Trabajo Fin de Máster

Sistema de Gestión de Vacaciones

# Índice

[Índice 2](#_Toc201339703)

[Índice de Imágenes 3](#_Toc201339704)

[1. Introducción 4](#_Toc201339705)

[2. Tecnologías y frameworks utilizados 5](#_Toc201339706)

[Backend 5](#_Toc201339707)

[Patrón Arquitectura Hexagonal 5](#_Toc201339708)

[Referencias 6](#_Toc201339709)

# Índice de Imágenes

[Figura 1 Arquitectura Hexagonal 5](#_Toc201339639)

# 1. Introducción

Este documento recoge la parte técnica del desarrollo de la aplicación web GetaBreak, la cual sirve para solicitar, administrar y gestionar las vacaciones de los empleados de una empresa. La herramienta esta pensada para contribuir y simplificar el proceso desde que un empleado solicita los días libres hasta que esta es revisada y aprobada por los responsables correspondientes.

La aplicación se ha desarrollado utilizando el siguiente stack tecnológico:

Backend

* Java
* Spring Boot

Frontend

* Angular

Base de datos

* MongoDB

En el backend se ha empleado el patrón de arquitectura hexagonal para mantener un diseño limpio, escalable y fácil de mantener. Se ha hecho uso de la separación en capas, uso de DTOs y MapStruct para agilizar el proceso de trabajo entre los servicios y controladores.

# 2. Tecnologías y frameworks utilizados

Para el desarrollo de la aplicación se ha utilizado un conjunto de tecnologías tanto en el backend como en el frontend con el fin de aportar sencillez, rendimiento y escalabilidad.

## Backend

Tecnologías utilizadas:

### Patrón Arquitectura Hexagonal

“La arquitectura Hexagonal propone que nuestro dominio sea el núcleo de las capas y que este no se acople a nada externo. En lugar de hacer uso explícito y mediante el principio de inversión de dependencias nos acoplamos a contratos (interfaces o puertos) y no a implementaciones concretas”.

Lo que hace es definir puertos de entrada y salida, interfaces y adaptadores. Para que así los otros módulos de la aplicación como la UI o la BBDD puedan implementarlos y comunicarse con la capa donde esta el negocio sin que esta deba saber de dónde la están utilizando.

Diagrama

El contenido generado por IA puede ser incorrecto.

Figura 1 Arquitectura Hexagonal

En resumidas cuentas, lo podemos plantear como:

* A la izquierda el lado del usuario
* La lógica de negocio en el centro
* A la derecha el lado del servidor

### Spring Boot

# Referencias

<https://medium.com/@oliveraluis11/arquitectura-hexagonal-con-spring-boot-parte-1-57b797eca69c> - Patrón Arquitectura Hexagonal

<https://www.ibm.com/es-es/topics/java-spring-boot> - Java Spring Boot