

# Documentación de Notificaciones

## ¿Qué es una Dependencia?

En el mundo de la programación cuando hacemos la creación de diversos proyectos, nos surge la pregunta si lo que creamos ¿Necesitara depender de algo dentro para hacerlo funcionar con otra cosa? La dependencia en este caso, es una relación de reutilización, es decir, cuando un objeto independiente usa como referencia a los objetos para realizar las acciones o comportamientos necesarios.

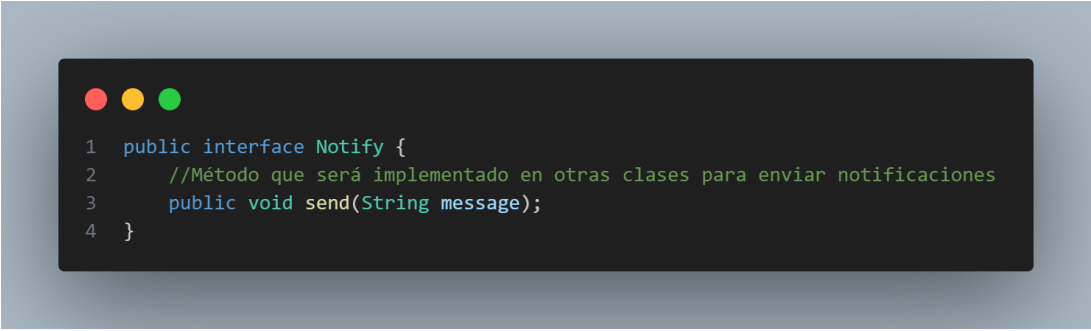
## ¿Qué es la inyección de dependencias?

Es una “técnica” que facilita el desarrollo de proyectos que promueve un diseño más flexible, modular y estable, en sí, es un patrón de diseño. Previamente como vimos en que es la dependencia, un objeto que otro necesita para funcionar. En caso de la inyección, es proporcionar las dependencias necesarias para que el objeto pueda funcionar por su propio medio.

Con estos conceptos claros, ahora podemos tener un proyecto aplicando las dependencias, en concreto la inyección de dependencias en Java, simulando un sistema de notificación por 2 métodos de vía.

## Clase Notify

En esta clase se define el método para su utilización en todo el proyecto de notificaciones, la cual será *send*, al ser una clase de tipo interfaz definirá el molde necesario para enviar un mensaje al sistema de notificaciones.

A screenshot of a code editor with a dark background and light-colored text. The code defines a public interface named 'Notify' with a single method 'send' that takes a 'String message' as a parameter. There are four lines of code, numbered 1 to 4 on the left. The code is as follows:

```
1 public interface Notify {  
2     //Método que será implementado en otras clases para enviar notificaciones  
3     public void send(String message);  
4 }
```

## Clase Service

En esta clase *Service*, vamos a realizar la inyección de dependencia a través de la interfaz sin necesidad de implementar dicha interfaz. Lo haremos a través del constructor de esta clase, para que se pueda usar y crear por sí mismo y estar listo.

```

1 public class Service {
2     //Aquí se realiza la inyección de dependencia de notificaciones
3     private Notify notify;
4
5     // El constructor recibe la dependencia de la interfaz sin necesidad de implementar
6     public Service(Notify notify){
7         this.notify = notify;
8     }
9
10    // Metodo para invocar a otras clases con un mensaje en concreto
11    public void notifyUser(String message){
12        notify.send(message);
13    }
14 }

```

## Clase TypeEmail & TypeSMS

En estas clases concretas vamos a crear los tipos de notificación que haremos, basándonos en la vida cotidiana, existen los métodos vía correo y SMS. Haremos la creación para cada uno de ellos, vamos a implementar en ambas la interfaz *Notify* para poder acceder al método que actúa como molde.

```

1 public class TypeSMS implements Notify{
2     //Se implementa el método de la interfaz para un tipo de notificación en especial
3     @Override
4     public void send(String message) {
5         System.out.println("Enviando notificación por SMS con este mensaje; " + message);
6     }
7 }

```

```

1 public class TypeEmail implements Notify{
2     //Se implementa el método de la interfaz para un tipo de notificación en especial
3     @Override
4     public void send(String message) {
5         System.out.println("Enviando notificación por correo con este mensaje; " + message);
6     }
7 }

```

## Clase Main

En esta clase observamos todo el funcionamiento de todo el código y observaremos la inyección de dependencias a través del sistema de notificaciones. En esta creamos instancias concretas y las inyectamos.

```
1 public class App {
2     //Implementamos un Excepcion para arrojar a consola un error en caso de que se presentará
3     public static void main(String[] args) throws Exception {
4         // Crear la instancia del objeto de cada tipo de Notify
5         Notify notify = new TypeEmail();
6         // Realizamos la inyección desde el servicio
7         Service emailService = new Service(notify);
8         emailService.notifyUser("Has recibido un mensaje vía correo");
9
10        // Creamos otra instancia del objeto ahora para otro tipo de Notify
11        Notify notify2 = new TypeSMS();
12        Service smsService = new Service(notify2);
13        smsService.notifyUser("Has recibido un mensaje vía SMS");
14    }
15 }
```