**UNIVERSIDAD NACIONAL DE INGENIERÍA**

**Desarrollo Adaptativo e Integración de Software SI 806 - V**

Logotipo

El contenido generado por IA puede ser incorrecto.

**Taller 03:**

**Sistema GeoAttend**

Docente: Carlos Nelson Ramos Montes

**Integrante:**

* MIGUEL SULLCA, DIEGO ALONSO
* PAREDES MATEO, EDUARD RAFAEL
* SOSA RAMIREZ, CHRISTIAN RENATO
* VALENCIA GREY, WILLIAM GERARDO

**2025 – II**

# INTRODUCCIÓN

Un sistema de asistencia por geocerca: la app Android detecta tu ubicación GPS y, si entras en un radio ≤ 2m de las coordenadas del aula, envía un “push” (ubicación, día/hora y tu código de alumno) al sistema web. El servidor web valida y registra la asistencia en la clase correspondiente. El profesor gestiona todo desde la web: crea su clase y añade alumnos.

# ANÁLISIS DEL SISTEMA

## Descripción del problema

En las instituciones educativas, el control de asistencia de los estudiantes se realiza tradicionalmente mediante métodos manuales, como el pase de lista o el registro físico en planillas. Estos métodos son propensos a errores, demoras y, en muchos casos, permiten el registro fraudulento de asistencia por parte de terceros.

El crecimiento del uso de dispositivos móviles y servicios de geolocalización ofrece una oportunidad para automatizar y asegurar la verificación de la presencia real del estudiante en el aula.

El sistema propuesto tiene como objetivo automatizar el proceso de registro de asistencia mediante el uso de datos de ubicación GPS, validando que el estudiante se encuentre dentro del área autorizada y en el horario programado de la clase. Esto mejora la eficiencia, precisión y transparencia del control de asistencia tanto para docentes como para administradores.

## Actores del sistema

|  |  |  |
| --- | --- | --- |
| Actor | Descripción | Responsabilidades principales |
| Administrador | Encargado de la gestión general del sistema. | Crear usuarios, asignar roles, crear cursos y aulas, supervisar reportes globales. |
| Docente | Responsable del control de asistencia de sus clases. | Crear sesiones de clase, validar asistencias, revisar reportes de sus estudiantes. |
| Estudiante | Usuario final que registra su asistencia desde la aplicación móvil. | Iniciar sesión, visualizar sus clases y registrar su asistencia mediante GPS. |
| Sistema de localización (GPS) | Servicio externo integrado en los dispositivos móviles. | Proporcionar las coordenadas de ubicación en tiempo real. |
| Servicio de autenticación (Auth Service) | Microservicio interno del sistema. | Gestionar login, tokens JWT y permisos de acceso. |

## Requisitos funcionales

|  |  |  |
| --- | --- | --- |
| Código | Requisito | Descripción |
| RF-01 | Registro de usuarios | El sistema debe permitir el registro de estudiantes, docentes y administradores. |
| RF-02 | Autenticación segura | Los usuarios deben autenticarse mediante credenciales y token JWT. |
| RF-03 | Gestión de cursos y horarios | El sistema debe permitir crear, modificar y eliminar cursos, aulas y horarios. |
| RF-04 | Registro automático de asistencia | El estudiante podrá registrar su asistencia solo si su ubicación GPS está dentro del rango permitido. |
| RF-05 | Validación de ubicación | El sistema debe calcular la distancia entre el punto GPS del estudiante y el aula registrada. |
| RF-06 | Generación de reportes | Los docentes y administradores podrán visualizar reportes por curso, estudiante o fecha. |
| RF-07 | Notificaciones | El sistema podrá enviar alertas sobre inasistencias o validaciones fallidas. |
| RF-08 | Dashboard administrativo | Visualización gráfica de estadísticas de asistencia y puntualidad. |

## Requisitos no funcionales

|  |  |  |
| --- | --- | --- |
| Categoría | Requisito | Descripción |
| Rendimiento | RNF-01 | El sistema debe responder a las solicitudes en menos de 2 segundos bajo carga normal. |
| Escalabilidad | RNF-02 | Cada microservicio debe poder desplegarse y escalarse de manera independiente. |
| Seguridad | RNF-03 | Toda la comunicación entre cliente y servidor debe usar HTTPS. |
| Disponibilidad | RNF-04 | El sistema debe mantener una disponibilidad mínima del 99%. |
| Mantenibilidad | RNF-05 | El código debe estar documentado y seguir estándares de desarrollo (PEP8, SOLID). |
| Usabilidad | RNF-06 | La interfaz móvil debe ser intuitiva y accesible para usuarios sin experiencia técnica. |
| Portabilidad | RNF-07 | La app móvil debe funcionar tanto en Android como en iOS. |
| Integrabilidad | RNF-08 | La API debe seguir un estándar REST con documentación OpenAPI. |

## CASOS DE USO

### Estudiante

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Caso de Uso | Descripción | Precondiciones | Flujo Principal (resumen) | Post condiciones | Flujos Alternativos | Servicio |
| UC-001 | **Registrar Usuario** | Crear cuenta nueva en la plataforma | Correo institucional válido; no estar registrado | Accede a registro → llena datos → valida correo → confirma → se crea cuenta | Cuenta creada y verificada | Correo existente; no institucional; no verificado en 24h | User Service |
| UC-002 | **Iniciar Sesión** | Acceso al sistema con credenciales | Cuenta verificada; credenciales válidas | Ingresa datos → valida → genera token JWT → redirige al dashboard | Sesión activa con token | Credenciales erróneas; cuenta inactiva; bloqueo tras 3 intentos | User Service |
| UC-003 | **Inscribirse en Curso** | Inscripción a curso disponible | Sesión activa; curso disponible | Busca curso → selecciona → confirma → valida cupos → registra inscripción | Inscripción confirmada | Sin cupos; ya inscrito; conflicto de horario | Course Service |
| UC-004 | **Registrar Asistencia (GPS Automático)** | Registro automático mediante ubicación | GPS activo; sesión activa; inscrito; horario válido | Envía ubicación → valida geocerca y horario → registra asistencia | Asistencia registrada | Fuera de geocerca; sin GPS; fuera de horario | Attendance Service |
| UC-005 | **Ver Asistencias** | Consultar sus asistencias | Sesión activa; curso inscrito | Accede a “Mis Asistencias” → ve fechas y estado → filtra o analiza | Historial visualizado | Sin registros o pendientes de validación | Attendance Service |
| UC-006 | **Recibir Notificaciones** | Recibir alertas de clases o asistencias | Sesión activa; notificaciones activas | Evento → genera notificación → se muestra en dispositivo | Notificación registrada | Sin conexión o notificaciones apagadas | Notification Service |

### Profesor

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Caso de Uso | Descripción | Precondiciones | Flujo Principal (resumen) | Postcondiciones | Flujos Alternativos | Servicio |
| UC-007 | **Crear Curso** | Crear un curso con información básica y geocerca | Sesión activa; rol profesor | Ingresa datos del curso → define aula y geocerca → crea curso | Curso creado y visible | Código duplicado; aula sin coordenadas | Course Service |
| UC-008 | **Configurar Aula** | Definir coordenadas y área del aula | Sesión activa; curso creado | Selecciona curso → marca ubicación en mapa → define radio | Aula configurada | Usar ubicación actual o búsqueda por nombre | Course Service |
| UC-009 | **Ver Lista de Asistencia** | Consultar lista de asistencias por clase | Sesión activa; curso con alumnos | Selecciona curso → fecha → muestra estados (presente, ausente, etc.) | Lista mostrada | Filtros, búsqueda o lista vacía | Attendance Service |
| UC-010 | **Generar Reporte** | Crear reportes estadísticos del curso | Sesión activa; asistencias registradas | Accede a “Reportes” → selecciona tipo → genera estadísticas → exporta | Reporte generado | Reportes automáticos o por email | Attendance Service |

### Administrador

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Caso de Uso | Descripción | Precondiciones | Flujo Principal (resumen) | Postcondiciones | Flujos Alternativos | Servicio |
| UC-011 | **Gestionar Usuarios** | Crear, editar o eliminar usuarios | Sesión activa; rol admin | Accede al módulo → busca usuario → modifica → guarda cambios | Usuario actualizado; log de auditoría | Reset contraseña; desactivar cuenta | User Service |
| UC-012 | **Configurar Aulas** | Registrar aulas y ubicaciones | Sesión activa; rol admin | Crea aula → define mapa y geocerca → guarda | Aula registrada | Importar CSV; clonar aula similar | Course Service |
| UC-013 | **Monitorear Sistema** | Ver estado de microservicios | Sesión activa; rol admin | Accede al dashboard → visualiza estado de servicios y métricas | Estado del sistema conocido | Configurar alertas o reiniciar servicios | Todos |
| UC-014 | **Generar Reportes** | Generar estadísticas institucionales | Sesión activa; rol admin | Selecciona filtros → agrega datos → genera reporte | Reporte global generado | Reportes automáticos o dashboards | Attendance Service |

Diagrama

El contenido generado por IA puede ser incorrecto.

# DISEÑO DEL SISTEMA

## Arquitectura del Sistema

### Estilo Arquitectónico

El sistema se construye bajo el paradigma de microservicios, donde cada módulo representa un dominio de negocio autónomo con responsabilidades bien definidas (usuarios, cursos, asistencia, notificaciones, etc.).

Esta arquitectura se complementa con un enfoque Event-Driven (orientado a eventos), lo que permite que los servicios se comuniquen de forma asíncrona mediante la publicación y suscripción de eventos.

De esta manera, el sistema alcanza alta escalabilidad, resiliencia, flexibilidad y una baja dependencia entre componentes, adaptándose mejor a entornos distribuidos y al crecimiento progresivo del proyecto.

|  |  |
| --- | --- |
| Factor | Beneficio Principal |
| Descomposición por capacidad de negocio | Cada microservicio representa un dominio funcional independiente (Usuarios, Asistencia, Cursos, Notificaciones), lo que mejora el modularidad y la comprensión del sistema. |
| Independencia en desarrollo y despliegue | Los equipos pueden trabajar de forma paralela en distintos servicios sin interferir entre sí, reduciendo el tiempo de entrega. |
| Base de datos por servicio | Cada microservicio gestiona su propio esquema (por ejemplo, user\_schema, attendance\_schema), evitando acoplamiento de datos y permitiendo mayor autonomía. |
| Comunicación asíncrona (Event-Driven) | Los eventos permiten ejecutar procesos en segundo plano (como notificaciones o actualizaciones de estado) sin bloquear el flujo principal. |
| Comunicación síncrona (REST API) | Se emplea para operaciones críticas o que requieren respuesta inmediata, garantizando coherencia en los flujos esenciales. |
| Desacoplamiento de servicios | Los servicios se comunican solo mediante contratos bien definidos (APIs o eventos), lo que facilita la evolución del sistema sin afectar a otros módulos. |
| Escalabilidad independiente | Se pueden escalar únicamente los microservicios con mayor carga (como el de asistencia GPS), optimizando el uso de recursos. |
| Tolerancia a fallos | Los eventos se almacenan y reprocesan en caso de error, garantizando la consistencia eventual y evitando pérdida de datos. |
| Reactividad y tiempo real | El sistema reacciona automáticamente ante eventos del entorno (como detección de asistencia o creación de cursos). |
| Mantenimiento y evolución sencilla | Facilita agregar nuevas funciones sin alterar los servicios existentes, reduciendo riesgos de regresión. |
| Alta disponibilidad y resiliencia | Al distribuir la carga entre microservicios, se evitan puntos únicos de fallo y se mejora la continuidad operativa. |
| Compatibilidad con entornos Cloud y DevOps | Permite despliegues ágiles con Docker, Kubernetes y CI/CD, adaptándose fácilmente a entornos productivos y escalables. |
| Monitoreo y trazabilidad | Cada evento y transacción puede registrarse, facilitando la auditoría y el análisis de comportamiento del sistema. |

### Componentes de Infraestructura

#### Capa de Presentación (Front-End / Clientes)

Esta capa corresponde a las interfaces de usuario desde las cuales los actores interactúan con el sistema:

* App Móvil (Estudiantes): registra asistencias mediante GPS y consulta historial.
* Web App (Profesores): permite crear cursos, configurar aulas y monitorear asistencias.
* App Móvil (Administradores): gestiona usuarios, monitorea servicios y genera reportes globales.

#### Capa Lógica (Microservicios y Comunicación)

La lógica de negocio está distribuida en varios microservicios independientes, desplegados en contenedores (Docker) y registrados en un Eureka Server para descubrimiento dinámico.

La comunicación entre ellos combina peticiones REST y eventos asincrónicos (Event-Driven), permitiendo flujos desacoplados y alta tolerancia a fallos.

|  |  |  |  |
| --- | --- | --- | --- |
| Servicio | Puerto | Responsabilidades Principales | Comunicación |
| User Service | 8081 | Registro de usuarios, autenticación JWT, roles y perfiles (RBAC) | REST + Eventos (creación/actualización de usuarios) |
| Course Service | 8082 | Gestión de cursos, aulas, horarios y geocercas GPS | REST + Suscriptor de eventos de usuario y asistencia |
| Attendance Service | 8083 | Validación de geocerca, registro de asistencias, reportes GPS | REST + Emisor de eventos de asistencia |
| Notification Service | 8084 | Envío de notificaciones push, correos y alertas del sistema | Event-Driven (suscriptor de eventos) |
| API Gateway | 8080 | Punto de entrada único, autenticación JWT, rate limiting, enrutamiento inteligente | Proxy HTTP |
| Config Server | 8888 | Configuración centralizada de entornos (dev, docker, prod) | REST |
| Eureka Server | 8761 | Descubrimiento y registro de servicios | Service Registry |

#### Capa de Datos (Persistencia y Migraciones)

Todos los microservicios utilizan una base de datos PostgreSQL, pero con schemas independientes para asegurar el aislamiento de los datos:

|  |  |  |
| --- | --- | --- |
| Microservicio | Esquema | Contenido Principal |
| User Service | user\_schema | Usuarios, roles, autenticación |
| Course Service | course\_schema | Cursos, aulas, coordenadas GPS |
| Attendance Service | attendance\_schema | Registros de asistencia, eventos GPS |
| Notification Service | notification\_schema | Notificaciones, alertas, logs de envío |

Además, el sistema usa Flyway para la gestión automática de migraciones de base de datos, esto permite actualizar estructuras de tablas de manera controlada y replicable entre entornos (desarrollo, prueba, producción).

Diagrama

El contenido generado por IA puede ser incorrecto.

## **Diseño de Microservicios**

### User Service (Puerto 8081)

**Responsabilidades:**

* Gestión de usuarios (CRUD)
* Autenticación y autorización
* Generación de tokens JWT
* Control de roles (RBAC)

**Modelo de datos:**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | Identificador único del usuario |
| code | String(20) | Código único del usuario (ej. matrícula) |
| email | String(255) | Correo electrónico del usuario |
| hashed\_password | String(255) | Contraseña cifrada |
| first\_name | String(100) | Nombre |
| last\_name | String(100) | Apellido |
| role | String(20) | Rol del usuario (student, teacher, admin) |
| is\_active | Boolean | Si la cuenta está activa |
| is\_verified | Boolean | Si el usuario ha verificado su cuenta |
| created\_at | DateTime | Fecha de creación |
| updated\_at | DateTime | Fecha de última actualización |

### Course Service (Puerto 8082)

**Responsabilidades**

* Gestión de cursos
* Configuración de aulas
* Definición de geocercas
* Gestión de inscripciones

**Modelo de datos**

* **Cursos**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | Identificador del curso |
| code | String(20) | Código único |
| name | String(255) | Nombre del curso |
| description | Text | Descripción |
| credits | Integer | Créditos del curso |
| academic\_year | String(10) | Año académico |
| semester | String(10) | Semestre |
| teacher\_id | Integer | ID del docente |
| teacher\_code | String(20) | Código del docente |
| max\_students | Integer | Límite de estudiantes |
| detection\_radius | Numeric(5,2) | Radio de detección GPS permitido |
| is\_active | Boolean | Estado del curso |
| created\_at | DateTime | Creación |
| updated\_at | DateTime | Actualización |

* **Salones**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | ID del aula |
| code | String(20) | Código del aula |
| name | String(255) | Nombre del aula |
| building | String(100) | Edificio |
| room\_number | String(20) | Número de aula |
| floor | Integer | Piso |
| latitude | Numeric(10,7) | Latitud |
| longitude | Numeric(10,7) | Longitud |
| altitude | Numeric(8,2) | Altitud (opcional) |
| gps\_radius | Numeric(5,2) | Radio de detección en metros |
| capacity | Integer | Capacidad máxima |
| equipment | Text | Equipos disponibles |
| is\_active | Boolean | Activa/Inactiva |
| created\_at | DateTime | Creación |
| updated\_at | DateTime | Actualización |

* **Aula de curso**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | ID de la asignación |
| course\_id | FK → courses.id | Curso asignado |
| classroom\_id | FK → classrooms.id | Aula asignada |
| day\_of\_week | Integer | Día de la semana (0=Lunes...6=Domingo) |
| start\_time | Time | Hora inicio |
| end\_time | Time | Hora fin |
| is\_primary | Boolean | Si es aula principal |
| is\_active | Boolean | Estado |
| created\_at | DateTime | Creación |

* **Cronograma**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | ID del horario |
| course\_id | FK → courses.id | Curso |
| day\_of\_week | Integer | Día de la semana |
| start\_time | Time | Hora de inicio |
| end\_time | Time | Hora de fin |
| classroom\_id | FK → classrooms.id | Aula (opcional) |
| is\_active | Boolean | Estado |
| created\_at | DateTime | Creación |

* **Inscripción**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | ID de matrícula |
| course\_id | FK → courses.id | Curso |
| student\_id | Integer | ID del estudiante (referencia externa) |
| student\_code | String(20) | Código del estudiante |
| status | String(20) | active, dropped, completed |
| enrollment\_date | DateTime | Fecha de matrícula |
| drop\_date | DateTime | Fecha de retiro |
| created\_at | DateTime | Creación |

### Attendance Service (Puerto 8083)

**Responsabilidades**

* Procesamiento de eventos GPS
* Validación de geocercas
* Registro automático de asistencia
* Generación de reportes
* Cálculo de estadísticas

**Modelo de datos**

* **Eventos GPS**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | ID del evento |
| user\_id | Integer | ID del usuario |
| user\_code | String(20) | Código del usuario |
| course\_id | Integer | ID del curso |
| course\_code | String(20) | Código del curso |
| latitude | Numeric(10,7) | Latitud |
| longitude | Numeric(10,7) | Longitud |
| accuracy | Numeric(8,2) | Precisión del GPS |
| altitude | Numeric(8,2) | Altitud |
| status | Enum(EventStatus) | pending, processed, rejected, error |
| processed\_at | DateTime | Fecha de procesamiento |
| calculated\_distance | Numeric(8,2) | Distancia al aula |
| within\_range | Boolean | Dentro del rango permitido |
| device\_id | String(255) | ID del dispositivo |
| event\_timestamp | DateTime | Momento del evento |
| created\_at | DateTime | Fecha de registro |

* **Registros de asistencia**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | ID del registro |
| gps\_event\_id | Integer | ID del evento GPS |
| user\_id | **Integer** | **ID del usuario** |
| course\_id | Integer | ID del curso |
| status | Enum(AttendanceStatus) | present, late, absent, excused |
| source | Enum(AttendanceSource) | gps\_auto, manual, imported, corrected |
| class\_date | DateTime | Fecha de clase |
| scheduled\_start | DateTime | Hora programada |
| actual\_arrival | DateTime | Hora real de llegada |
| recorded\_distance | Numeric(8,2) | Distancia detectada |
| is\_late | Boolean | Indicador de tardanza |
| created\_by | String(50) | Quién creó el registro (system, teacher, admin) |
| created\_at | DateTime | Creación |
| updated\_at | DateTime | Última modificación |

* **Sesiones de asistencia**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | Integer (PK) | ID de sesión |
| course\_id | Integer | ID del curso |
| course\_code | String(20) | Código del curso |
| teacher\_id | Integer | ID del docente |
| session\_date | DateTime | Fecha |
| scheduled\_start | DateTime | Hora inicio |
| scheduled\_end | DateTime | Hora fin |
| total\_enrolled | Integer | Total matriculados |
| total\_present | Integer | Total presentes |
| total\_absent | Integer | Total ausentes |
| is\_completed | Boolean | Si terminó la sesión |
| created\_at | DateTime | Creación |

### Notification Service

**Responsabilidades**

* Envío de notificaciones push
* Envío de emails
* Gestión de templates
* Histórico de notificaciones

**Modelo de datos**

* **Notificaciones**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | **Integer (PK)** | **ID** |
| user\_id | **Integer** | **ID del usuario** |
| type | **Enum(NotificationType)** | **email, push, sms** |
| status | **Enum(NotificationStatus)** | **pending, sent, failed** |
| priority | **Enum(NotificationPriority)** | **low, normal, high, urgent** |
| subject | **String(255)** | **Asunto** |
| body | **Text** | **Cuerpo del mensaje** |
| recipient\_email | **String(255)** | **Email destino** |
| recipient\_phone | **String(20)** | **Teléfono** |
| related\_course\_id | **Integer** | **Referencia a curso** |
| related\_attendance\_id | **Integer** | **Referencia a asistencia** |
| created\_at | **DateTime** | **Creación** |
| sent\_at | **DateTime** | **Envío** |

* **Plantillas de notificación**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | **Integer (PK)** | **ID** |
| name | **String(100)** | **Nombre** |
| type | **Enum(NotificationType)** | **Tipo (email, push, sms)** |
| body\_template | **Text** | **Plantilla** |
| is\_active | **Boolean** | **Estado** |
| variables | **JSON** | **Variables dinámicas** |
| created\_at | **DateTime** | **Creación** |

* **Preferencia de notificaciones del usuario**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| id | **Integer (PK)** | **ID** |
| user\_id | **Integer** | **Usuario** |
| email\_enabled | **Boolean** | **Activar correo** |
| push\_enabled | **Boolean** | **Activar push** |
| sms\_enabled | **Boolean** | **Activar SMS** |
| attendance\_notifications | **Boolean** | **Activar alertas de asistencia** |
| course\_notifications | **Boolean** | **Activar alertas de curso** |
| system\_notifications | **Boolean** | **Activar alertas del sistema** |
| device\_tokens | **JSON** | **Tokens de dispositivos** |
| created\_at | **DateTime** | **Creación** |
| updated\_at | **DateTime** | **Actualización** |

**Imagen que contiene Escala de tiempo

El contenido generado por IA puede ser incorrecto.**

# INFRAESTRUCTURA

## Configuración de Entorno

La configuración del entorno se basa en un enfoque modular, con variables definidas por servicio y gestionadas a través de archivos .env.

|  |  |
| --- | --- |
| Elemento | Detalle |
| Lenguaje base | Python (FastAPI) |
| Gestor de dependencias | pip y requirements.txt por microservicio |
| Configuración de entorno | Variables como DB\_URL, EUREKA\_SERVER\_URL, PORT, SECRET\_KEY |
| Descubrimiento de servicios | Mediante registro dinámico en **Eureka Server** |
| Mensajería (Event-driven) | Implementada con colas de mensajes (p. ej. RabbitMQ o eventos internos según microservicio) |
| Autenticación | JWT Tokens generados en el User Service |
| Despliegue | Local o en contenedores individuales (Dockerfiles disponibles por servicio) |

## Monitoreo y Observabilidad

Cada microservicio implementa mecanismos para garantizar trazabilidad, diagnóstico y resiliencia:

|  |  |
| --- | --- |
| Herramienta o técnica | Descripción |
| Logging estructurado | Cada servicio registra eventos en formato JSON con niveles (INFO, ERROR, WARNING). |
| Health Checks | Rutas /health o /status permiten verificar la disponibilidad de los servicios registrados en Eureka. |
| Eureka Dashboard | Permite visualizar el estado y la conexión entre microservicios registrados. |
| Traza de eventos | Se implementa mediante identificadores de correlación entre microservicios. |
| Monitoreo de rendimiento | Se pueden integrar herramientas como Prometheus y Grafana para métricas de CPU, memoria y latencia de respuesta. |

## Descubrimiento de Servicios (Eureka Server)

El proyecto utiliza EurekaServer como componente central para el registro, descubrimiento y balanceo dinámico de servicios.  
Esto reemplaza el uso de *docker-compose* como orquestador local, dado que cada microservicio se registra automáticamente en Eureka al iniciarse.

|  |  |
| --- | --- |
| Elemento | Función |
| Service Registry | Cada microservicio se registra indicando su nombre lógico (user-service, attendance-service, etc.) y su URL. |
| Heartbeat | Eureka mantiene comunicación constante con los servicios para determinar su disponibilidad. |
| Load Balancing | Las peticiones pueden ser distribuidas entre múltiples instancias de un mismo servicio. |
| Failover automático | Si un servicio cae, Eureka redirige las peticiones a las instancias activas. |

# CONSIDERACIONES DEL DISEÑO

## Escalabilidad

### Horizontal Scaling:

* Cada microservicio puede escalar independientemente
* Eureka permite descubrimiento dinámico de instancias
* Load balancing automático con Spring Cloud LoadBalancer

### Horizontal Scaling:

* Cada microservicio puede escalar independientemente
* Eureka permite descubrimiento dinámico de instancias
* Load balancing automático con Spring Cloud LoadBalancer

## Resiliencia

El sistema implementa estrategias de tolerancia a fallos y gestión de errores controlada, garantizando la continuidad de los servicios ante fallos temporales o desconexiones de red.

**Circuit Breaker Pattern**

El uso del patrón Circuit Breaker está implementado mediante la anotación:

**@CircuitBreaker**(name **=** "courseService", fallbackMethod **=** "fallbackGetCourse")

public Course getCourse(Long courseId) { **...** }

public Course fallbackGetCourse(Long courseId, **Exception** ex) { **...** }

Este mecanismo evita sobrecargar servicios externos cuando se detectan errores recurrentes, ofreciendo una respuesta de respaldo o cacheada. De esta forma, se mantiene la disponibilidad del sistema y se previenen caídas en cascada.

**Timeouts y Retries**

Los tiempos de espera y reintentos están definidos dentro de la configuración del cliente de comunicación, donde se establecen límites razonables para:

* Timeout de conexión: 5 segundos
* Timeout de lectura: 10 segundos
* Retries automáticos con backoff exponencial

Estos parámetros mejoran la resiliencia ante latencias altas o caídas intermitentes en la red o en servicios dependientes.

## Mantenibilidad

El código fuente refleja un compromiso con las buenas prácticas de ingeniería de software y principios de diseño orientado a objetos, que garantizan su extensibilidad y facilidad de mantenimiento.

**Principios SOLID**

* Single Responsibility: Cada microservicio cumple una única función bien definida (por ejemplo, *UserService*, *CourseService*, *GPSService*).
* Open/Closed: El sistema es extensible mediante eventos o colas, sin necesidad de modificar el código existente.
* Dependency Inversion: Se hace uso de interfaces y clases abstractas para desacoplar las dependencias entre capas, especialmente en el acceso a datos y la comunicación interservicios.

**Clean Code**

El código mantiene una estructura legible y coherente, con nombres descriptivos para clases y métodos, funciones específicas y comentarios precisos donde es necesario. Esto facilita la incorporación de nuevos desarrolladores y reduce el riesgo de errores por interpretación incorrecta del código.

**Testing**

Existen pruebas unitarias por servicio, junto con pruebas de integración usando TestContainers, lo que permite verificar la interacción con bases de datos y dependencias reales dentro de entornos aislados.

Además, se incluyen tests de contrato entre microservicios, asegurando la compatibilidad de los endpoints durante los despliegues o cambios de versión.

## Métricas Clave

|  |  |  |
| --- | --- | --- |
| Métrica | Descripción | Responsable |
| Tiempo de respuesta promedio | Mide la latencia de los endpoints REST y eventos procesados. | API Gateway |
| Tasa de disponibilidad (%) | Porcentaje de servicios activos registrados en Eureka. | Eureka Server |
| Mensajes procesados/minuto | Monitorea la capacidad de procesamiento de eventos en el Notification y Attendance Service. | Event Broker |
| Errores de autenticación | Conteo de intentos fallidos de login o tokens inválidos. | User Service |
| Tasa de entrega de notificaciones | Mide la efectividad del envío (sent vs failed). | Notification Service |
| Consumo de recursos (CPU/RAM) | Supervisión de carga por microservicio. | Infraestructura |

## Deployment

El flujo de despliegue y entrega continua está automatizado mediante un pipeline CI/CD, que cubre las etapas desde la compilación hasta la publicación en contenedores.

CI/CD Pipeline

El pipeline sigue una secuencia clara:

Commit → Build → Tests → Docker Image → Registry → Deploy

Cada etapa incluye validaciones automáticas, pruebas de integración y generación de imágenes Docker para los microservicios. Esto permite una entrega continua y controlada, reduciendo el tiempo de implementación y los errores humanos.

**Despliegue Contenerizado**

El uso de Docker y Docker Compose/Kubernetes posibilita un entorno reproducible y escalable, asegurando la coherencia entre los entornos de desarrollo, pruebas y producción.

Se admite el despliegue mediante Rolling Updates, que permite actualizar instancias de los servicios sin interrupciones, garantizando la disponibilidad continua del sistema.

# Conclusión