# Instituto Politécnico Nacional

Escuela Superior de Cómputo

## ESCOM

Proyecto final

# Sistema de generación de grafos de conocimiento a partir de noticias

Fecha de entrega: 14 de marzo de 2025

**Ingeniería de software**

**Grupo**: 6CV3


Presentan:

**García García Aram Jesua**

**Hernández Díaz Roberto Ángel**

**Hernández Jiménez Irmin**

**Trejo Flores Johann Daniel**

**Toral Hernández Leonardo Javier**


# Índice

# Índice de figuras

# Índice de tablas

# Tabla de versiones del documento

| Versión | Cambio | Fecha | Descripción |
|---|---|---|---|
| 1.0 | Versión inicial del documento. | 05/03/2025 | En esta primera versión se agregaron los puntos fundamentales del documento. |
| 1.1 | Ampliación del documento | 12/03/2025 | Se añadieron secciones de glosario, referencias, visión general, usuarios, restricciones, y suposiciones y dependencias. |

**Instituto Politécnico Nacional**

Escuela Superior de Cómputo

# Documento de requerimientos

# Introducción

En este documento se abordan las funcionalidades principales con las que contará el sistema de generación de grafos de conocimientos a partir de noticias, proporcionando una descripción detallada de las características, funcionalidades y limitaciones del sistema a desarrollar para resolver las necesidades que se solicitan.

## Objetivos

A continuación, se listan los objetivos que se buscan alcanzar con el desarrollo de este proyecto.

### *Objetivo General*

Procesar contenido textual (en este caso, son notas informativas) para generar grafos de conocimiento mediante técnicas de procesamiento de lenguaje natural (PLN).

### *Objetivo Particular*

Diseñar un sistema para identificar entidades, relaciones y conceptos clave, permitiendo una visualización estructurada y procesable del conocimiento.

## Alcance del sistema

El sistema permitirá procesar contenido textual de noticias para generar grafos de conocimiento mediante técnicas de procesamiento de lenguaje natural (PLN). El sistema abarcará desde la ingesta de noticias en diferentes formatos hasta la visualización y exportación de grafos de conocimiento estructurados. Además, incluirá funcionalidades de gestión de usuarios con diferentes niveles de acceso y capacidades de análisis semántico del contenido procesado.

## Metodología

La metodología que se utilizará para el desarrollo de este proyecto de software es scrum, una metodología de desarrollo ágil que permite trabajar el proyecto mediante incrementos, con un enfoque de entregas continuas, a diferencia de una

metodología incremental, scrum enfoca los esfuerzos en el desarrollo, minimizando los esfuerzos de una documentación excesiva.

# Visión general del sistema

En esta sección, se explica de forma general las principales funciones y características del sistema.

## Descripción general

El Sistema de Generación de Grafos de Conocimiento a partir de Noticias es una aplicación web que permitirá a los usuarios procesar contenido noticioso para identificar entidades, relaciones y conceptos clave. El sistema extraerá conocimiento estructurado de textos no estructurados, facilitando el análisis, la visualización y la obtención de insights a partir de grandes volúmenes de noticias.

## Contexto del sistema

El sistema se integra en un ecosistema de análisis de información para:

- Identificar patrones y tendencias en la cobertura noticiosa.
- Descubrir relaciones entre entidades que no son evidentes a simple vista.
- Generar representaciones visuales de información compleja.
- Facilitar la comprensión de un tamaño considerable de volúmenes de noticias.

## Características principales

- Procesamiento automático de noticias mediante PLN.
- Extracción de entidades y relaciones.
- Generación y visualización de grafos de conocimiento.
- Análisis semántico del contenido.
- Gestión de usuarios con diferentes niveles de acceso.
- Exportación de grafos en formatos estándar.

- API REST para integración con otros sistemas.

## Usuarios del sistema

A continuación, se presentan los usuarios con los que operará el sistema, explicando los roles y sus funciones.

### Usuario regular

- **Características**: Usuarios que utilizan el sistema para procesar noticias y generar grafos de conocimiento.
- **Nivel de experiencia**: Conocimiento básico en análisis de datos y familiaridad con interfaces web.
- **Frecuencia de uso**: Uso regular (semanal o diario).
- **Privilegios**: Puede cargar noticias, generar grafos, consultar y exportar resultados.

### Administrador

- **Características:** Usuarios con capacidades de gestión del sistema.
- **Nivel de experiencia:** Conocimiento avanzado en administración de sistemas y gestión de usuarios.
- **Frecuencia de uso:** Uso periódico para tareas de mantenimiento y gestión.
- **Privilegios:** Todos los privilegios de usuario regular, más capacidades de gestión de usuarios, configuración del sistema y monitoreo.

# Requerimientos funcionales

**ID:** AUTH-1

**Nombre del Requerimiento:** Registro de Usuarios.

**Descripción:** Permitir que los usuarios se registren con un nombre, correo electrónico y contraseña.

**Usuario al que Aplica:** Todos los usuarios.

**Precondiciones:** Ninguna.

**Flujo Principal:**

1. El sistema muestra un formulario en el cual el usuario debe ingresar los siguientes datos del alumno:
   - Nombre
   - Correo electrónico
   - Contraseña

2. El usuario completa el formulario con la información solicitada y selecciona Registrar.

3. El sistema valida que el correo electrónico no esté registrado previamente y que los campos sean correctos.

4. Si la información es válida, el sistema almacena los datos del usuario en la base de datos y crea la cuenta del usuario.

5. El sistema redirige a la página de inicio al usuario.

**Flujo Alterno:**

1. El usuario ingresa un correo electrónico ya registrado.
2. El sistema notifica al usuario que el correo ya está en uso y solicita el reingreso de los datos, que sea distintos a los ingresados.

**Postcondiciones:** El usuario queda registrado en el sistema y recibe un correo de confirmación.

**ID:** AUTH-2

**Nombre del Requerimiento:** Inicio de sesión.

**Descripción:** Permitir que los usuarios inicien sesión con su correo electrónico y contraseña.

**Usuario al que Aplica:** Todos los usuarios.

**Precondiciones:** El usuario debe estar registrado.

**Flujo Principal:**

1. El usuario accede a la página de inicio de sesión del sistema.

2. El sistema muestra un formulario donde el usuario debe ingresar los siguientes datos:

   - Correo electrónico.
   - Contraseña.

3. El usuario ingresa sus credenciales y selecciona la opción Iniciar sesión.

4. El sistema valida las credenciales:

- Si el usuario y la contraseña son correctos, el sistema genera un JWT para validar y mantener activa su sesión.
- Según el tipo de usuario, el sistema redirige al dashboard correspondiente dependiendo del rol con el que este cumpla.

5. El usuario accede al sistema y puede realizar las funciones asociadas a su rol.

**Flujo Alterno:**

1. El usuario ingresa credenciales incorrectas.
2. El sistema notifica al usuario que las credenciales son inválidas y solicita el ingreso de credenciales correctas.

**Postcondiciones:** El usuario queda autenticado y recibe un JWT para la sesión.

**ID:** CRD-1

**Nombre del Requerimiento:** Visualización de Información del usuario.

**Descripción:** Permitir que los usuarios vean su propia información una vez hayan iniciado sesión.

**Usuario al que Aplica:** Todos los usuarios.

**Precondiciones:** El usuario debe estar autenticado.

**Flujo Principal:**

1. El usuario accede a su cuenta.
2. El sistema muestra la información del usuario.

**Flujo Alterno:** Ninguno.

**Postcondiciones:** El usuario puede ver su información personal.

**ID:** CRD-2

**Nombre del Requerimiento:** Acceso al dashboard de administrador.

**Descripción:** Permitir que los usuarios con rol de Administrador accedan a un dashboard donde se listan todos los usuarios.

**Usuario al que Aplica:** Administrador.

**Precondiciones:** El usuario debe estar autenticado y tener el rol de Administrador

**Flujo Principal:**

1. El administrador accede a su cuenta.
2. El sistema verifica el rol del usuario.
3. El sistema muestra el dashboard con la lista de todos los usuarios existentes y validados en el sistema.

**Flujo Alterno:**

1. Un usuario sin rol de Administrador intenta acceder al dashboard.
2. El sistema deniega el acceso e indica al usuario la falta de permisos para acceder a dicha sección.

**Postcondiciones:** El administrador puede ver la lista de todos los usuarios.

**ID:** CRD-3

**Nombre del Requerimiento:** Actualización de Información de usuarios por administrador.

**Descripción:** Permitir que los administradores actualicen la información de otros usuarios.

**Usuario al que Aplica:** Administrador.

**Precondiciones:** El administrador debe estar autenticado y tener acceso al dashboard.

**Flujo Principal:**

1. El administrador selecciona un usuario en el dashboard.
2. El administrador actualiza la información del usuario.
3. El sistema valida y guarda los cambios.

**Flujo Alterno:**

1. El correo al que se actualiza ya está en uso
2. El sistema rechaza la solicitud y no realiza ningún cambio

**Postcondiciones:** La información del usuario se actualiza en la base de datos.

**ID:** CRD-4

**Nombre del Requerimiento:** Eliminación de usuarios por un administrador.

**Descripción:** Permitir que los administradores eliminen a otros usuarios.

**Usuario al que Aplica:** Administrador.

**Precondiciones:** El administrador debe estar autenticado y tener acceso al dashboard.

**Flujo Principal:**

1. El administrador selecciona un usuario en el dashboard.
2. El administrador elimina al usuario.
3. El sistema remueve la información del usuario de la base de datos.

**Flujo Alterno:** Ninguno.

**Postcondiciones:** El usuario es eliminado del sistema.

**ID:WG-1**

**Nombre del requerimiento:** Visualizar un listado de grafos

**Descripción:** Permitir visualizar una noticia/grafo mediante un filtro bajo propiedades ya establecidas con la posibilidad de interactuar con este mismo.

**Usuario que aplica:** Administrador, Usuario registrado

**Precondiciones**: El administrador/usuario debe estar autenticado y tener acceso al dashboard

**Flujo principal:**

1. El administrador/usuario selecciona la opción referente a la visualización de grafos
2. Filtra mediante un campo dado la categoría de noticia
3. El sistema valida y retorna las noticias relacionadas con la selección del administrador/usuario

**Flujo alterno:**

1. El usuario selecciona un grafo
2. El sistema valida la opción dada y retorna la información relacionada con el grafo

**Postcondiciones:** El usuario se encuentra bajo el dashboard de un listado de noticias o un grafo relacionado a una noticia en especifico

## Requerimientos no funcionales

**ID:** SEC-1

**Nombre:** Seguridad.

**Descripción:** El sistema debe utilizar JWT para el manejo de sesiones y validación de permisos de usuario para cada acción.

**ID:** CNC-1.

**Nombre:** Concurrencia.

**Descripción:** El sistema debe ser capaz de manejar al menos 1000 usuarios concurrentes en un día sin degradación significativa en el rendimiento (en las condiciones de hardware óptimas).

**ID:** ESC-1.

**Nombre:** Escalabilidad.

**Descripción:** El sistema debe ser escalable para soportar futuras expansiones y aumentos en la base de usuarios, utilizando un enfoque modular en el backend.

**ID:** CMP-1.

**Nombre:** Compatibilidad.

**Descripción:** El frontend debe ser compatible con los navegadores modernos y dispositivos móviles.

**ID:** MNT-1

**Nombre:** Mantenibilidad.

**Descripción:** El código del sistema debe estar bien documentado y seguir buenas prácticas de desarrollo para facilitar su mantenimiento y actualización.

**ID:** USB-1

**Nombre:** Usabilidad

**Descripción:** El sistema debe ser fácil de usar e intuitivo tanto para usuarios regulares como para administradores. Pudiendo acceder al 80% de las funcionalidades con menos de 5 clics a los hipervínculos o botones para acceder a los recursos.

**ID:** DSP-1

**Nombre:** Disponibilidad.

**Descripción:** El sistema debe tener una alta disponibilidad, con un tiempo de inactividad mínimo. Siendo así, que se cumpla una disponibilidad del 99% del tiempo.

**Instituto Politécnico Nacional**

Escuela Superior de Cómputo

# Documento FURPS+

# FURPS+ para un Sistema de Generación de Grafos de Conocimiento a partir de Noticias

**Propósito**: El propósito del sistema es procesar contenido textual (noticias) para generar grafos de conocimiento mediante técnicas de procesamiento de lenguaje natural (PLN). Este sistema está diseñado para identificar entidades, relaciones y conceptos clave, permitiendo una visualización estructurada y procesable del conocimiento.

## Funcionales (F)

1. **Procesamiento de entradas:**
    - Cargar noticias y documentos desde múltiples fuentes (archivos .txt, .pdf (que cumplan con cierta estructura), URLs).
    - Preprocesar textos: eliminación de ruido, stopwords, y aplicación de técnicas de lematización y tokenización.
    - Extraer entidades nombradas (personas, organizaciones, ubicaciones, eventos) y detectar relaciones (por ejemplo, trabaja_en, ubicado_en).
2. **Generación y visualización de grafos:**
    - Crear nodos para cada entidad y establecer aristas para representar relaciones.
    - Exportar grafos en formatos estándar (RDF, JSON-LD, Neo4j).
    - Proveer una interfaz interactiva para explorar los grafos (zoom, arrastre, búsqueda, filtrado por tipo).
3. **Análisis semántico:**
    - Determinar temas principales y sentimientos asociados a las entidades.
    - Generar resúmenes automáticos basados en la estructura del grafo.
    - (Evolución) Incorporar algoritmos de redes neuronales para

optimizar la extracción de relaciones.

4. **Endpoints y Seguridad de la API:**
   - Proveer un conjunto de endpoints REST para interactuar con el sistema. Ejemplos:
     - POST /api/news/upload: Subir archivos o URLs para procesamiento.
     - GET /api/graphs: Obtener grafos generados con filtros por tema, entidad o fecha.
     - GET /api/entities: Listar entidades detectadas en un conjunto de datos.
   - Generar y validar tokens JWT para la autenticación de usuarios.
   - Configurar reglas de autorización y protección de endpoints.
5. **Gestión de usuarios**
   - Registro e Inicio de Sesión
     - Para usuarios regulares y administradores, con campos básicos (nombre, correo electrónico, contraseña).
   - Módulo de Usuarios
     - Visualización de los datos del usuario.
     - Para administradores: listado, actualización y eliminación de usuarios.

## Usabilidad (U)

1. **Interfaz intuitiva:**
   - Diseño adaptable a dispositivos móviles y navegadores modernos.
   - Menús claros y navegación similar a sistemas populares que faciliten el acceso rápido a las funcionalidades.
2. **Interacción con el sistema:**
   - Formularios accesibles y amigables para la carga de archivos, registros, y datos de usuario.
   - Validación automática de formatos y tamaños, mostrando mensajes claros de error o éxito (feedback positivo).

3. **Documentación y Componentes Reutilizables:**
    - Manual de usuario y tutoriales interactivos que guíen al usuario durante el uso del sistema.
    - Uso de componentes gráficos (botones, tarjetas, entradas) consistentes y reutilizables.

## Confiabilidad (R)

1. **Respaldo y recuperación de datos**
    - Implementación de copias de seguridad automáticas de los datos y grafos generados.
    - Mecanismos de recuperación automática en caso de fallos emergentes.
2. **Gestión de errores y excepciones**
    - Logs detallados para detectar fallos en la carga y procesamiento de datos.
    - Reintentos automáticos y manejo robusto de excepciones para evitar el colapso de la aplicación.
3. **Disponibilidad e Integridad**
    - Garantizar un tiempo de actividad del sistema (por ejemplo, 99.9% de disponibilidad).
    - Asegurar que las transacciones críticas (registro, procesamiento, evaluación) mantengan la integridad de la información.

## Rendimiento (P)

1. **Velocidad y eficiencia:**
    - Procesar noticias individuales (500 palabras) en menos de 5 segundos.
    - Generar un grafo para un conjunto de hasta 1,000 noticias en menos de 60 segundos.
2. **Optimización de Consultas y Recursos:**

- Indexación y optimización de consultas a la base de datos para consultas rápidas en grafos grandes.
- Configuración y uso de Docker para asegurar un entorno de ejecución consistente.
- Utilización de herramientas como Gradle para el manejo eficiente de dependencias y compilación.

3. **Gestión de Solicitudes Concurrentes**
    - Capacidad para gestionar múltiples solicitudes simultáneas sin afectar el rendimiento perceptible del usuario.

## Soporte (S)

1. **Compatibilidad y Actualizaciones**
    - Integración con bases de datos de grafos (por ejemplo, Neo4j) y exportación a formatos estándar.
    - Actualizaciones regulares para incorporar mejoras y nuevos modelos de procesamiento de lenguaje natural (PLN).
2. **Mantenimiento y Documentación Técnica**
    - Código modular y reutilizable, siguiendo buenas prácticas (SOLID, revisiones de código).
    - Manuales técnicos para desarrolladores y administradores, y un sistema de tickets para seguimiento de incidencias.
3. **Requerimientos Tecnológicos y de Desarrollo**
    - **Backend:** Java con Spring Boot.
    - **Frontend:** React.
    - **Base de datos:** PostgreSQL.
    - Despliegue en contenedores Docker, compatible con entornos Linux y Windows, y soporte para los navegadores más recientes (Chrome, Firefox, Safari, Edge).
4. **Seguridad y Protección**
    - Implementar autenticación y autorización robusta (JWT, HTTPS, almacenamiento de contraseñas con hash BCrypt).

- Protección contra accesos no autorizados mediante roles diferenciados para usuarios regulares y administradores.

## Extensiones (+)

1. **Cumplimiento normativo:**
   - Asegurar el cumplimiento de normativas de protección de datos (por ejemplo, GDPR y normativas locales).
2. **Ampliación multilingüe:**
   - Capacidad de procesar contenido en múltiples idiomas, ampliando las funcionalidades del análisis semántico.
3. **Conexión con ontologías y Fuentes Externas:**
   - Integración con bases de conocimiento externas como DBpedia o Wikidata para enriquecer la extracción de entidades y relaciones.
4. **Análisis avanzado y Machine Learning:**
   - Incorporar algoritmos de redes neuronales para mejorar la precisión en la extracción y análisis de relaciones, ofreciendo análisis predictivos y recomendaciones basadas en datos.

**Instituto Politécnico Nacional**

Escuela Superior de Cómputo

# Casos de uso

# Diagramas de casos de uso

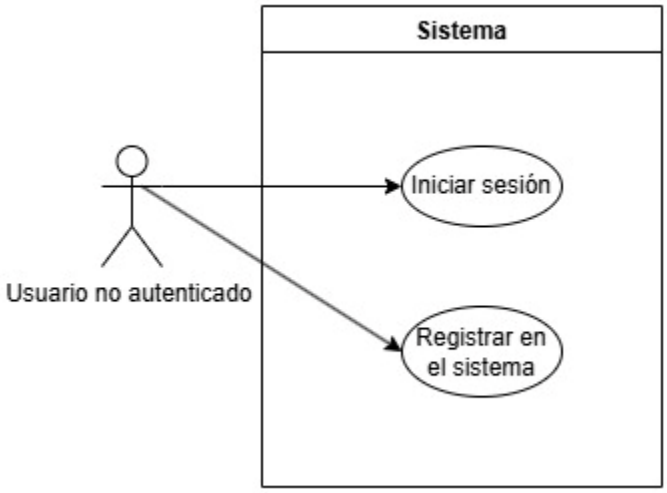


*Figura 1. Diagrama de casos de uso de autenticación.*

En la Figura 1 se muestra el diagrama de casos de uso referente a la autenticación de los usuarios, ya sea registro o inicio de sesión, en las tablas Tabla 1 y Tabla 2 se desglosa la especificación de los casos de uso.

| | | |
|---|---|---|
| Nombre CU: | Registrar en el sistema | |
| Dependencias | Usuario no autenticado | |
| Descripción | Permitir el registro de un usuario dentro del sistema | |
| Precondición | El usuario no debe tener las mismas credenciales previamente registradas | |
| | | |
| Secuencia normal | Paso | Acción |
| | 1. | El usuario accede a la función aludida al registro de un usuario |
| | 2. | El sistema muestra un formulario en el cual el usuario debe ingresar los siguientes datos |

| | | |
|---|---|---|
| | 3. | El usuario llena los campos para su posterior envió |
| | 4. | El sistema valida las credenciales con datos iguales a los ingresados |
| | 5. | El sistema almacena los datos del usuario en la base de datos y crea un usuario |
| | 6. | El sistema redirige a la página de inicio del usuario |
| | | |
| Postcondiciones | El usuario queda registrado dentro del sistema, su dashboard actual es el inicio de este usuario además de recibir un correo de confirmación | |
| Flujo alterno | Paso | Acción |
| | 6. | El sistema detecta campos duplicados con la información dada y la base de datos |
| | 7. | Permite reingresar datos del usuario |
| | | |
| Comentarios | | |
| CU necesario para el login del sistema para un posterior uso sobre la interacción con el resto de las funcionalidades del sistema | | |

*Tabla 1. Tabla de especificación de casos de uso para el registro en el sistema.*

| | | |
|---|---|---|
| | | |
| Nombre CU: | Inicio sesión | |
| Dependencias | Usuario no autenticado | |
| Descripción | Permitir que los usuarios inicien sesión con su correo electrónico y contraseña. | |
| Precondición | El usuario debe de estar previamente registrado dentro del sistema | |
| | | |
| | Paso | Acción |

| | | |
|---|---|---|
| Secuencia normal | 1. | El usuario ingresa al inicio de sesión de la pagina |
| | 2. | El sistema muestra los campos a llenar relacionados con el inicio de sesión |
| | 3. | El usuario llena los campos para su posterior envió |
| | 4. | El sistema valida las credenciales dadas por el usuario |
| | 5. | El sistema crea la sesión |
| | | |
| Postcondiciones | El usuario es redirigido a la pantalla principal del sistema | |
| Flujo alterno | Paso | Acción |
| | 6. | El sistema devuelve la inconsistencia de las credenciales |
| | 7. | Permite reingresar credenciales del usuario |
| | | |
| Comentarios | | |
| CU necesario para el inicio del sistema hacia la realización del proceso involucrado hacia la interacción con el resto de las funcionalidades del sistema | | |

*Tabla 2. Tabla de especificación de casos de uso para el inicio de sesión en el sistema.*

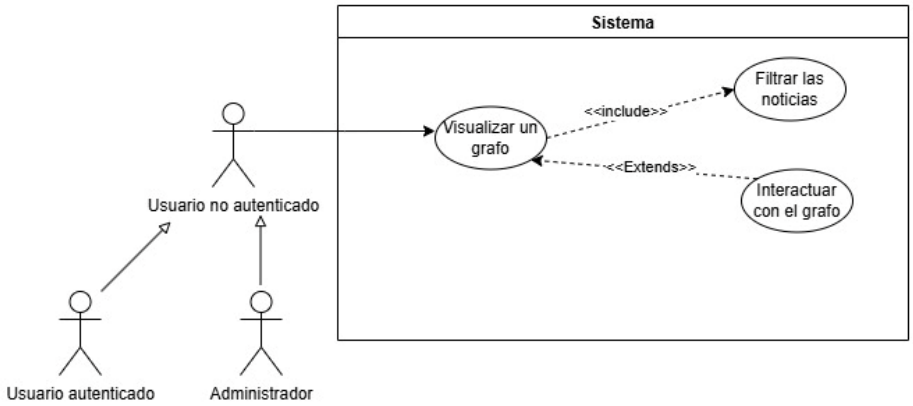


*Figura 2. Diagrama de casos de uso para la visualización del grafo de conocimiento.*

En la Figura 2 se muestra el caso de uso referente a la visualización de un grafo de conocimiento, la especificación de este caso de uso se explica en la Tabla 3.

| | | |
|---|---|---|
| Nombre CU: | Visualizar grafo | |
| Dependencias | Usuario autenticado, administrador y usuario no autenticado | |
| Descripción | Permitir visualizar el grafo desplegado con los detalles relacionados sobre una noticia | |
| Precondición | El usuario debe estar previamente registrado en el sistema, así como haber ingresado al mismo | |
| | | |
| Secuencia normal | Paso | Acción |
| | 1. | El usuario debe acceder a la acción aludida a la visualización de una noticia |
| | 2. | El sistema muestra los campos a llenar para el filtrado de la información |
| | 3. | El usuario llena la información involucrada para el filtrado, así como él envió de esta |
| | 4. | El sistema arroja las noticias adecuadas con los campos llenados |
| | | |
| Postcondiciones | El usuario podrá visualizar un listado de las noticias adecuadas a los campos llenados | |
| Flujo alterno 1 | Paso | Acción |
| | 3. | Usuario selecciona una noticia para su indagación |
| | 4. | El sistema devuelve el grafo involucrado con la noticia |
| | | |
| Comentarios | | |

| |
|---|
| CU enfocado a la visualización de un listado de noticias, con base a la petición de un usuario, así como la indagación de una noticia en concreto |

*Tabla 3. Especificación del caso de uso para la visualización de un grafo de conocimiento.*

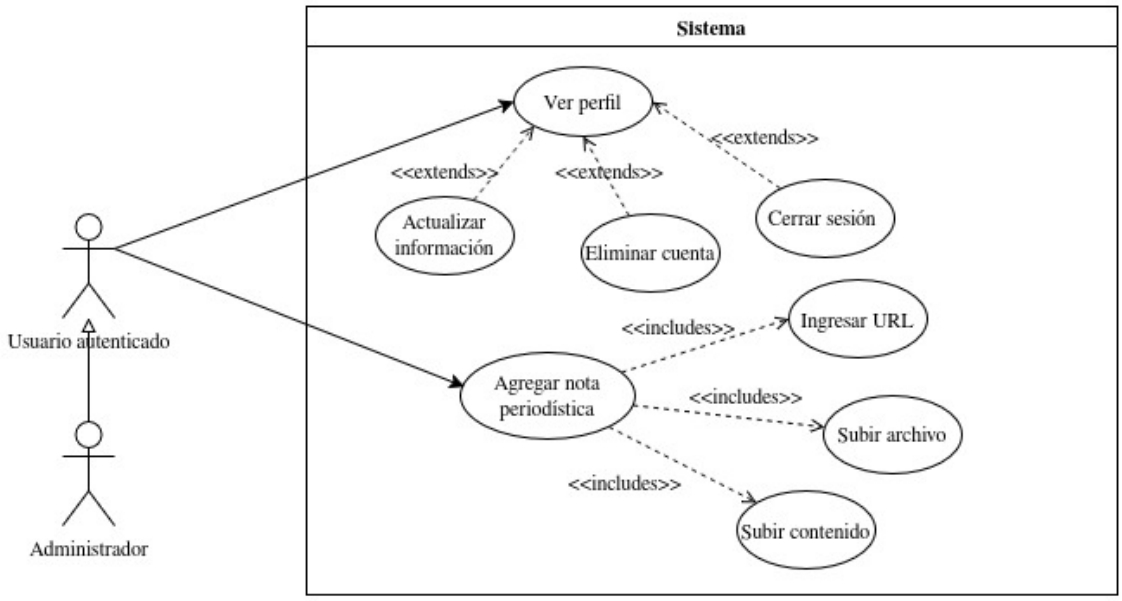


*Figura 3. Diagrama de casos de uso para las acciones de un usuario autenticado.*

En la Figura 3 se muestran los casos de uso para un usuario autenticado, así como la relación de herencia entre usuarios. En las tablas Tabla 4 y Tabla 5 se muestra la especificación de casos de uso de los presentados en el diagrama.

| | | |
|---|---|---|
| Nombre CU: | Agregar una nota periodística | |
| Dependencias | Administrador, usuario autenticado | |
| Descripción | Permitir que los usuarios puedan subir una URL con la cual se desarrollara un grafo | |
| Precondición | Ninguna | |
| | | |
| Secuencia normal | Paso | Acción |
| | 1. | El usuario debe acceder a la acción aludida al envió de una nota periodística |

| | | |
|---|---|---|
| | 2. | El sistema muestra los campos a llenar |
| | 3. | El usuario llena los campos para su posterior envió |
| | 4. | El sistema valida la información dada por el usuario |
| | 5. | El sistema crea el grafo relacionado |
| | | |
| Postcondiciones | La noticia se encontrará dentro de la base de datos para su posterior procesamiento e interacción | |
| Flujo alterno | Paso | Acción |
| | 6. | El sistema devuelve la inconsistencia la información proporcionada |
| | 7. | Permite reingresar los datos al usuario |
| | | |
| Comentarios | | |
| CU necesario para el llenado del sistema por parte de los usuarios | | |

*Tabla 4. Especificación de caso de uso para el agregado de una nota periodística.*

| | | |
|---|---|---|
| | | |
| Nombre CU: | Visualizar perfil | |
| Dependencias | Usuario autenticado | |
| Descripción | Permitir visualizar la información relacionada con el usuario una vez ya iniciada la sesión | |
| Precondición | El usuario debe estar previamente registrado en el sistema, así como dentro de este | |
| | | |
| Secuencia normal | Paso | Acción |
| | 1. | El usuario debe acceder a la acción aludida a la visualización de su perfil |
| | 2. | El sistema muestra la información relacionada con el mismo |

| | | |
|---|---|---|
| Postcondiciones | El usuario podrá visualizar la información relacionada con su perfil | |
| Flujo alterno 1 | Paso | Acción |
| | 3. | usuario selecciona la opción aludida a la modificación del perfil |
| | 4. | El sistema devuelve los campos a ser modificados |
| | 5. | El usuario ingresa la información a los campos |
| | 6. | El sistema la validez de los campos |
| | 7. | El sistema retorna un mensaje del estatus de los cambios |
| Flujo alterno 2 | 3. | El usuario selecciona la opción aludida a la eliminación del perfil |
| | 4. | El sistema devuelve una pantalla de confirmación sobre la eliminación del perfil |
| | 5. | El usuario acciona la confirmación o la cancelación de esta acción |
| | 6. | El sistema retorna un mensaje de confirmación en caso de seleccionarse la eliminación y da de baja al usuario |
| Flujo alterno 3 | 3. | El usuario selecciona la opción aludida al cierre de sesión |
| | 4. | El sistema muestra una confirmación sobre la acción |
| | 5. | El usuario confirma dicha acción |
| | 6. | El sistema cierra la sesión |
| Comentarios | | |

| |
|---|
| CU enfocado a la visualización relacionada con el usuario y modificación de este |

*Tabla 5. Especificación de caso de uso para la visualización del grafo de conocimiento.*

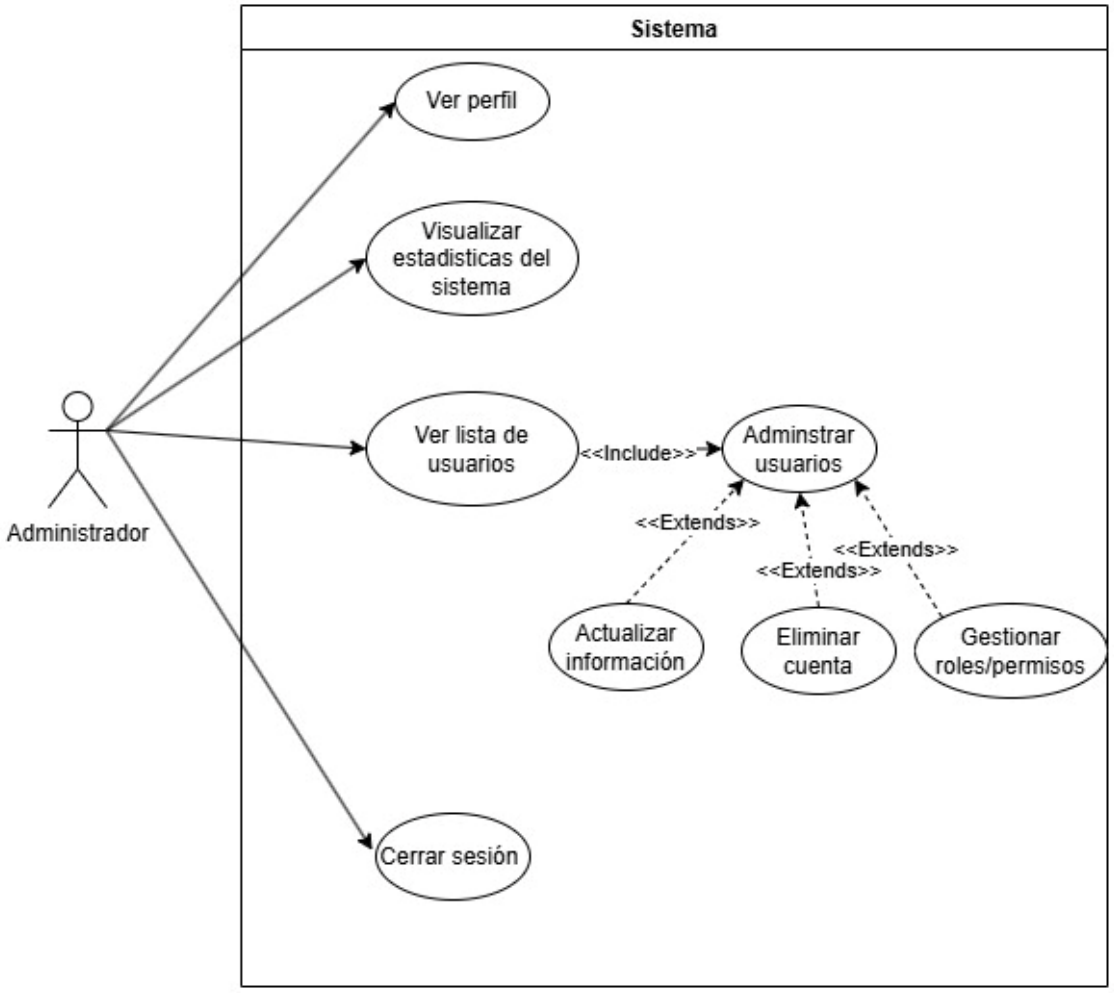


*Figura 4. Diagrama de casos de uso para un administrador.*

En la Figura 4 se muestra el diagrama de casos de uso para un actor administrador, mostrando las acciones que puede realizar. En las tablas Tabla 6 y Tabla 7 se especifican los casos de uso más importantes de este.

| | |
|---|---|
| Nombre CU: | Visualizar estadísticas del sistema |
| Dependencias | Administrador |

| | | |
|---|---|---|
| Descripción | Permitir visualizar un dashboard relacionado con las estadísticas del sistema referentes a noticias y usuarios | |
| Precondición | El usuario debe estar previamente registrado en el sistema así tener un rol de administrador | |
| | | |
| Secuencia normal | Paso | Acción |
| | 1. | El usuario debe acceder a la acción aludida a la visualización de estadísticas del sistema |
| | 2. | El sistema muestra un dashboard referente a las estadísticas del sistema involucradas con usuarios y noticias |
| | | |
| Postcondiciones | El usuario podrá visualizar un dashboard con las estadísticas generales del sistema | |
| Flujo alterno 1 | Paso | Acción |
| | ------ | Ninguna |
| | | |
| Comentarios | | |
| CU enfocado a la visualización de estadísticas referentes a usuarios, así como noticias | | |

*Tabla 6. Especificación de casos de uso para la visualización de estadísticas.*

| | |
|---|---|
| | |
| Nombre CU: | Visualizar ver lista de usuarios |
| Dependencias | Administrador |
| Descripción | Permitir visualizar los usuarios dentro del sistema |
| Precondición | El usuario debe estar previamente registrado en el sistema, así como tener rol de administrador |
| | |

| | | |
|---|---|---|
| Secuencia normal | Paso | Acción |
| | 1. | El usuario debe acceder a la acción aludida a la visualización de usuarios dentro del sistema |
| | 2. | El sistema muestra un dashboard dedicado al despliegue de los usuarios dentro del sistema |
| | 3. | El usuario tiene un dashboard con la capacidad de modificar usuarios |
| | | |
| Postcondiciones | El usuario podrá visualizar un dashboard con los usuarios del sistema | |
| Flujo alterno 1 | Paso | Acción |
| | 4. | El usuario selecciona la opción aludida a la actualización de información de un usuario en especifico |
| | 5. | El sistema muestra los campos con la información del usuario a ser cambiada |
| | 6. | El usuario modifica los campos para su posterior envió |
| | 7. | El sistema valida y guarda los cambios |
| Flujo alterno 2 | 4. | El usuario selecciona la opción aludida a dar de baja un perfil un perfil en especificad |
| | 5. | El sistema da de baja al usuario seleccionado |
| | 6. | El sistema arroja una ventana de confirmación |
| Flujo alterno 3 | 4. | El usuario selecciona la opción aludida a modificar el rol de un usuario especifico |
| | 5. | El sistema arroja un dropdown con las opciones disponibles |

| | | |
|---|---|---|
| | 6. | El usuario selecciona el rol y confirma el cambio |
| | 7. | El sistema arroja una ventana de confirmación |
| | | |
| Comentarios | | |
| CU enfocado a la visualización de los usuarios, así como su modificación | | |

*Tabla 7. Especificación de caso de uso para la administración de usuarios.*

## Caso de uso principal

El caso de uso principal para asegurar un correcto funcionamiento de nuestro sistema es subir las notas periodísticas al sistema, de tal forma que se deje todo preparado a la API de procesamiento de lenguaje natural. Se considero como caso de uso principal por las funcionalidades que se ofrecen, además de ser una funcionalidad base para el proyecto, con base en la implementación de este, se podrá realizar de forma más sencilla la implementación de otros casos de uso.

**Instituto Politécnico Nacional**

Escuela Superior de Cómputo

# Guía técnica

# Guía técnica

En esta sección, se describen las herramientas tecnológicas utilizadas para el proyecto, describiendo a detalle su aportación y a su vez, dando una guía de instalación del proyecto.

## Configuración de la base de datos

La aplicación Graph News utiliza PostgreSQL como sistema de gestión de base de datos relacional. La configuración busca implementar buenas prácticas de seguridad para el manejo de credenciales de accesos empleando variables de entorno y un sistema de entidades JPA.

### *URL de Conexión*

- **Propiedad:** spring.datasource.url=${DATABASE_URL}
- **Descripción:** Define la URL de conexión a la base de datos PostgreSQL.
- **Implementación:** Utiliza una variable de entorno DATABASE_URL que típicamente sigue el formato jdbc:postgresql://<host>:<port>/<database_name>.
- **Ejemplo en desarrollo:** jdbc:postgresql://db:5432/crud (según el docker-compose.yml)

### *Credenciales de Acceso*

- **Propiedades**:
  - spring.datasource.username=${DATABASE_USER}
  - spring.datasource.password=${DATABASE_PASSWORD}
- **Descripción**: Establecen el usuario y contraseña para autenticarse con la base de datos.
- **Implementación**: Utilizan variables de entorno para evitar hardcodear credenciales en el código.

## Configuración de JPA/Hibernate

- **Propiedad:** spring.jpa.hibernate.ddl-auto=update

- **Descripción:** Controla cómo Hibernate maneja el esquema de base de datos.
- **Comportamiento:** Con el valor update, Hibernate:
    - Actualiza automáticamente el esquema cuando detecta cambios en las entidades
    - Crea tablas si no existen
    - No elimina datos existentes ni tablas no referenciadas
    - Adecuado para entornos de desarrollo, pero debe usarse con precaución en producción

### *Driver y Dialecto*

- **Propiedad:** spring.datasource.driver-class-name=org.postgresql.Driver
- **Descripción:** Especifica el driver JDBC para PostgreSQL.
- **Propiedad:** spring.jpa.database-platform = org.hibernate.dialect.PostgreSQLDialect
- **Descripción:** Define el dialecto SQL que Hibernate debe usar.
- **Importancia:** Permite a Hibernate generar SQL optimizado específicamente para PostgreSQL.

## Control de versiones

- **Git**: Git es la herramienta que sirve para llevar un correcto control de versiones, teniendo una gran forma de optimizar el tamaño de todos los cambios, así como contar con implementaciones extra que sirven para el trabajo colaborativo.
- **GitHub**: Es la plataforma remota o en la nube que almacena el código fuente del sistema, además que cuenta con herramientas útiles para la creación del sistema.

# Configuración del sistema de seguridad (Spring Security)

Sistema usado por medio de JWT

Dentro del archivo JWTAuthenticationFilter.java se define un filtro ejecutado una vez por solicitud hacia la validación de un token

## Servicio de detalles de usuario

La clase UserDetailsServiceImplementation.java extiende la interfaz estándar de Spring Security que permite que el sistema de seguridad trabaje directamente con las entidades de la aplicación. Algunas de las facilidades que nos proporciona este servicio son:

- Cargar información detallada del usuario desde la base de datos
- Transformar el modelo de dominio User en el objeto UserDetails que Spring Security reconoce
- Facilitar la verificación de credenciales durante el proceso de autenticación

## Unidades de la gestión JWT

La utilidad de JWT, se da por medio del archivo JwtUtil.java, que contiene métodos para generar y validar tokens JWT, así como para extraer información de estos, ofreciendo algunos métodos para:

- Generar nuevos tokens con información personalizada (claims)
- Validar la autenticidad y vigencia de los tokens existentes
- Extraer información específica contenida dentro de los tokens
- Manejar la firma criptográfica de los tokens mediante algoritmos seguros

## Configuraciones de Seguridad.

La configuración de seguridad se define en el archivo SecurityConfig.java, pues aquí se especifican las reglas de autorización para los endpoints de acceso público, acceso para administradores y los accesos para el servicio de autenticación:

## Roles, permisos y endpoints protegidos

Aquí, se definen los diferentes roles y endpoints disponibles en el sistema. Así mismo, se especifica su verbo de petición, una breve descripción y el tipo de acceso que este tiene que puede ser de tipo público (para cualquier usuario), autenticado (requiere de una sesión en el sistema) o de tipo ADMIN (solamente usuarios registrados con el rol de 'ADMIN' pueden acceder).

## Roles:

- **USER**: Usuario regular.
- **ADMIN**: Administrador del sistema.

## EndPoints:

*User endpoints:*

1. /api/user/login
    - **Método**: POST
    - **Descripción**: Permite a un usuario iniciar sesión.
    - **Permisos**: Público.
2. /api/user/signup
    - **Método**: POST
    - **Descripción**: Permite a un nuevo usuario registrarse.
    - **Permisos**: Público.

3. /api/user/me

- **Método**: GET
- **Descripción**: Obtiene la información del usuario autenticado.
- **Permisos**: Autenticado.

4. /api/user/{id}

- **Método:** GET
- **Descripción:** Obtiene la información de un usuario específico por su ID.
- **Permisos:** Autenticado y el usuario debe ser el propietario de la cuenta.

5. /api/user/all

- **Método:** GET
- **Descripción:** Obtiene la lista de todos los usuarios.
- **Permisos:** ADMIN.

6. /api/user/{id}

- **Método:** DELETE
- **Descripción:** Elimina un usuario específico por su ID.
- **Permisos:** ADMIN.

7. /api/user/{id}

- **Método:** PUT
- **Descripción:** Actualiza la información de un usuario específico por su ID.

- **Permisos:** ADMIN.

8. /api/user/update/me

- **Método:** PUT
- **Descripción:** Actualiza la información pública del usuario propietario de la cuenta especificado por su ID.
- **Permisos:** Autenticado.

9. /api/user/update/me/password

- **Método:** PUT
- **Descripción:** Actualiza la información sensible (contraseña) del usuario propietario de la cuenta especificado por su ID.
- **Permisos:** Autenticado.

10. /api/user/debug

- **Método:** GET
- **Descripción:** Obtiene los datos del usuario actual que se encuentra navegando en el sistema y sus permisos, únicamente se muestra a un super usuario.
- **Permisos:** ADMIN.

11. /api/user/image/{id}

- **Método:** GET
- **Descripción:** Obtiene la foto de perfil de un usuario específico por su ID.
- **Permisos:** Autenticado y el usuario debe ser el propietario de la cuenta.

12. /api/user/create/user

- **Método:** POST
- **Descripción:** Permite a un super usuario realizar el registro de un usuario distinto.
- **Permisos:** ADMIN.

13. /api/user/delete/me

- **Método:** DELETE
- **Descripción:** Elimina un usuario específico por su ID.
- **Permisos:** Autenticado y el usuario debe ser el propietario de la cuenta.

*News endpoints:*

1. /api/news/upload/url

- **Método**: POST
- **Descripción**: Permite extraer y guardar contenido de noticias desde una URL.
- **Permisos**: Autenticado.

2. /api/news/upload/file

- **Método**: POST
- **Descripción**: Crea una noticia a partir de un archivo (PDF, DOCX, TXT).
- **Permisos**: Autenticado.

3. /api/news/upload/content

- **Método**: POST

- **Descripción**: Crea una noticia a partir de contenido de texto proporcionado directamente.
- **Permisos**: Autenticado.

4. /api/news
   - **Método**: GET
   - **Descripción**: Devuelve una lista paginada de todas las noticias disponibles en el sistema.
   - **Permisos**: Autenticado.
5. /api/news/{id}
   - **Método**: GET
   - **Descripción**: Obtiene una noticia específica por su identificador único.
   - **Permisos**: Autenticado.
6. /api/news/user/{userId}
   - **Método**: GET
   - **Descripción**: Obtiene todas las noticias creadas por un usuario específico por su identificador único.
   - **Permisos**: Autenticado.
7. /api/news/user/{userId}/paged
   - **Método**: GET
   - **Descripción**: Obtiene la versión paginada del endpoint anterior.
   - **Permisos**: Autenticado.

8. /api/news/search

- **Método**: GET
- **Descripción**: Permite buscar noticias que contengan un texto específico en el título o contenido.
- **Permisos**: Autenticado.

9. /api/news/latest

- **Método**: GET
- **Descripción**: Obtiene las últimas noticias añadidas al sistema.
- **Permisos**: Autenticado.

10. /api/news/date-range

- **Método**: GET
- **Descripción**: Filtra noticias creadas dentro de un intervalo de tiempo específico.
- **Permisos**: Autenticado.

11. /api/news/{ID}

- **Método**: DELETE
- **Descripción**: Elimina una noticia específica por su ID. Solo el autor original de la noticia o un usuario con rol de administrador pueden realizar esta operación.
- **Permisos**: Autenticado.

## Permisos

**Acceso público:** Estos únicamente incluyen los endpoints para efectuar solicitudes sin una autenticación necesariamente, así como la obtención de tokens.

- Registro de nuevo usuarios (POST /api/user/signup)
- Acceso al sistema (POST/api/user/login)

**Acceso administrativo**: Las operaciones de gestión de usuarios como:

- Listado de todos los usuarios (GET /api/user/all)
- Eliminación de usuarios (DELETE /api/user/**)
- Modificación de usuarios (PUT /api/user/**) están restringidas exclusivamente a usuarios con rol ADMIN.

**Acceso autenticado**: Todos los demás endpoints de la aplicación requieren que el usuario esté autenticado, independientemente de su rol específico.

# Archivos relacionados

En esta sección se listan los archivos relacionados que no se involucran directamente con las funcionalidades del negocio.

## application.properties

El archivo aplication properties contiene configuraciones principales para la aplicación Spring Boot, como la configuración de la base de datos, el puerto del servidor, y las propiedades de seguridad JWT, por ende, tiene por tarea el definir cómo la aplicación se conecta a la base de datos y maneja la autenticación y autorización.

### Conexión a la base de datos:

- Se emplean variables de entorno como: (${DATABASE_URL}, ${DATABASE_USER}, ${DATABASE_PASSWORD}) para las credenciales de PostgreSQL.

- Se configura el Driver para PostgreSQL con spring.datasource.driver-class-name=org.postgresql.Driver.
- Dialecto Hibernate para trabajar con PostgreSQL en spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect.
- La propiedad spring.jpa.hibernate.ddl-auto=update permite actualizar el esquema automáticamente.

**Puerto del Servidor:**

- Configurado dinámicamente con server.port=${PORT}.

**Seguridad JWT:**

- Secret key y tiempo de expiración configurados mediante variables de entorno.
- security.jwt.secret=${JWT_SECRET} para la clave secreta de firma.
- security.jwt.expiration=${JWT_EXPIRATION} para el tiempo de validez del token.

## docker-compose.yml

El archivo Docker-compose.yml tiene por función el definir los servicios Docker requeridos para la aplicación, incluyendo la base de datos PostgreSQL y el backend de la aplicación, correspondiendo al entorno de producción.

- Las variables de entorno se definen en docker-compose.yml para el entorno de desarrollo.
- Incluye credenciales de base de datos, puerto, secret JWT y tiempo de expiración.

## build.gradle

El archivo build. gradle.kts define las dependencias y configuraciones de Gradle para el proyecto backend, este incluye las dependencias necesarias para Spring

Boot, seguridad, JWT, y PostgreSQL, así como configuraciones específicas de compilación y pruebas, gestionando las bibliotecas y plugins que la aplicación necesita para funcionar.

## SecurityConfig.java

SecurityConfig.java se encarga de configurar la seguridad de Spring Security, incluyendo la configuración de los filtros JWT y las reglas de autorización, es decir, define qué endpoints son accesibles para cada rol y cuáles requieren autenticación, asegurando que solo los usuarios autorizados puedan acceder a ciertas partes de la aplicación.

**Autenticación y Autorización:**

- Implementa SecurityFilterChain para definir reglas de acceso.
- Se utiliza CSRF en modo deshabilitado a través de la instrucción: csrf(AbstractHttpConfigurer::disable).
- Se configura CORS para permitir solicitudes desde http://localhost:5173, esto para que la aplicación cliente (react) pueda consumir los endpoints definidos.

**Reglas de Acceso por Endpoint:**

- Endpoints públicos: POST /api/user/signup y POST /api/user/login.
- Endpoints restringidos para administradores:
    - GET /api/user/all
    - DELETE /api/user/**
    - PUT /api/user/**

- Todas las demás rutas requieren autenticación.

**Integración JWT:**

- Se agrega JWTAuthenticationFilter en la cadena de filtros antes de UsernamePasswordAuthenticationFilter.

**Encriptación de Contraseñas:**

- Se utiliza BCryptPasswordEncoder para el hashing seguro de contraseñas.

**Gestor de Autenticación:**

- Se configura un 'bean' AuthenticationManager personalizado que utiliza el servicio de detalles de usuario y el codificador de contraseñas para gestionar las autenticaciones del usuario.

## JwtUtil.java

Este archivo se encarga de la generación, validación y extracción de la información de tokens firmados con HMAC-SHA256 que incluye los claims y detalles como el email de los usuarios registrados, las fechas en que se crean y expiran los tokens. Además, incluye métodos para extraer información específica de los usuarios.

**Generación de Tokens:**

- generateToken(): Se utiliza para crea un token con claims, email del usuario, fecha de emisión y expiración.
    - Utiliza HMAC-SHA256 para firmar los tokens.

**Validación de Tokens:**

- isTokenValid(): Verifica que el email en el token coincida y que no haya expirado.
- isTokenExpired(): Comprueba si la fecha de expiración es anterior a la actual.

**Extracción de Información:**

- extractClaim(): Permite extraer claims específicos del token.
- extractAllClaims(): Obtiene todos los claims del token.

**Configuración de Seguridad:**

- Secret key y tiempo de expiración se inyectan desde las propiedades de la aplicación, definidas en el archivo: “application.properties”
- La clave se codifica en Base64 y se utiliza para crear un SecretKeySpec.

## CORS (Cross-Origin Resource Sharing)

Se implementa directamente en SecurityConfig.java para permitir el intercambio de recursos entre el backend y el frontend.

- **Accesos**: Permite solicitudes desde http://localhost:5173 (origen del frontend en desarrollo).
- **Métodos permitidos**: GET, POST, PUT, DELETE.
- **Cabeceras**: Permite todas las cabeceras con setAllowedHeaders(List.of("*")).
- **Credenciales**: Habilita las credenciales con setAllowCredentials(true).

## Variables de Entorno

**Docker-Compose**:

- Se definen en docker-compose.yml para el entorno de desarrollo.
- Incluye credenciales de base de datos, puerto, secret JWT y tiempo de expiración.

**Application Properties**:

- En el archivo application.properties referencia las variables de entorno con la sintaxis ${VARIABLE}.

**Seguridad:**

- JWT Secret en base64: c2VjcmV0S2V5MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTIzNA.
- Tiempo de expiración configurado en milisegundos (24 horas: 86400000).

**Contenerización:**

- El Dockerfile construye la aplicación y la ejecuta sin exponer directamente las variables sensibles en la imagen.

**Instituto Politécnico Nacional**

Escuela Superior de Cómputo

# Apéndices

# Apéndice A: Glosario de términos

**API:** Application Programming Interface, conjunto de definiciones y protocolos para construir e integrar software.

**Entidad:** Objeto o concepto del mundo real que puede ser identificado y representado (ej. persona, lugar, organización).

**Grafo de conocimiento:** Estructura de datos que representa entidades y sus relaciones en forma de nodos (entidades) y aristas (relaciones).

**JSON-LD:** JSON for Linking Data, método para codificar datos enlazados usando JSON.

**JWT:** JSON Web Token, estándar para la creación de tokens de acceso.

**Lematización:** Proceso de reducir una palabra a su forma base o raíz.

**Neo4j:** Base de datos de grafos de código abierto.

**PLN:** Procesamiento de Lenguaje Natural, conjunto de técnicas computacionales para analizar y representar textos en lenguaje humano.

**RDF:** Resource Description Framework, estándar para la representación de información en la web.

**Relación:** Conexión o asociación entre dos o más entidades.

**Stopwords:** Palabras comunes que aportan poco valor semántico (ej. "el", "la", "un").

**Tokenización:** Proceso de dividir un texto en unidades básicas como palabras o frases.