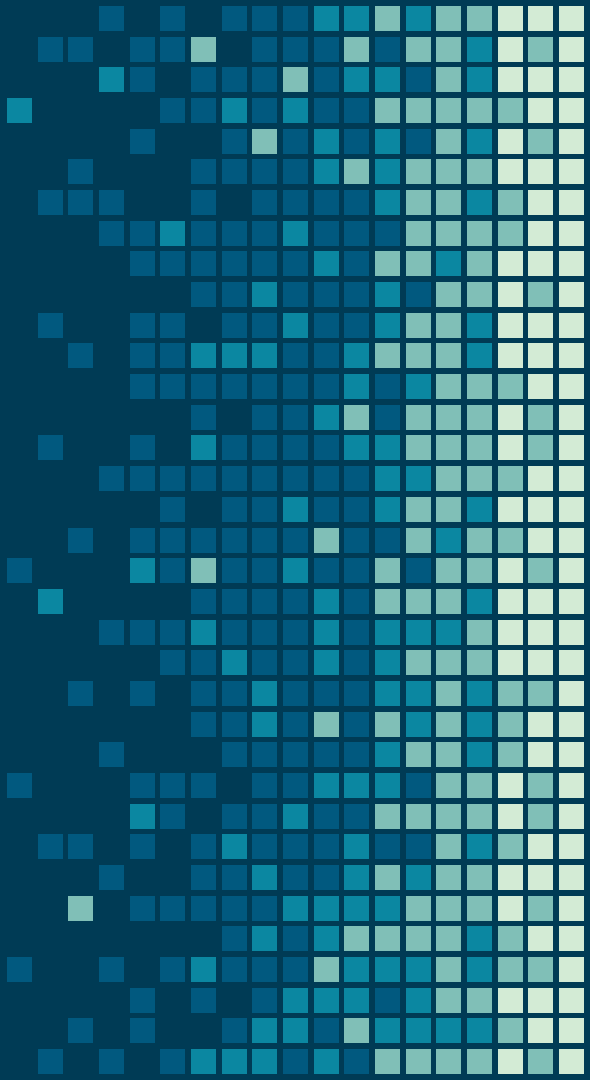
INVERSIÓN DE CONTROL E INYECCIÓN DE DEPENDENCIA

Tópicos Especiales de Software



INTEGRANTES:

- **LUIS VASQUEZ**
- **MIGUEL OVIEDO**
- **NELSON SANABIO**

INTRODUCCIÓN

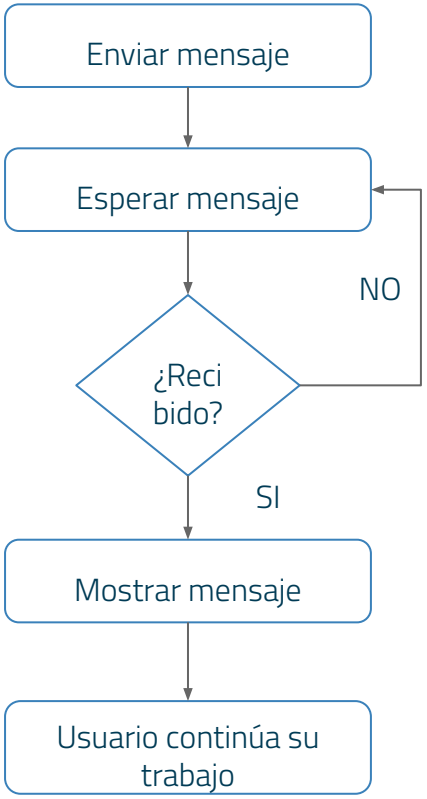
El paradigma de la Programación Orientada a Objetos cambió el modo de desarrollar sistemas, haciéndolos más complejos, pero también más eficientes. Implementó los conceptos de **modularidad** y **reutilización**, pero con ello también se generaron problemas de **dependencia entre módulos** cuando los sistemas se hacían más grandes siendo muy complicado su escalabilidad, fue ahí que aparecieron dos nuevos conceptos: **la Inversión de Control (IoC)** y la **Inyección de Dependencia (ID)**.

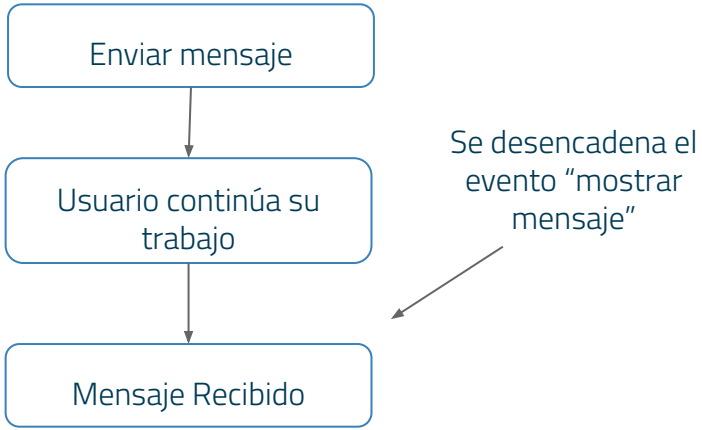
INVERSIÓN DE CONTROL (IoC)

IoC es un principio de diseño de software en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales. En éstos la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones donde el programa tiene el control de ejecución. En cambio, en la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

IoC se basa en el principio de Hollywood " No nos llames, nosotros te llamamos", para evitar estar recibiendo llamadas de aspirantes preguntando si han sido aceptados o no.

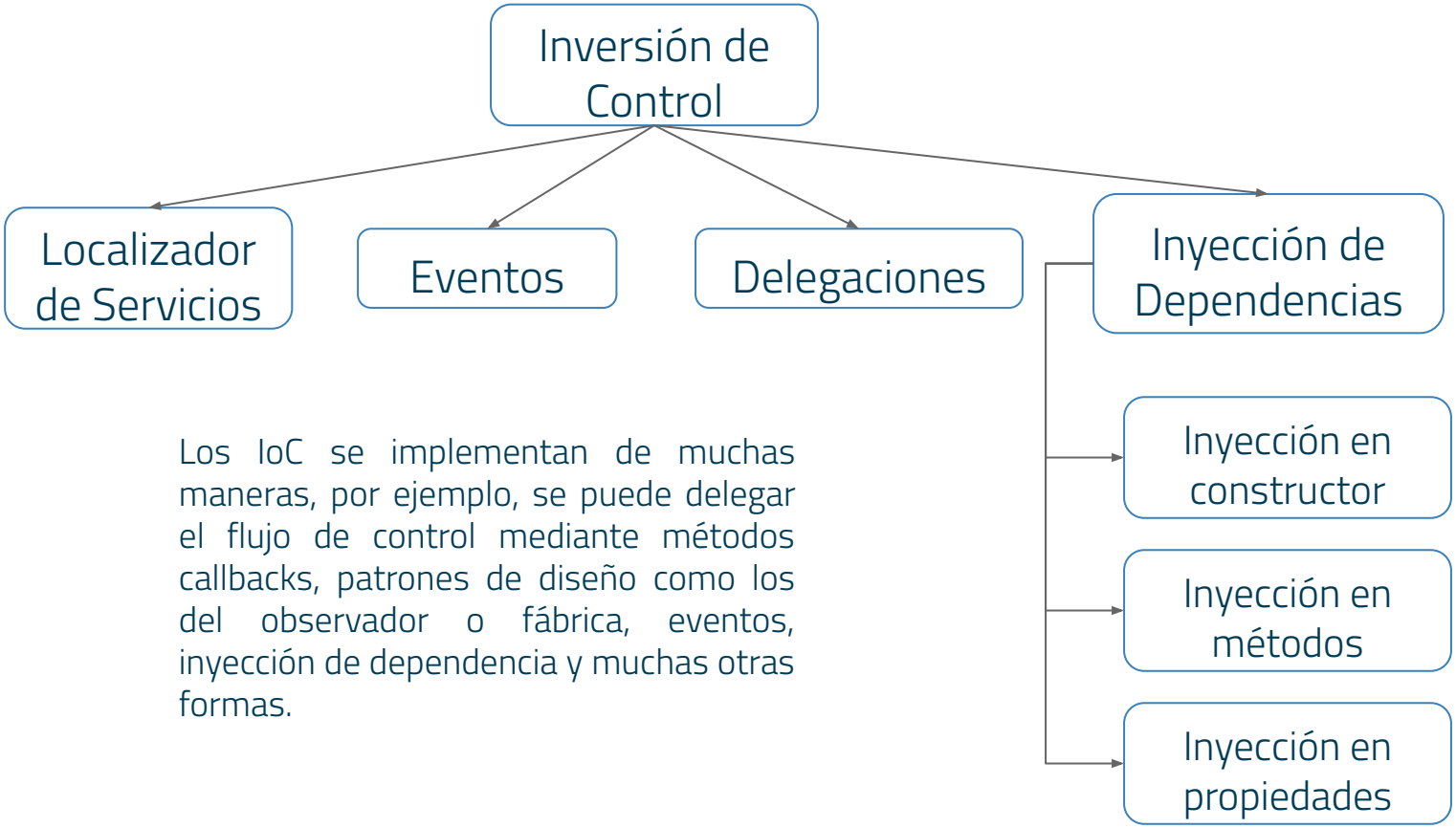
FLUJO SECUENCIAL Y POR EVENTOS





La aplicación escucha los eventos. Si llega un mensaje, el evento se activa y el mensaje se recibe y se muestra.

IMPLEMENTAR IoC



Los IoC se implementan de muchas maneras, por ejemplo, se puede delegar el flujo de control mediante métodos callbacks, patrones de diseño como los del observador o fábrica, eventos, inyección de dependencia y muchas otras formas.

INYECCIÓN DE DEPENDENCIAS (DI)

DI es un patrón de diseño utilizado para implementar IoC. Permite la creación de objetos dependientes fuera de una clase, es decir, las clases no crean los objetos que necesitan, sino que se los suministra otra clase 'contenedora' que inyectará la implementación deseada. Usando DI, movemos la creación y el enlace de los objetos dependientes fuera de la clase que depende de ellos.

PRINCIPIO DE INYECCIÓN DE DEPENDENCIAS (DIP)

DIP es uno de los cinco principios SOLID, que son básicos en la programación orientados a objetos. Fue definido por Robert Martin. Este principio establece dos puntos fundamentales:

1. Las clases de alto nivel no deberían depender de las clases de bajo nivel. Ambas deberían depender de las abstracciones.
2. Las abstracciones no deberían depender de los detalles. Los detalles deberían depender de las abstracciones.

EJEMPLO

```java
public class Person {

        private String name;

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }
}
```

```java
public class MySql {

        public void savePerson(Person person) {
                System.out.println("Save person ok...");
        }

}
```

```java
public class ServicePerson {

        public void savePerson(Person person) {
                MySql mysql = new MySql();
                mysql.savePerson(person);
        }

}
```

¡Abstracción!

Aquí se genera acoplamiento

```java
public interface IPersistence {

        public void save(Object object);


}
```

```java
public class MySql implements IPersistence {

        public void save(Object object) {
                System.out.println("Save  ok...");

        }

}
```

```java
public class ServicePerson {

        private IPersistence persistence;

        public ServicePerson(IPersistence persistence) {
                this.persistence = persistence;
        }

        public void savePerson(Person person) {
                persistence.save(person);
        }

}
```

Se instancia la clase que se quiera implementar

FORMAS DE INYECTAR DEPENDENCIAS

Existen varias formas de inyectar dependencias, depende de los lenguajes de programación, pero se pueden destacar dos fundamentales:

1. Inyección por constructors
2. Inyección por métodos setters

INYECCIÓN POR CONSTRUCTOR

```
public class Vehiculo
{
    private IMotor m;

    public Vehiculo(IMotor motorVehiculo) {
        m = motorVehiculo;
    }

    //...
}
```

INYECCIÓN POR MÉTODOS SETTERS

```
public class Vehiculo
{
    private IMotor m;

    public Vehiculo() {}

    public void setMotor(IMotor motor) {
        this.m = motor;
    }

    //...
}
```

CONTENEDORES IoC

Un contenedor IoC es un framework para implementar la inyección de dependencia automáticamente. Administra la creación de objetos y su tiempo de vida, y también inyecta dependencias a la clase.

El contenedor IoC crea un objeto de la clase especificada y también inyecta todos los objetos de dependencia a través de un constructor, una propiedad o un método en el tiempo de ejecución y lo dispone en el momento adecuado. Esto se hace para que no tengamos que crear y administrar objetos manualmente.

CONTENEDORES IoC

Algunos contenedores son los siguientes:

- Unity
- Spring
- StructureMap
- Castle Windsor
- Ninject

REFERENCIAS

1. http://www.davidvalverde.com/blog/inversion-de-control/
2. https://www.tutorialsteacher.com/ioc/ioc-container
3. https://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias
4. https://medium.com/eduesqui/principio-de-inversi%C3%B3n-de-dependencias-solid-d2ad865ac0c3

DEMO :)