# 📘 Guía Maestra: Sistema de Triage y Gestión de Solicitudes Académicas

## 1. Contexto Institucional

El Programa de Ingeniería de Sistemas y Computación cuenta con una comunidad de más de 1.400 estudiantes, docentes y administrativos. De manera permanente, se realizan diversas solicitudes académicas y administrativas (registro de asignaturas, homologaciones, cupos, etc.) a través de múltiples canales como atención presencial, correo electrónico y sistemas académicos.

### 🚨 El Problema

Actualmente, la gestión de estas solicitudes es ineficiente debido a la falta de una estructura unificada, la ausencia de mecanismos formales de clasificación y la carencia de trazabilidad clara. Esto genera sobrecarga operativa y respuestas inoportunas para los estudiantes.

## 2. Propósito y Objetivos

El propósito es aplicar conocimientos de **arquitectura empresarial, patrones de diseño, Spring Boot y Angular** para resolver este problema real.

**Objetivos del Sistema:**

- Registrar solicitudes de manera estructurada.
- Clasificar y priorizar las solicitudes mediante reglas de negocio.
- Asignar responsables de forma controlada.
- Gestionar el ciclo de vida completo y mantener un historial auditable.
- Proveer información clara sobre el estado de cada trámite.

### 🤖 Valor Agregado: Asistencia mediante IA

El sistema podrá integrar opcionalmente un modelo de lenguaje (LLM) para:

- sugerir tipos de solicitud,
- proponer prioridades o

- generar resúmenes del historial.

  **Nota importante:** El sistema debe implementar al menos uno de los elementos anteriores. Sin embargo, debe ser plenamente funcional sin la IA; esta actúa solo como un asistente (RF-11).

# 3. Organización y Metodología

- **Equipo de Trabajo:** Máximo 3 integrantes.
- **Metodología:** Desarrollo incremental dividido en tres hitos de 5 semanas cada uno.
- **Stack Tecnológico:** Java (Spring Boot), TypeScript (Angular), herramientas ORM y API REST.
- **Rol del Estudiante:** Actuarán como un equipo de desarrollo encargado del análisis, diseño, implementación e integración de la solución.

# 4. Requisitos Funcionales (RF)

El sistema debe cumplir con los siguientes requerimientos mínimos:

## RF-01. Registro de solicitudes académicas

El sistema debe permitir registrar una solicitud académica ingresada por un usuario, almacenando al menos:

- Tipo de solicitud
- Descripción de la solicitud
- Canal de origen (CSU, correo, SAC, telefónico, etc.)
- Fecha y hora de registro
- Identificación del solicitante

📌 Justificación: Centralizar solicitudes provenientes de múltiples canales.

## RF-02. Clasificación de solicitudes

El sistema debe permitir clasificar una solicitud académica según su tipo, por ejemplo:

- Registro de asignaturas
- Homologación
- Cancelación de asignaturas
- Solicitud de cupos
- Consulta académica

📌 Justificación: Facilitar el tratamiento diferenciado de las solicitudes.

## RF-03. Priorización de solicitudes

El sistema debe asignar una prioridad a cada solicitud con base en reglas definidas, tales como:

- Tipo de solicitud
- Impacto académico
- Fecha límite asociada

La prioridad debe quedar registrada junto con una justificación.

📌 Justificación: Garantizar atención oportuna a solicitudes críticas.

## RF-04. Gestión del ciclo de vida de la solicitud

El sistema debe gestionar el ciclo de vida de una solicitud, permitiendo los siguientes estados mínimos:

- Registrada
- Clasificada
- En atención
- Atendida
- Cerrada

El sistema debe validar que las transiciones entre estados sean coherentes.

📌 Justificación: Controlar el flujo de atención y evitar inconsistencias.

## RF-05. Asignación de responsables

El sistema debe permitir asignar una solicitud a un responsable autorizado, garantizando que:

- El responsable esté activo
- La asignación quede registrada en el historial

📌 Justificación: Asegurar responsabilidad y trazabilidad.

## RF-06. Registro del historial de la solicitud

El sistema debe mantener un historial auditable de cada solicitud, registrando:

- Fecha y hora de cada acción
- Acción realizada
- Usuario responsable
- Observaciones asociadas

📌 Justificación: Garantizar trazabilidad y auditoría.

## RF-07. Consulta de solicitudes

El sistema debe permitir consultar solicitudes según diferentes criterios, tales como:

- Estado
- Tipo de solicitud
- Prioridad
- Responsable asignado

📌 Justificación: Facilitar el seguimiento y la gestión operativa.

## RF-08. Cierre de solicitudes

El sistema debe permitir cerrar una solicitud únicamente cuando:

- La solicitud haya sido atendida
- Se registre una observación de cierre

Una solicitud cerrada no podrá ser modificada.

📌 Justificación: Formalizar el fin del proceso.

## RF-09. Generación de resúmenes de solicitudes (IA – opcional)

El sistema podrá generar un resumen textual del estado y el historial de una solicitud utilizando un modelo de lenguaje externo.

📌 Justificación: Mejorar la comprensión rápida del caso por parte de los responsables.

## RF-10. Sugerencia automática de clasificación (IA – opcional)

El sistema podrá sugerir el tipo de solicitud y su prioridad a partir del texto descriptivo ingresado, utilizando un modelo de lenguaje externo.

Estas sugerencias deberán ser confirmadas o ajustadas por un usuario humano.

📌 Justificación: Asistir, no reemplazar, la toma de decisiones.

## RF-11. Funcionamiento independiente de IA

El sistema debe operar correctamente sin la integración de modelos de lenguaje, garantizando todas las funcionalidades básicas.

📌 Justificación: Evitar dependencias críticas de servicios externos.

## RF-12. Exposición de servicios mediante API REST

El sistema debe exponer sus funcionalidades principales a través de una API REST, permitiendo su consumo por aplicaciones frontend u otros sistemas.

📌 Justificación: Separación de responsabilidades y arquitectura distribuida.

## RF-13. Autorización básica de operaciones

El sistema debe restringir ciertas operaciones según el rol del usuario, por ejemplo:

- Registro de solicitudes
- Clasificación y priorización
- Cierre de solicitudes

📌 Justificación: Control básico de acceso acorde al contexto académico.

# 📅 Cronograma y RoadMap de Entregas

| Entrega | Título | Semanas | Enfoque Principal |
|---|---|---|---|
| **Hito 1** | **Diseño y Modelado** | 1 - 5 | Modelado de dominio (UML), estados y diseño de contratos API. |
| **Hito 2** | **Backend y Lógica** | 6 - 10 | Implementación en Spring Boot, persistencia con ORM y motor de reglas. |
| **Hito 3** | **Frontend y Seguridad** | 11 - 15 | Interfaz en Angular, consumo de API, seguridad JWT y despliegue. |

# 📝 Rúbrica de Evaluación

Cada entrega será evaluada bajo los siguientes criterios ajustados al alcance del hito:

| Criterio | Excelente (5.0) | Aceptable (3.5) | Insuficiente (1.0-2.0) |
|---|---|---|---|
| **Arquitectura** | Patrones correctos y desacoplamiento claro. | Implementa el patrón con acoplamientos menores. | Estructura confusa o inexistente. |
| **Funcionalidad** | Cumple todos los RF del hito sin errores. | Cumple la mayoría de RF con errores menores. | Funcionalidades críticas no operan. |
| **Calidad de Código** | Limpio, documentado y con estándares. | Legible pero inconsistente en errores. | Difícil de leer y sin estándares. |
| **Valor Agregado** | IA fluida e independiente (RF-11). | Integración básica o dependiente. | Sin integración o rompe el sistema. |