Podríamos definir un framework como un esquema o marco de trabajo predefinido que nos brinda una serie de herramientas que nos permiten simplificar tareas complejas en procesos más simples.

Entre estas herramientas se pueden incluir librerías (bibliotecas), anotaciones, generadores, plantillas, entre otras.

¿Por qué utilizar un framework?

- **Eficiencia**: Los frameworks vienen con una amplia variedad de funcionalidades preintegradas que eliminan la necesidad de escribir código repetitivo. Esto acelera significativamente el proceso de desarrollo.
- **Consistencia y Estándares**: Al seguir las convenciones y estructuras de un framework, los desarrolladores pueden mantener un código más limpio y coherente. Esto facilita la colaboración entre diferentes miembros del equipo y la comprensión del código a largo plazo.
- **Mantenimiento y Escalabilidad**: Los frameworks están diseñados para facilitar el mantenimiento y la escalabilidad de las aplicaciones. Su estructura modular permite agregar nuevas funcionalidades o modificar las existentes de manera más sencilla.
- **Seguridad**: Muchos frameworks incluyen mecanismos de seguridad integrados, como la gestión de autenticación y autorización, la protección contra ataques comunes (por ejemplo, inyecciones SQL) y la validación de datos.
- **Comunidad y Soporte**: Los frameworks populares tienen grandes comunidades de desarrolladores. Esto significa que hay una gran cantidad de recursos, tutoriales, documentación y soporte disponibles.

¿En qué se diferencia un framework de una librería?

Los frameworks tienen una característica bastante marcada que los diferencia por completo de una librería específicamente me refiero a la inversion de control.

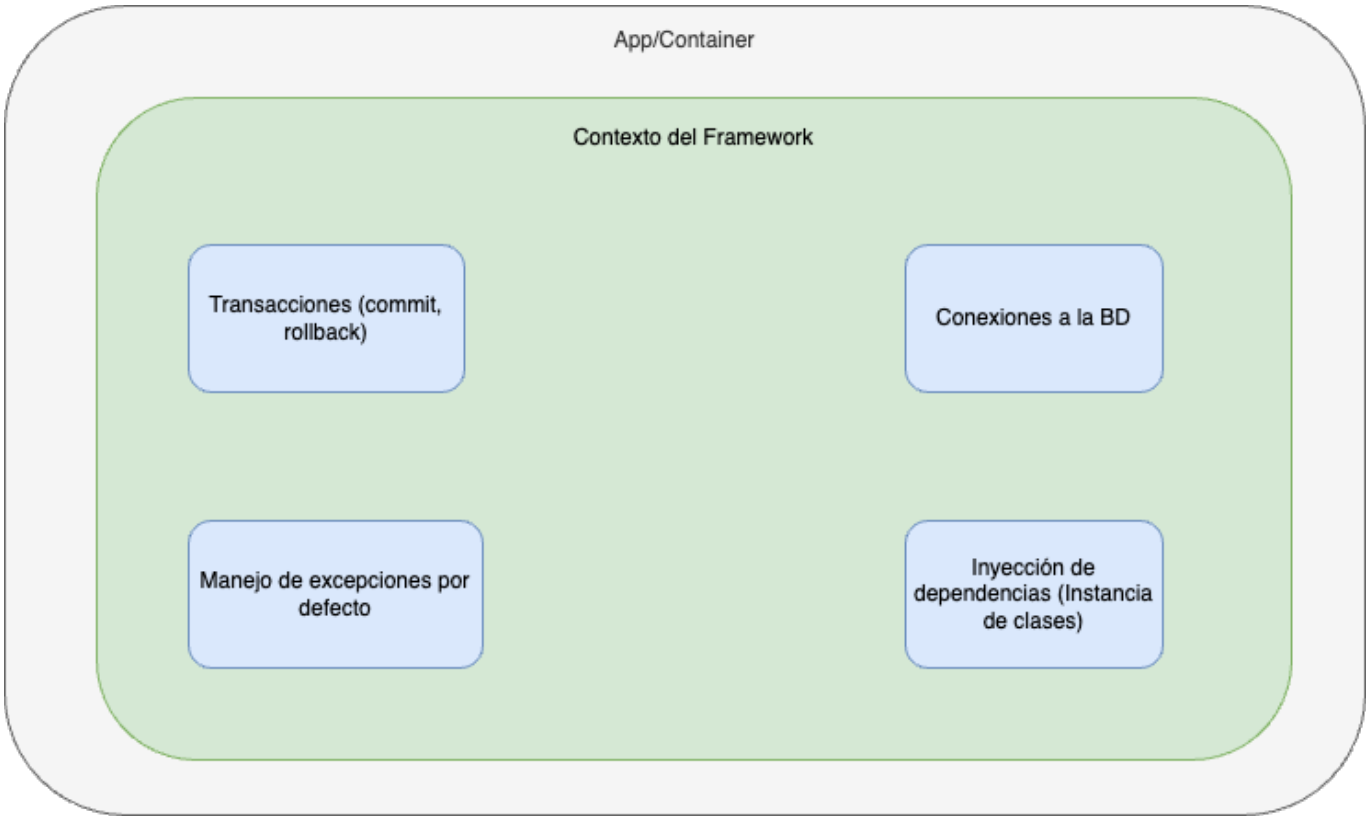
¿Qué es la inversion de control (IoC)?

La inversión de control (inversion of control) es un concepto fundamental en los

frameworks, cuando hablamos de IoC nos referimos a entregar el control del flujo de una aplicación al framework y quitarnos esa tarea como desarrolladores.

Por flujo no me refiero a la lógica de negocio del sistema, me refiero al flujo de los componentes del mismo, por ejemplo, el framework se encargará de manejar las transacciones hacía la bd, crear instancias de clases, administrar conexiones a la base de datos, manejo de errores y excepciones.

Nuestra única tarea generalmente es configurar el framework para indicarle lo que necesitamos que administre, esto se hace de diferentes formas, depende de cada framework, pero la más común son las anotaciones (@Service, @Bean, @Controller).



Tal y como se ve en la imagen, los framework contienen todas esas funciones de nuestra aplicación de forma automática y además las invoca, algo que si lo pensamos bien sería muy complejo de realizar por nosotros mismos.

En resumen los framework son los que invocan nuestro código y contienen el flujo de nuestro aplicación.

¿Entonces las librerías no tienen IoC?

La respuesta es no. Ninguna librería funciona de esa manera porque nosotros somos quienes invocamos a la librería y sus diferentes funciones, mientras que como viste antes los frameworks cuentan con la característica de IoC y son ellos quienes invocan nuestro código, veamos un ejemplo rápido de una librería bastante popular "**Lodash**"

Lodash es una librería de utilidades que proporciona muchas funciones útiles para manipular arrays, objetos, cadenas de texto, entre otros. Es muy popular debido a su rendimiento, modularidad y amplia gama de funcionalidades.

Primero la instalamos:

```
npm install lodash
```

Luego tenemos que incluirla en nuestro código:

```
const _ = require('lodash');
```

Ahora hacemos uso de la librería:

```
let string = 'Foo Bar';
let camelCasedString = _.camelCase(string);
console.log(camelCasedString); // 'fooBar'
```

La diferencia es clara, en el caso del framework no tenemos que incluirlo en nuestro código para hacer uso de él, a menos que estemos configurando una parte puntual de su funcionamiento, generalmente es todo lo contrario, nuestro código "esta dentro del framework".