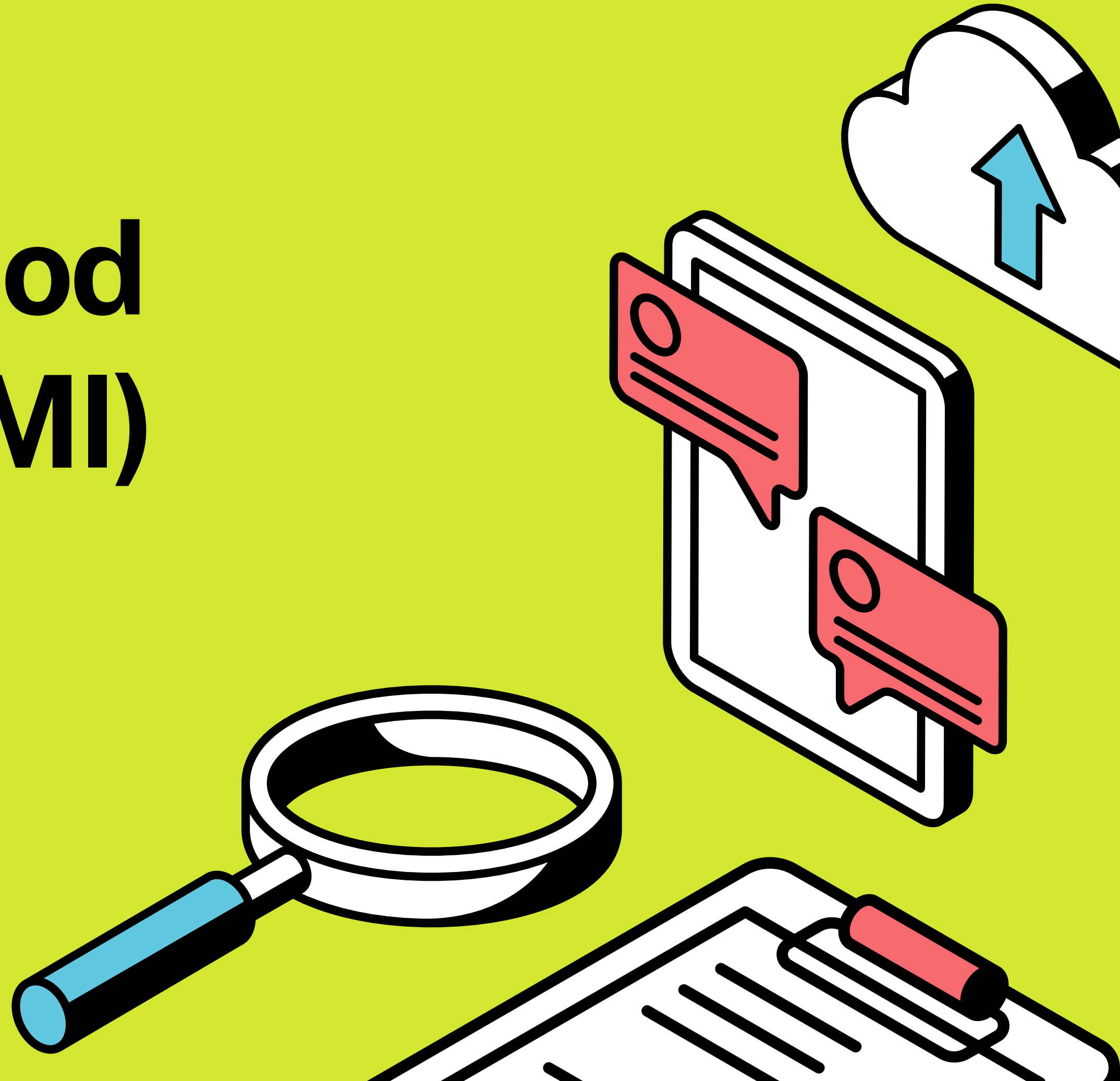


Remote Method Invocation (RMI)

Integrantes:

- CHICA VALFRE VALESKA SOFIA
- PIÑA PAREDES MIGUEL ANGEL
- VERA MACIAS JOHN KLEINER
- VILCACUNDO CHILUISA JORDY ALEJANDRO



Introducción

Concepto

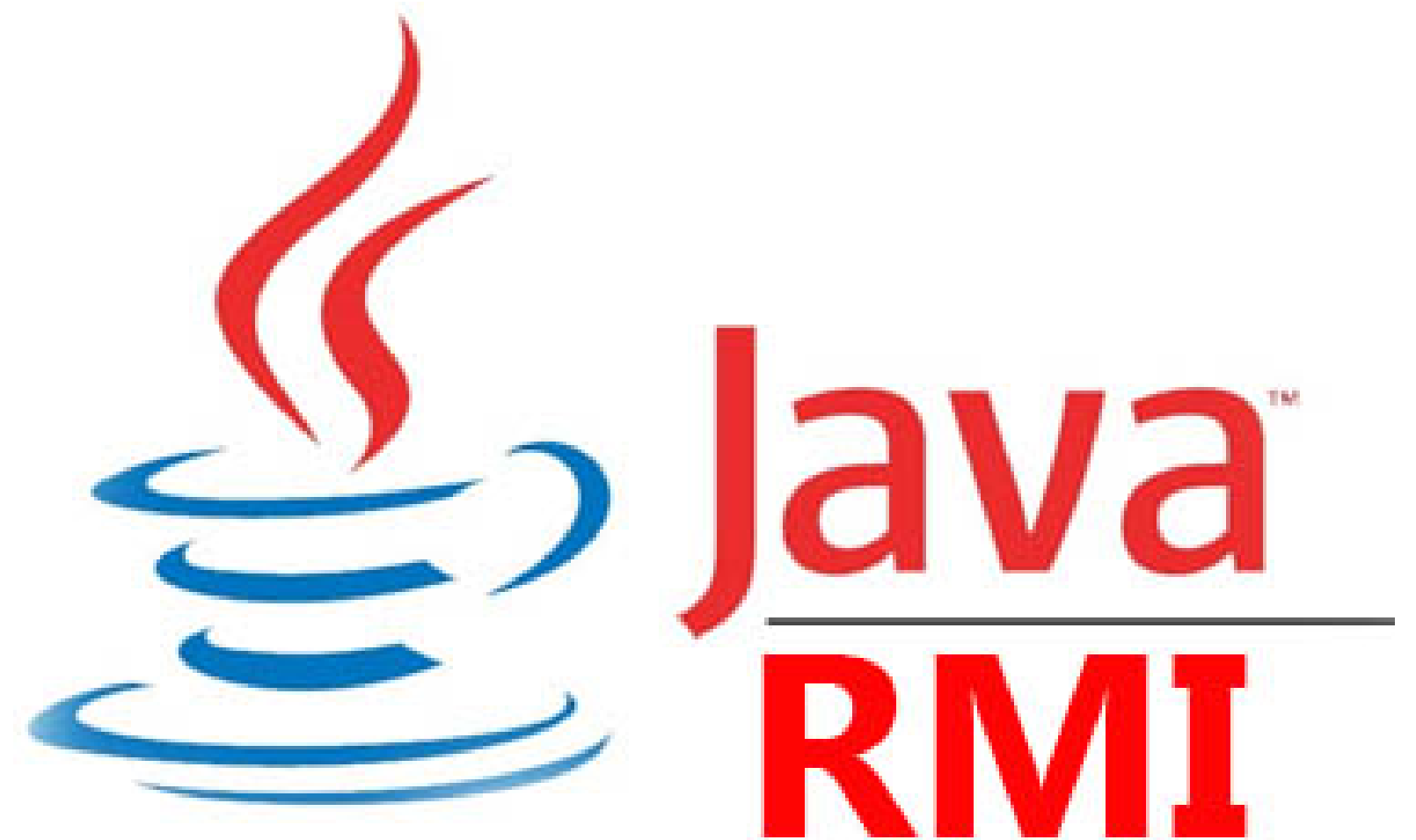
El paradigma RMI en Java facilita la comunicación eficiente entre objetos distribuidos, superando barreras físicas y permitiendo la operación eficaz en sistemas geográficamente dispersos. Este resumen destaca la importancia de RMI en sistemas distribuidos, ilustra su aplicabilidad con un ejemplo práctico y aborda los desafíos de seguridad en la comunicación remota.



Propósito de utilizar RMI

Beneficios

- Abstracción de la comunicación remota: Simplifica el desarrollo y promueve una interfaz intuitiva.
- Simplificación del desarrollo: El modelo de programación orientado a objetos simplifica el desarrollo de aplicaciones distribuidas.



Beneficios de utilizar RMI

Beneficios

- Abstracción de la comunicación remota: Simplifica el desarrollo y promueve una interfaz intuitiva.
- Simplificación del desarrollo: El modelo de programación orientado a objetos simplifica el desarrollo de aplicaciones distribuidas.



Contexto en el que se utiliza comúnmente

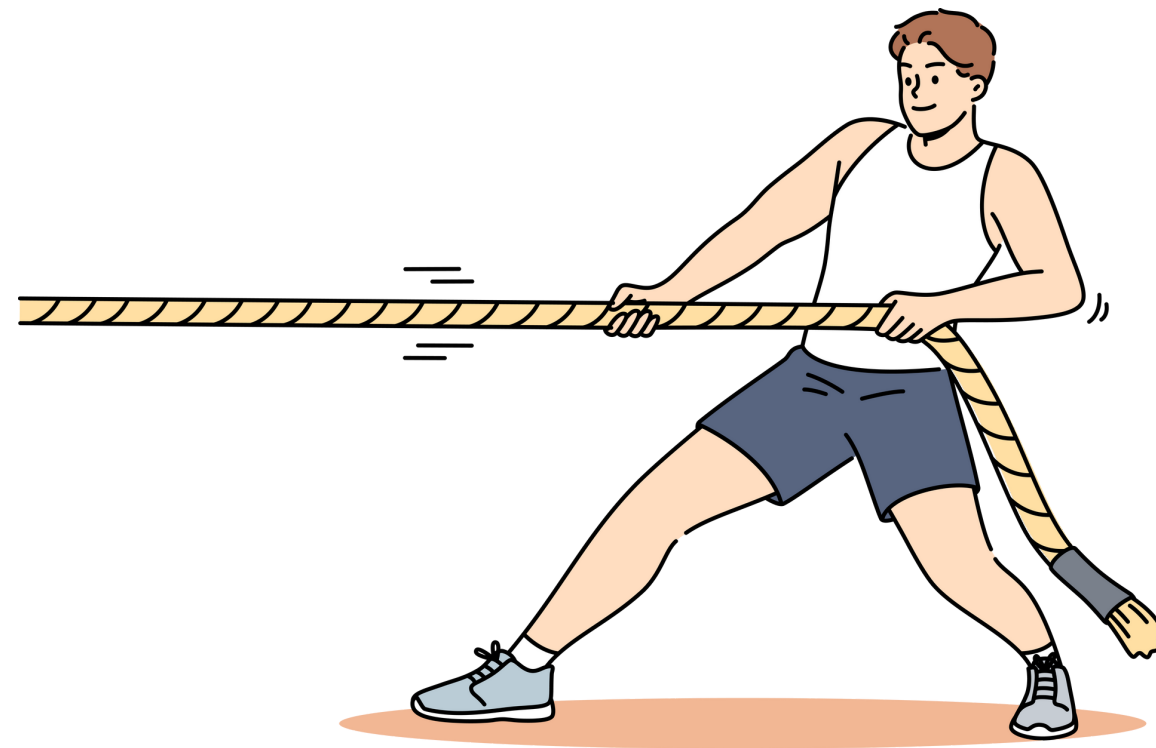
Uso Común: RMI es crucial en sistemas distribuidos para permitir la ejecución de métodos en objetos remotos, facilitando la comunicación entre componentes de software en entornos distintos, especialmente en el desarrollo de aplicaciones empresariales en Java.



Contexto en el que se utiliza comúnmente

Desafíos:

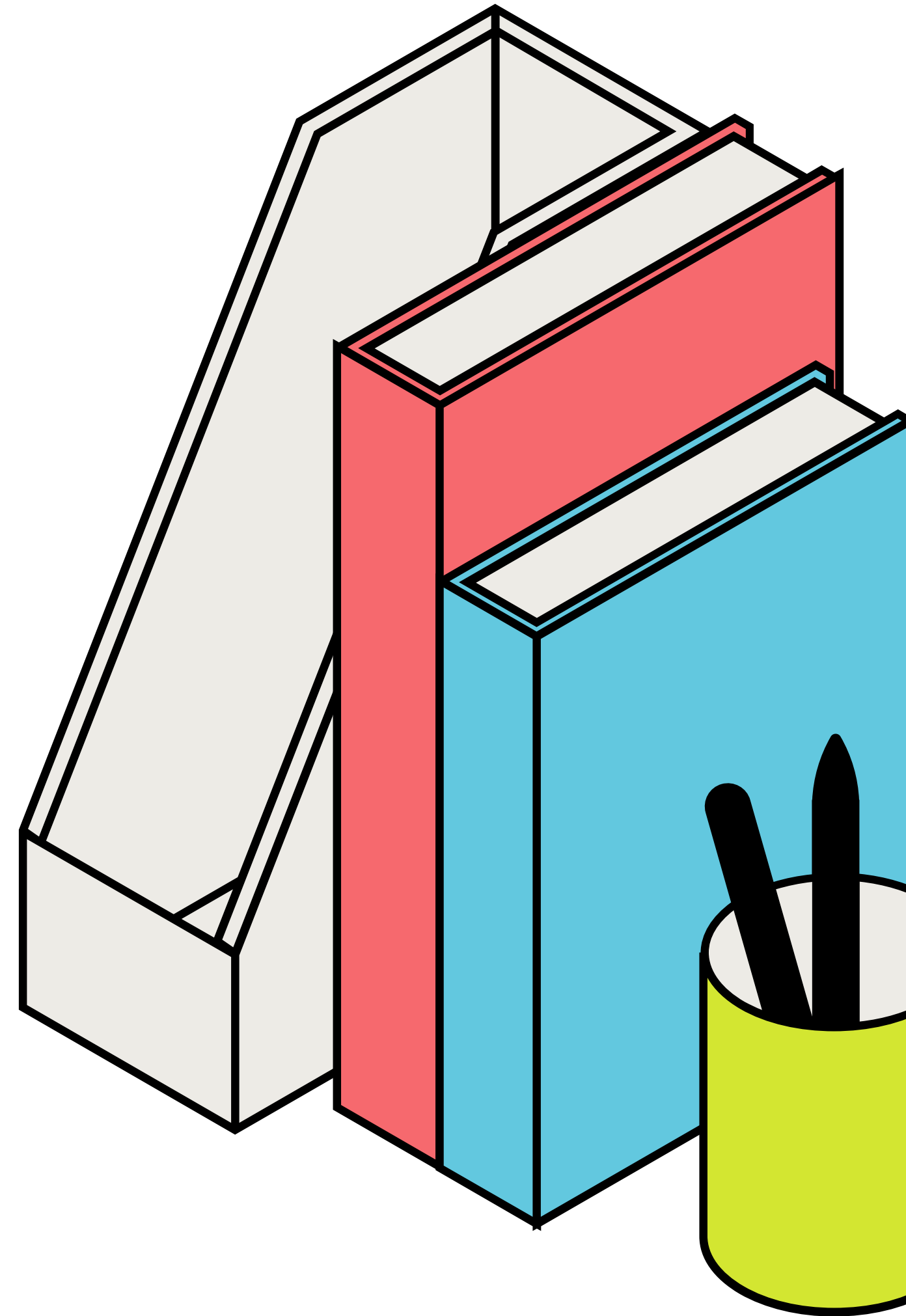
- Limitada interoperabilidad con tecnologías no Java, lo que puede ser un obstáculo en entornos heterogéneos.
- Dificultades para atravesar proxies y firewalls, afectando la conectividad en redes corporativas.
- Preocupaciones de seguridad debido a la descarga remota y ejecución de código, requiriendo prácticas sólidas de codificación y configuración segura.



Arquitectura de RMI

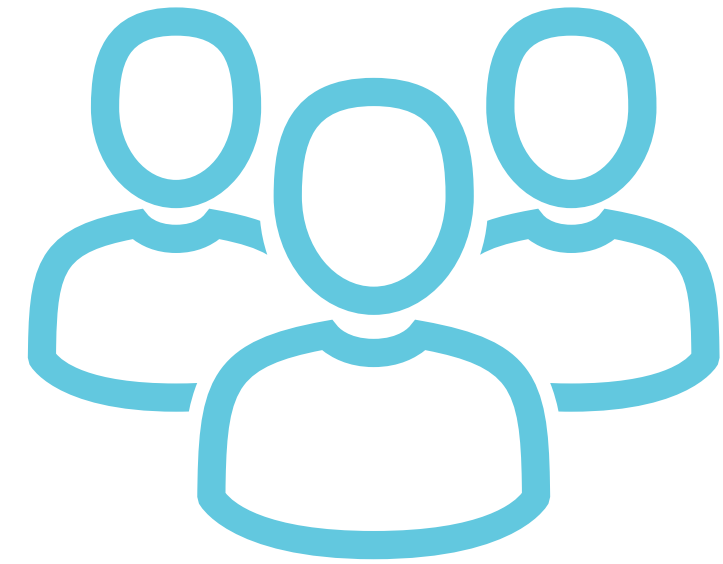
Componentes clave de la arquitectura RMI

En RMI, la interfaz remota define métodos invocables remotamente, extendiendo `java.rmi.Remote` y manejando errores con `RemoteException`. Los objetos remotos implementan estas interfaces y pueden ser invocados por clientes. El Registro actúa como directorio para localizar objetos remotos. Stubs y Esqueletos facilitan la comunicación, transmitiendo invocaciones entre el cliente y el objeto remoto.



Roles en RMI (Cliente, Servidor, Registro RMI)

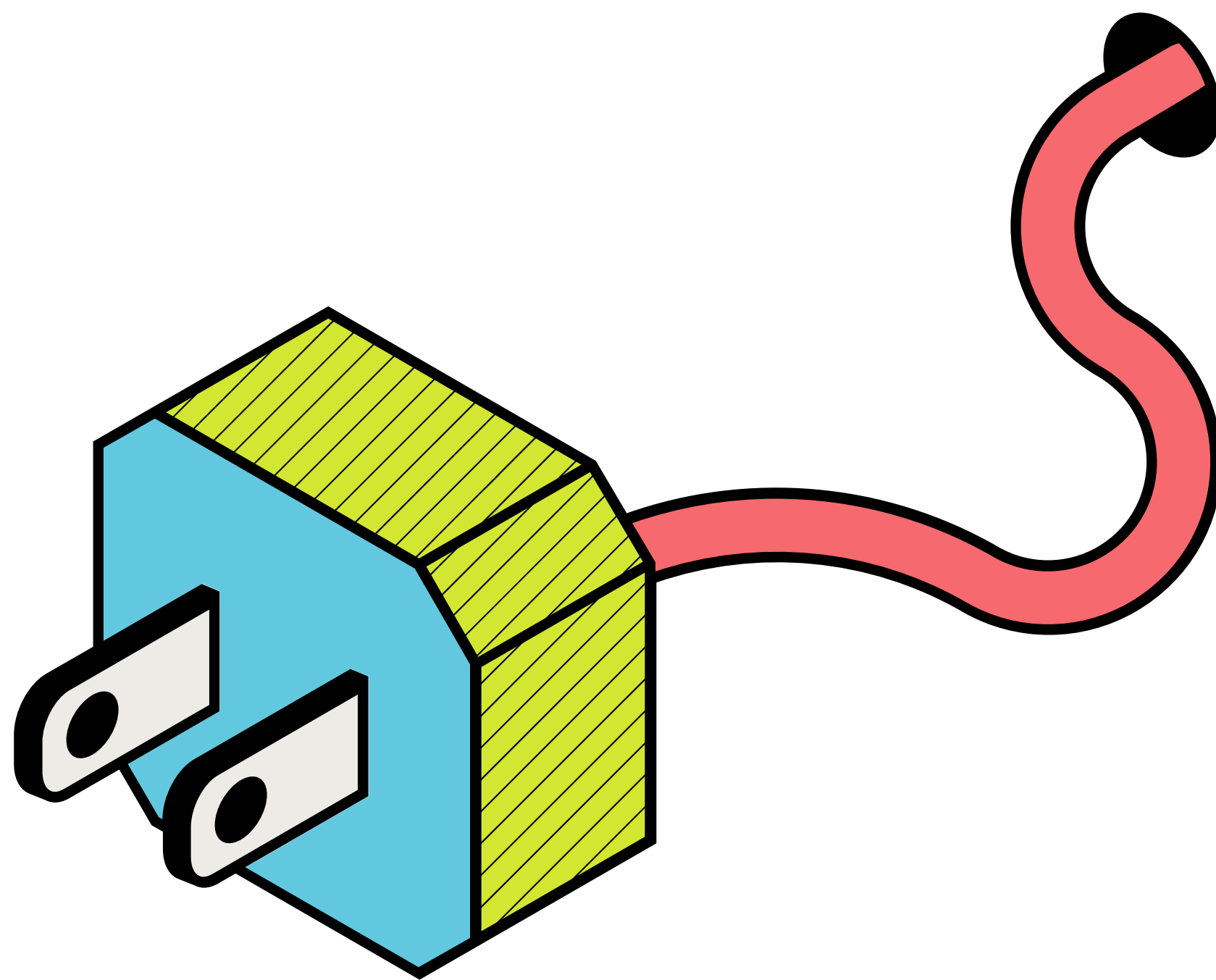
El Cliente inicia la comunicación remota y consulta el Registro RMI para referencias a objetos distribuidos. El Servidor contiene y ejecuta estos objetos, proporcionando servicios a los clientes. El Registro RMI funciona como un directorio central para conectar clientes y servidores en entornos distribuidos.



Como funciona RMI

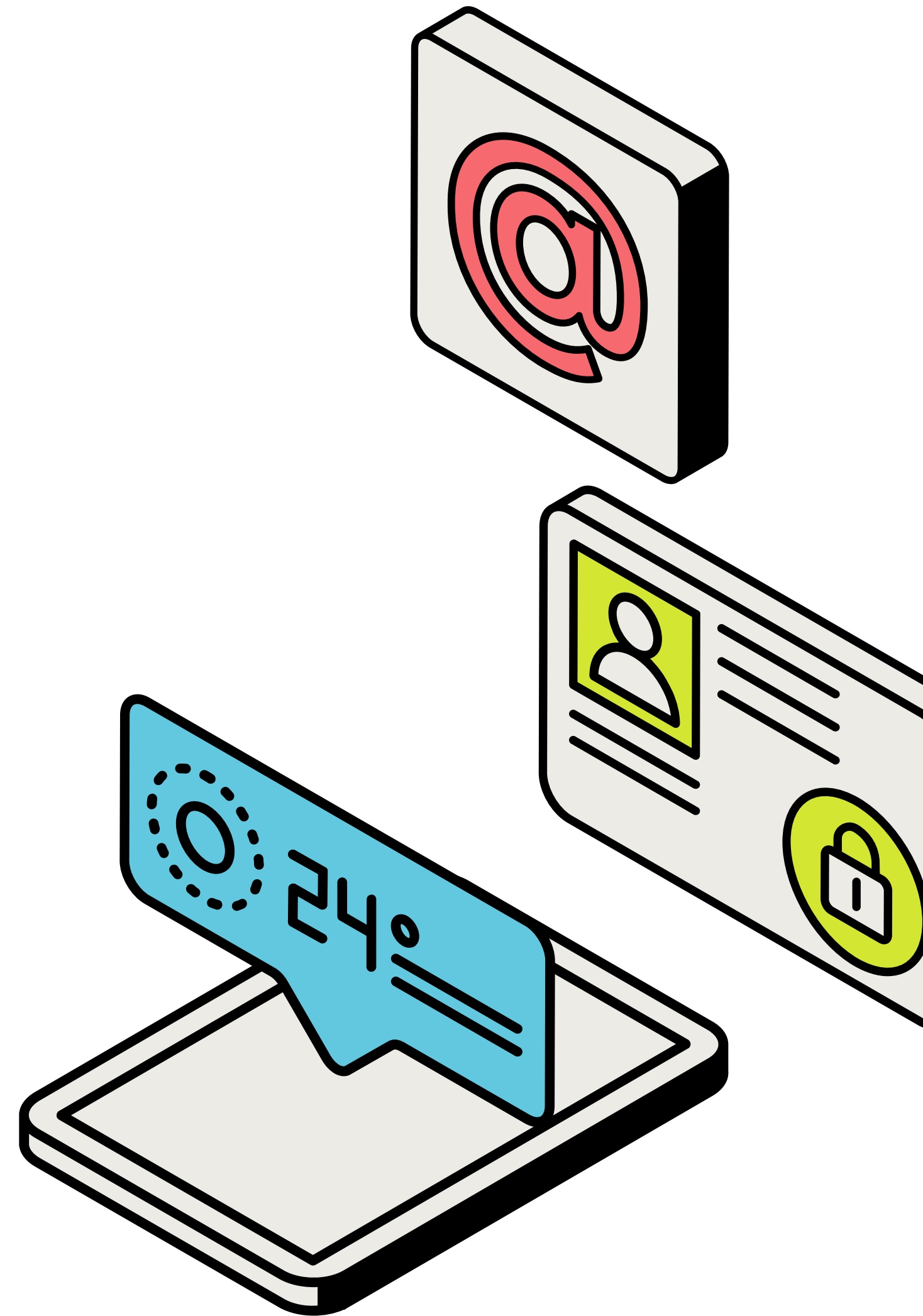
Explicación del proceso de invocación de métodos remotos

La Invocación Remota de Métodos (RMI) en Java es un proceso para llamar métodos en sistemas remotos. Incluye crear interfaces y clases, generar stubs y esqueletos para comunicación cliente-servidor, registrar objetos en el servidor, y establecer conexiones desde el cliente. Un ejemplo práctico ayuda a entender y aplicar RMI en proyectos reales.



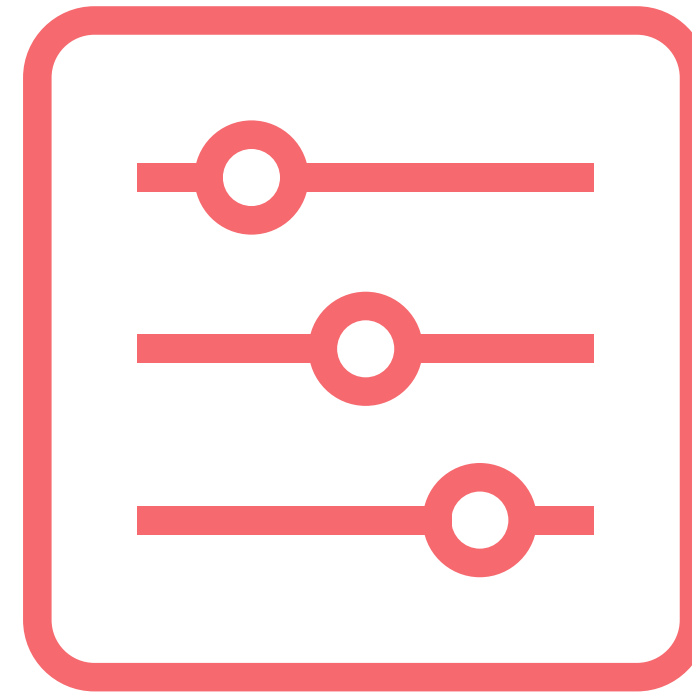
Comunicación con RMI en Objetos Distribuidos

La Comunicación entre Objetos Distribuidos (COD) en Java, a través de RMI, permite que objetos en distintas JVMs interactúen en una estructura cliente-servidor. Usa stubs y esqueletos como intermediarios, con interfaces remotas que extienden `java.rmi.Remote`. RMI maneja la carga dinámica de stubs, y los argumentos y retornos deben ser serializables para transferir información entre JVMs.



Paso de Parámetros y Resultados en RMI

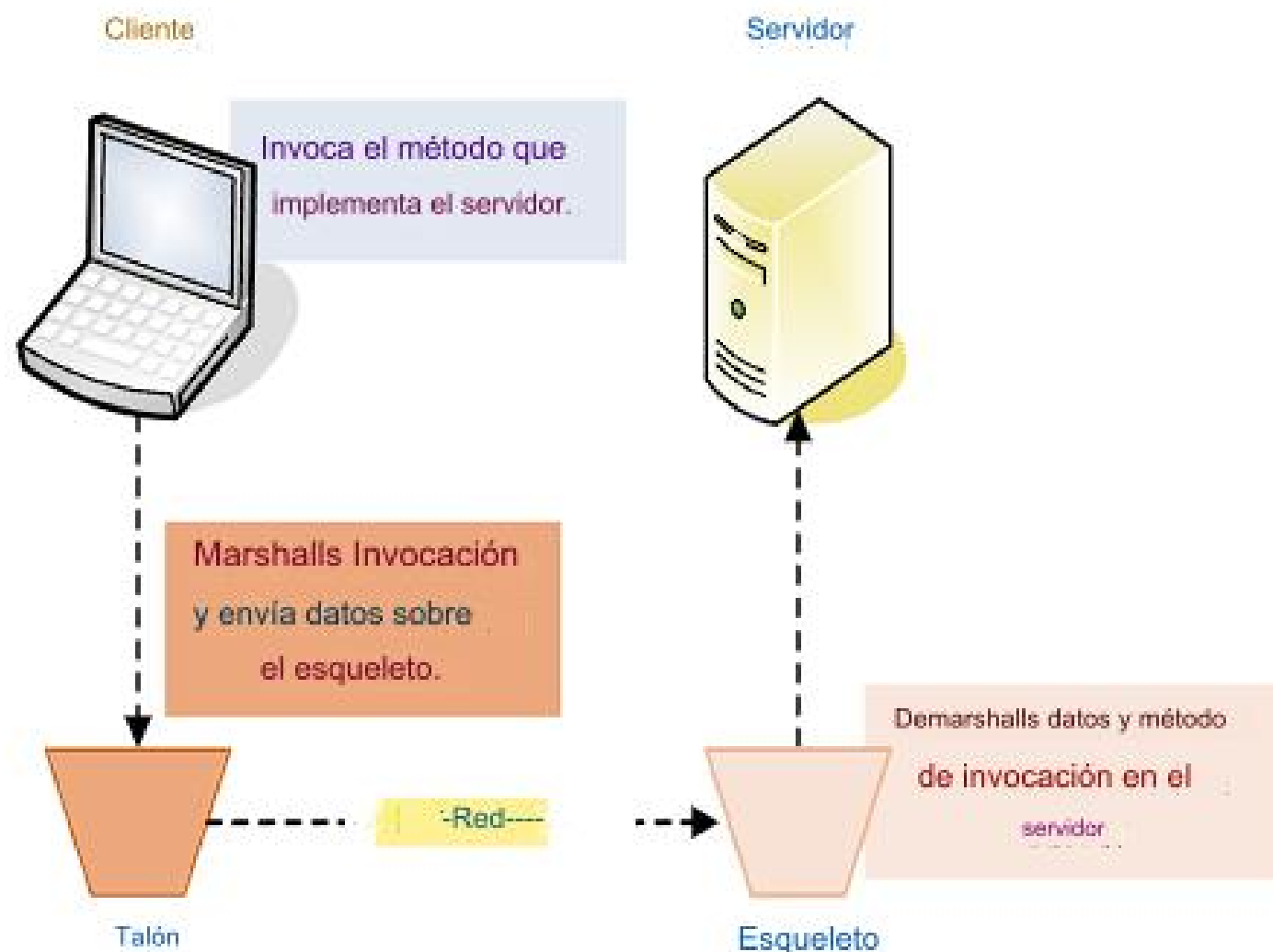
RMI permite la invocación simple de métodos en objetos remotos, usando flujos de bytes para transferir datos y automatizando la serialización/deserialización. Soporta el paso de parámetros por valor y referencia, facilitando la comunicación con objetos complejos serializables.



Implementación de RMI

Uso de interfaces remotas

El concepto de interfaces remotas en RMI se basa en el uso de interfaces en Java. Las interfaces son una forma de definir un conjunto de métodos que una clase debe implementar. En RMI, las interfaces remotas se utilizan para definir los métodos que un objeto remoto debe proporcionar.

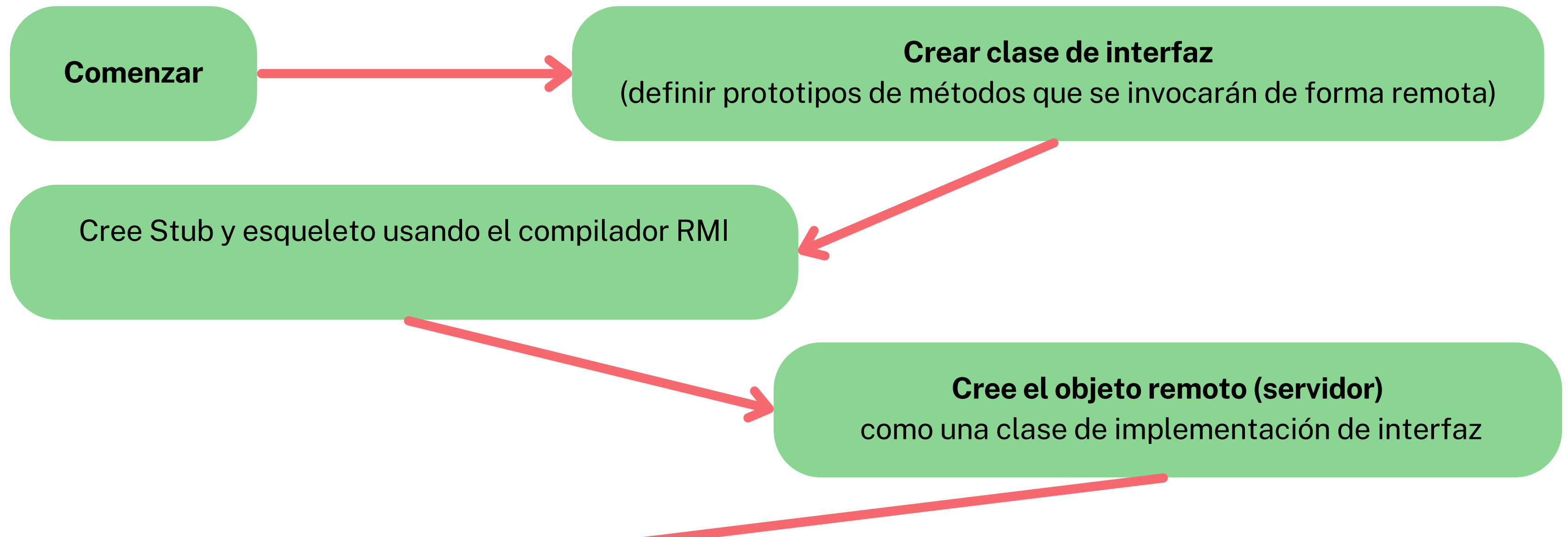


El uso de interfaces remotas en RMI tiene varias ventajas:

- El cliente y el servidor pueden estar en diferentes máquinas y aun así comunicarse de manera efectiva.
- El cliente puede realizar llamadas a métodos en el objeto remoto como si fuera un objeto local.
- RMI permite que el servidor proporcione servicios a múltiples clientes simultáneamente

Creación de objetos remotos

La creación de objetos remotos en RMI implica varios pasos clave. Primero, se define una interfaz remota que especifica los métodos que el objeto remoto proporcionará.



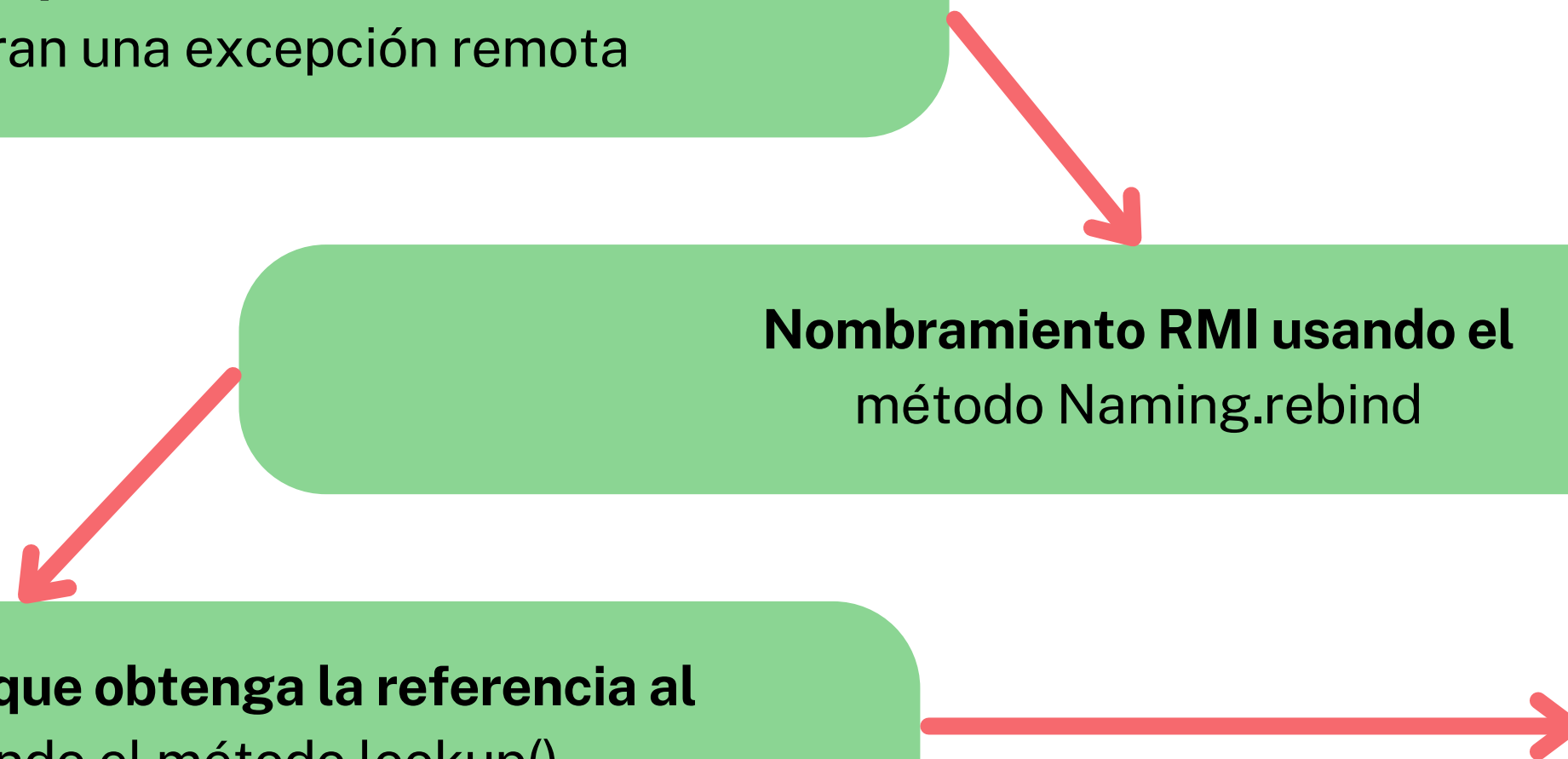
Creación de objetos remotos

Cree servidor de programación que defina todos los métodos del servidor como si lanzaran una excepción remota

Nombramiento RMI usando el método Naming.rebind

Cree una clase de cliente que obtenga la referencia al objeto remoto utilizando el método lookup()

Fin



Registro de objetos remotos en el registro RMI

El registro RMI es un servicio que permite a los clientes buscar objetos remotos por nombre y obtener referencias a ellos. Cuando se registra un objeto remoto en el registro RMI, se le asigna un nombre único que se puede utilizar para buscarlo.

- El registro RMI se ejecuta en una máquina específica y mantiene una lista de nombres de objetos remotos y sus referencias.
- El registro RMI es una parte fundamental de la infraestructura de RMI, ya que facilita la comunicación entre clientes y objetos remotos.



Registro de objetos remotos en el registro RMI

Para registrar objetos remotos en el registro RMI, se utiliza la clase `java.rmi.Naming`. Aquí está la información relevante sobre el registro de objetos remotos en el registro RMI:

1

Uso de `java.rmi.Naming`: La clase `java.rmi.Naming` proporciona métodos estáticos para asociar nombres con objetos remotos.



2

Búsqueda de un objeto remoto: Para buscar un objeto remoto en el registro RMI, se utiliza el método `lookup` de la clase `java.rmi.Naming`. Por ejemplo:

```
```java
MyRemoteInterface obj = new MyRemoteObject();
Naming.rebind("MyRemoteObject", obj);
```
```

Registro de objetos remotos en el registro RMI

3

Registro de un objeto remoto: Para registrar un objeto remoto en el registro RMI, se utiliza el método `rebind` de la clase `java.rmi.Naming`. Por ejemplo: `MyRemoteObject`

```
```java  

MyRemoteInterface obj = (MyRemoteInterface)
Naming.lookup("rmi://localhost/MyRemoteObject");

```
```

Seguridad de RMI

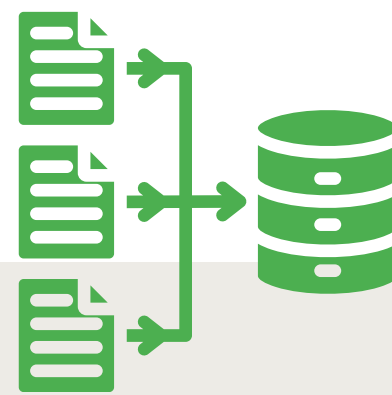
Desafíos de seguridad en el contexto de RMI.

Los sistemas distribuidos actualmente se enfrentan a varios desafíos de seguridad.



Autenticación de Clientes y Servidores:

Garantizar que tanto el cliente como el servidor sean quienes dicen ser, evitando la suplantación de identidad.



Integridad de los Datos:

Proteger la integridad de los datos transmitidos entre sistemas distribuidos para evitar manipulaciones no autorizadas.



Confidencialidad:

Asegurar que la información sensible no sea accesible para usuarios no autorizados durante la transmisión.

Métodos para garantizar la seguridad RMI.

La seguridad es muy importante, es por ello por lo que RMI hace uso de los componentes de seguridad implícitos de Java, lo que permite que su estructura esté protegida cuando los clientes realizan descargas de ejecuciones.



Implementación de Certificados SSL/TLS:

El uso de Secure Sockets Layer (SSL) o Transport Layer Security (TLS) proporciona una capa de seguridad adicional al cifrar la comunicación entre el cliente y el servidor, garantizando la confidencialidad y la integridad de los datos.



Configuración de Mecanismos de Autenticación:

Establecer mecanismos robustos de autenticación, como el uso de tokens de seguridad o la integración con sistemas de autenticación centralizados, contribuye a verificar la identidad de los participantes en la comunicación RMI.

Métodos para garantizar la seguridad RMI.

También, en situaciones extremas, un servidor tiene la opción de rechazar la descarga de cualquier aplicación.



Autorización basada en Roles:

Definir roles y permisos específicos para los usuarios y sistemas involucrados en la RMI. Esto asegura que solo las operaciones autorizadas se lleven a cabo, reduciendo el riesgo de acceso no autorizado.



Uso de Firewalls y Configuraciones de Red Seguras:

Implementar firewalls y configuraciones de red seguras ayuda a prevenir ataques externos y proteger la infraestructura de RMI contra accesos no autorizados.

Conclusión

La implementación de Remote Method Invocation (RMI) en Java es un enfoque fundamental para la comunicación entre procesos en entornos distribuidos. A través de la definición de interfaces remotas, la creación de objetos remotos, la generación de stubs y esqueletos, y el registro en el RMI registry, se facilita la conexión entre clientes y servidores de manera transparente. La capacidad de RMI para gestionar la invocación eficiente de métodos en objetos remotos, así como el manejo de la serialización de objetos para su transmisión a través de la red, lo posiciona como una herramienta esencial en el desarrollo de aplicaciones distribuidas en Java.



Gracias.

