**Departamento de Ciencias de la Computación (DCCO)**

**Carrera de Ingeniería de Software**

**Curso de Aplicaciones Distribuidas**

Diseño de Seguridad con OAuth 2.0

Presentado por: Marlyn Almeida, Sebastian Bolaños, Nicole Lara, Axel Pullaguari

Tutor: Morales, Dario.

Ciudad: Sangolquí, Ecuador

Fecha: 01/02/2025

Contenido

[**1.** **Introducción** 3](#_Toc192426552)

[**2.** **Objetivo** 3](#_Toc192426553)

[**3.** **Implementación de Microservicios** 3](#_Toc192426554)

[**3.1.** **Código Fuente Documentado** 3](#_Toc192426555)

[**3.2.** **Registros de Pruebas** 4](#_Toc192426556)

[**4.** **Diagrama de Interacción entre Microservicios** 4](#_Toc192426557)

**Desarrollo de Microservicios: Código Fuente Documentado y Registros de Pruebas**

# **Introducción**

El desarrollo de microservicios es una parte fundamental de la arquitectura del sistema de gestión de recursos humanos, ya que permite una separación clara de responsabilidades, escalabilidad y mantenibilidad del software. En esta fase, se ha implementado cada microservicio siguiendo las especificaciones definidas en las etapas anteriores, asegurando su correcto funcionamiento mediante pruebas unitarias y de integración. Este documento presenta el código fuente documentado y los registros de pruebas realizadas para validar su correcto desempeño.

# **Objetivo**

Garantizar el desarrollo de microservicios eficientes, modulares y seguros mediante la implementación de código documentado y la ejecución de pruebas unitarias e integradas, asegurando su correcto funcionamiento e interoperabilidad con otros componentes del sistema.

# **Implementación de Microservicios**

Cada microservicio ha sido desarrollado en Spring Boot, siguiendo principios de arquitectura RESTful. A continuación, se describe la funcionalidad de cada uno:

|  |  |
| --- | --- |
| **Microservicio** | **Funcionalidad** |
| auth-service | Manejo de autenticación y autorización con OAuth 2.0 y JWT. |
| api-gateway | Enrutamiento y seguridad de acceso a los microservicios. |
| employee-service | Gestión CRUD de empleados dentro del sistema. |
| attendance-service | Registro y consulta de asistencia del personal. |
| evaluation-service | Administración y evaluación del desempeño de los empleados. |

## **Código Fuente Documentado**

Se ha utilizado JavaDoc para documentar las clases y métodos principales. Ejemplo de documentación en código:

|  |
| --- |
| /\*\*  \* Controlador de empleados que gestiona las operaciones CRUD.  \*/  @RestController  @RequestMapping("/api/employees")  public class EmployeeController {    private final EmployeeService employeeService;  public EmployeeController(EmployeeService employeeService) {  this.employeeService = employeeService;  }  /\*\*  \* Obtiene todos los empleados.  \* @return Lista de empleados  \*/  @GetMapping  public List<Employee> getAllEmployees() {  return employeeService.getAllEmployees();  }  } |

## **Registros de Pruebas**

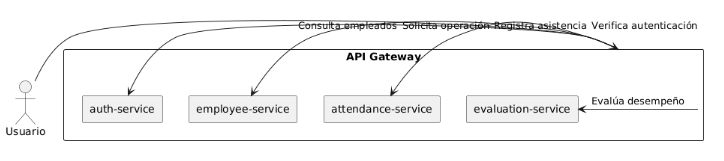
Se han desarrollado pruebas unitarias e integradas para cada microservicio. A continuación, se muestra un ejemplo de una prueba unitaria para employee-service:

|  |
| --- |
| @SpringBootTest  @RunWith(SpringRunner.class)  public class EmployeeServiceTest {    @Autowired  private EmployeeService employeeService;    @Test  public void testGetAllEmployees() {  List<Employee> employees = employeeService.getAllEmployees();  assertNotNull(employees);  }  } |

Se han ejecutado pruebas unitarias con JUnit y pruebas de integración con Postman y Mockito.

# **Diagrama de Interacción entre Microservicios**

El siguiente diagrama en PlantUML representa la interacción entre los microservicios:



* **El usuario** envía una solicitud a través del API Gateway.
* **El API Gateway** valida la autenticación mediante auth-service.
* Si la autenticación es válida, la solicitud se reenvía al microservicio correspondiente (employee-service, attendance-service o evaluation-service).
* **Cada microservicio** procesa la solicitud y responde con la información requerida.