Instituto Politécnico Nacional

ESCOM

UA: Ingeniería de Software

Profesor: Gabriel Hurtado Avilés

Torres Ruiz Axel Alejandro (2022630178)

Flores Rangel Mauricio Alberto (2020630114)

Grupo: 6CV4

Contenido

Introducción

Objetivo

El objetivo de este documento es definir de manera detallada los requerimientos del sistema, proporcionando una guía clara para su desarrollo e implementación. Se describen las funcionalidades clave, restricciones, actores y casos de uso para asegurar una comprensión integral del sistema por parte de todos los involucrados.

Contexto

El sistema está diseñado para proporcionar a los usuarios una plataforma que permita la gestión de autenticación, búsqueda y recomendación de artículos científicos. Utiliza APIs externas para obtener metadatos y proporciona herramientas avanzadas de filtrado y personalización. La seguridad del sistema se basa en Spring Security, garantizando un acceso seguro y un control adecuado de sesiones.

Glosario

Autenticación: Proceso de validación de identidad de un usuario.

Endpoint: Punto de acceso de la API para interactuar con el sistema.

CRUD: Acrónimo de Create, Read, Update y Delete. Representa las operaciones básicas para la gestión de datos en una base de datos o sistema

Requerimientos Funcionales y No Funcionales

Requerimientos Funcionales

ID	Nombre del requerimiento	Descripción	Criterios de Aceptación	Prioridad	Dependencias
F-01	Registro de Usuario	Permite a nuevos usuarios registrarse	Email único, contraseña segura, rol por defecto.	Alta	---
F-02	Inicio de Sesión	Los usuarios pueden autenticarse con su email y contraseña.	Validación contra base de datos, redirección según rol.	Alta	F-001
F-03	Panel de Administración	Los administradores pueden gestionar usuarios.	CRUD de usuarios permitido solo para administradores.	Alta	F-02
F-04	Búsqueda de artículos	Permitir a los usuarios buscar artículos por palabras clave, autor, año o tema.	El usuario puede ingresar criterios de búsqueda y obtener resultados relevantes.	Alta	---
F-05	Filtros avanzados	Implementar filtros por idioma, revista o tipo de artículo.	Se pueden aplicar múltiples filtros simultáneamente y los resultados se actualizan en tiempo real.	Media	F-04
F-06	Visualización de artículos	Mostrar detalles clave del artículo (título, autores, resumen, año, enlace).	Los datos deben presentarse de manera clara y organizada.	Alta	F-04
F-07	Visor embebido	Permitir la visualización de documentos PDF dentro del sistema.	Solo debe mostrarse si el documento está en acceso abierto.	Media	F-06
F-09	Recomendación personalizada	Ofrecer recomendaciones basadas en historial de búsqueda y favoritos.	Las recomendaciones deben actualizarse dinámicamente.	Alta	F-04, F-06
F-10	Historial y favoritos	Registrar el historial de búsqueda y permitir marcar artículos favoritos.	El usuario puede acceder a su historial y gestionar favoritos.	Alta	F-04, F-06
F-11	Integración con APIs	Conectar el sistema con APIs como CrossRef y Semantic Scholar.	Las respuestas en JSON deben ser correctamente procesadas.	Alta	---

Requerimientos No Funcionales

Usabilidad (U)

- **Interfaz amigable**: Diseño minimalista y responsivo para facilitar la navegación en dispositivos móviles y escritorio.
- **Accesibilidad**: Compatibilidad con lectores de pantalla y navegación por teclado.
- **Experiencia de usuario**: Implementación de tutoriales interactivos y mensajes de error claros.

8.2 Confiabilidad (R)

- **Disponibilidad**: Garantizar un tiempo de actividad del 99.9% mediante infraestructura en la nube.
- **Gestión de errores**: Manejo robusto de fallos en APIs externas y validación estricta de datos de entrada.
- **Seguridad**: Implementar autenticación segura con tokens y cifrado de datos sensibles, protegiendo contra ataques como inyección SQL y fuerza bruta.

8.3 Rendimiento (P)

- **Tiempos de respuesta**: Consultas a la base de datos y generación de recomendaciones en menos de 3 segundos.
- **Escalabilidad**: Capacidad para soportar hasta 10,000 usuarios concurrentes con optimización de recursos.
- **Eficiencia**: Uso de técnicas de caché y procesamiento distribuido para mejorar el rendimiento.

8.4 Soporte y Mantenimiento (S)

- **Mantenibilidad**: Código modular y documentado para facilitar futuras actualizaciones.
- **Compatibilidad**: Integración con APIs académicas estándar como CrossRef y Semantic Scholar.
- **Actualizaciones**: Ciclos regulares de mantenimiento y mejoras basadas en el feedback del usuario.

Casos de Uso

Core Search & User Management Documentation

1. Reglas de negocio asociadas a cada entidad

- **Users**:
  - Los usuarios deben tener cuentas únicas identificadas por un correo electrónico o un identificador único (UUID).
  - Los datos de los usuarios deben ser siempre consistentes y cumplir con las políticas de privacidad y seguridad.
  - Solo usuarios con permisos específicos pueden acceder a ciertas funciones de búsqueda avanzadas.
- **Searches**:
  - Cada búsqueda realizada debe registrar quién la ejecutó y en qué momento.
  - Las búsquedas deben estar optimizadas para tiempos de respuesta rápidos.
  - Se debe verificar que el usuario tenga acceso a los resultados antes de mostrarlos (según rol y permisos).

2. Invariantes del sistema

- **Users**:
  - Todos los usuarios activos deben tener roles asignados (por ejemplo: Admin, Premium, Regular).
  - No puede existir más de un usuario con el mismo correo electrónico.
- **Searches**:
  - Cada búsqueda debe estar asociada a un usuario.
  - El sistema debe garantizar que las búsquedas no excedan los límites de la API de terceros.

3. Diccionario de Datos

Entidad: Users

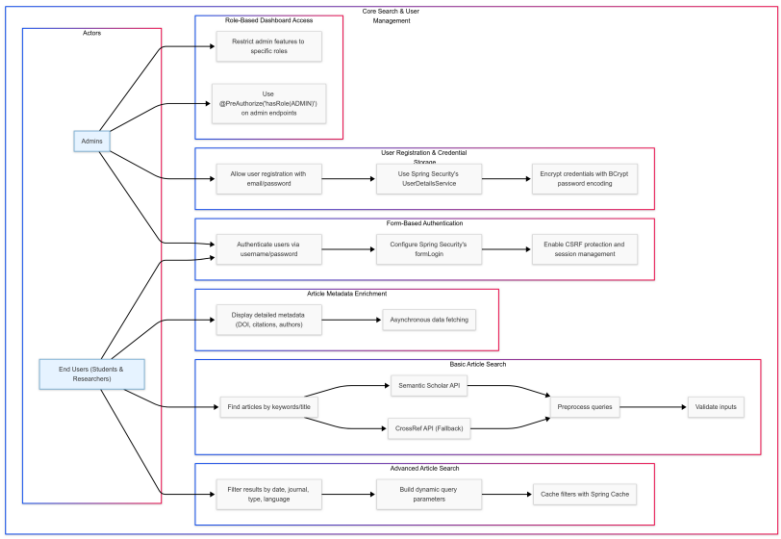
- **Descripción:** Representa a los usuarios del sistema.
- **Atributos:**
  - `id` (UUID): Identificador único (clave primaria).
  - `name` (String): Nombre del usuario.
  - `email` (String): Correo electrónico (único y obligatorio).

    - `role` (String): Rol asignado al usuario (Admin, Regular, etc.).
    - `password` (String): Contraseña en formato seguro (hashed).
- **Restricciones:**
    - `email`: Único y obligatorio.
    - `password`: Debe cumplir con la política de contraseñas seguras.
- **Relaciones:** Relación 1:N con Searches.

Entidad: Searches

- **Descripción:** Representa las búsquedas realizadas por los usuarios.
- **Atributos:**
    - `id` (UUID): Identificador único (clave primaria).
    - `timestamp` (Datetime): Fecha y hora de la búsqueda.
    - `query` (String): La consulta realizada por el usuario.
    - `userId` (UUID): Identificador del usuario que hizo la búsqueda (clave foránea).
- **Restricciones:**
    - `query`: Obligatoria.
    - `userId`: Debe existir en Users.
- **Relaciones:** Relación N:1 con Users.

Diagrama



Personalization & History Documentation

1. Reglas de negocio asociadas a cada entidad

- **Users**:
  - Cada usuario debe tener una lista personal de artículos favoritos, con la opción de añadir o eliminar elementos fácilmente.
  - Las sugerencias de búsqueda personalizadas deben basarse en el historial de búsqueda del usuario.
  - Los datos personales y de historial deben estar protegidos y solo accesibles por el usuario autenticado correspondiente.
- **Search History**:
  - Todas las consultas de búsqueda deben registrarse con marcas de tiempo para su análisis futuro.
  - El historial de búsqueda no debe exceder un tamaño predefinido para optimizar el almacenamiento y rendimiento.
- **Favorites Management**:
  - Los usuarios deben poder exportar su lista de artículos favoritos en formatos compatibles como CSV o JSON.
  - Las operaciones de guardar y eliminar favoritos deben estar vinculadas a roles y permisos del usuario.

2. Invariantes del sistema

- **Sugerencias Personalizadas**:
  - Las palabras clave sugeridas deben basarse en patrones de búsqueda histórica relevantes.
  - Las sugerencias deben ser actualizadas periódicamente para reflejar cambios en el comportamiento del usuario.
- **Gestión de Favoritos**:
  - Cada artículo favorito debe estar vinculado exclusivamente a un usuario.
  - Los datos exportados deben incluir solo elementos accesibles y visibles para el usuario autenticado.
- **Historial de Búsqueda**:
  - Las entradas del historial deben estar ordenadas cronológicamente.
  - Las consultas no válidas o duplicadas deben ser eliminadas para mantener la integridad de los datos.

3. Diccionario de Datos

Entidad: Users

- **Descripción:** Representa a los usuarios que interactúan con el sistema.
- **Atributos:**
  - `id` (UUID): Identificador único del usuario.
  - `name` (String): Nombre del usuario.
  - `email` (String): Correo electrónico único.
  - `preferences` (JSON): Configuraciones de personalización específicas del usuario.
- **Restricciones:**
  - `email`: Obligatorio y único.
  - `preferences`: Deben ser válidas y consistentes con las configuraciones definidas.

Entidad: Search History

- **Descripción:** Representa el historial de búsqueda del usuario.
- **Atributos:**
  - `historyId` (UUID): Identificador único del registro de historial.
  - `userId` (UUID): Identificador único del usuario asociado.
  - `query` (String): Consulta de búsqueda realizada.
  - `timestamp` (Datetime): Fecha y hora de la búsqueda.
- **Restricciones:**
  - `query`: Obligatorio.
  - `timestamp`: Obligatorio.
- **Relaciones:** Relación N:1 con Users.

Entidad: Favorites

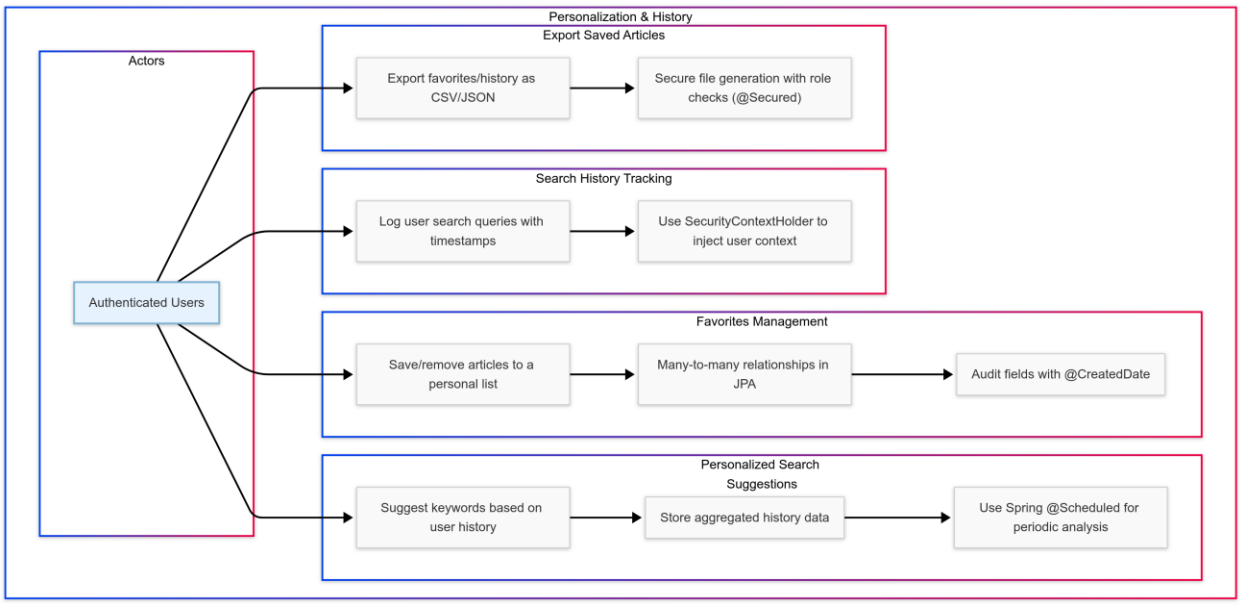
- **Descripción:** Representa los artículos favoritos guardados por los usuarios.
- **Atributos:**
  - `favoriteId` (UUID): Identificador único del artículo favorito.
  - `userId` (UUID): Identificador único del usuario asociado.
  - `articleData` (JSON): Información del artículo guardado.
- **Restricciones:**
  - `userId`: Obligatorio y debe existir en Users.
  - `articleData`: Obligatoria y válida.

- **Relaciones:** Relación N:1 con Users.

Entidad: Export Logs

- **Descripción:** Registra las exportaciones realizadas por los usuarios.
- **Atributos:**
  - `exportId` (UUID): Identificador único de la exportación.
  - `userId` (UUID): Identificador único del usuario que realizó la exportación.
  - `format` (String): Formato del archivo exportado (CSV/JSON).
  - `timestamp` (Datetime): Fecha y hora de la exportación.
- **Restricciones:**
  - `format`: Obligatorio y válido.
  - `timestamp`: Obligatorio.
- **Relaciones:** Relación N:1 con Users.

Diagrama



Compliance & Data Security Documentation

1. Reglas de negocio asociadas a cada entidad

- **Admins**:

    - Los administradores son responsables de configurar las políticas para cumplir con las regulaciones, incluyendo GDPR.
    - Pueden generar reportes de cumplimiento y revisar registros de auditoría para identificar acciones sospechosas.
    - Tienen acceso para gestionar claves API de terceros y configurarlas de manera segura.
- **System**:
    - El sistema debe monitorear automáticamente el uso de la API para asegurar que no se excedan los límites de terceros.
    - Debe eliminar datos de usuarios que soliciten su eliminación, cumpliendo con las regulaciones.
    - Todas las acciones relacionadas con el manejo de datos deben ser registradas para auditorías.

2. Invariantes del sistema

- **API Rate Limit Compliance**:
    - Todas las solicitudes a APIs externas deben ser rastreadas para evitar exceder los límites permitidos.
    - Los reintentos deben ser limitados para no comprometer la cuota de uso disponible.
- **GDPR-Compliant Data Deletion**:
    - Los datos personales de los usuarios deben eliminarse completamente (incluyendo datos relacionados) al ser solicitados.
    - Se debe garantizar que las relaciones en cascada se manejen correctamente al eliminar los datos.
- **Secure API Key Storage**:
    - Las claves API deben estar protegidas y no deben ser expuestas en el código fuente.
    - La configuración de claves debe ser accesible solo mediante permisos específicos.
- **Audit Logging**:
    - Todos los eventos relevantes, como inicio de sesión y búsqueda, deben estar registrados con marca de tiempo y detalles del usuario.
    - Los registros de auditoría deben ser inmutables y únicamente accesibles por roles autorizados.

3. Diccionario de Datos

Entidad: Admins

- **Descripción:** Representa a los administradores que supervisan el cumplimiento de seguridad y regulaciones.
- **Atributos:**
  - `id` (UUID): Identificador único del administrador.
  - `email` (String): Correo electrónico del administrador.
  - `permissions` (JSON): Lista de permisos asignados.
- **Restricciones:**
  - `email`: Obligatorio y único.
  - `permissions`: Obligatorios para definir responsabilidades específicas.

Entidad: API Usage

- **Descripción:** Registra el uso de APIs externas por el sistema.
- **Atributos:**
  - `usageId` (UUID): Identificador único del registro de uso.
  - `apiName` (String): Nombre de la API utilizada.
  - `requestCount` (Integer): Número de solicitudes realizadas.
  - `lastRequestTimestamp` (Datetime): Fecha y hora de la última solicitud.
- **Restricciones:**
  - `requestCount`: No debe exceder el límite definido.
  - `apiName`: Obligatoria.

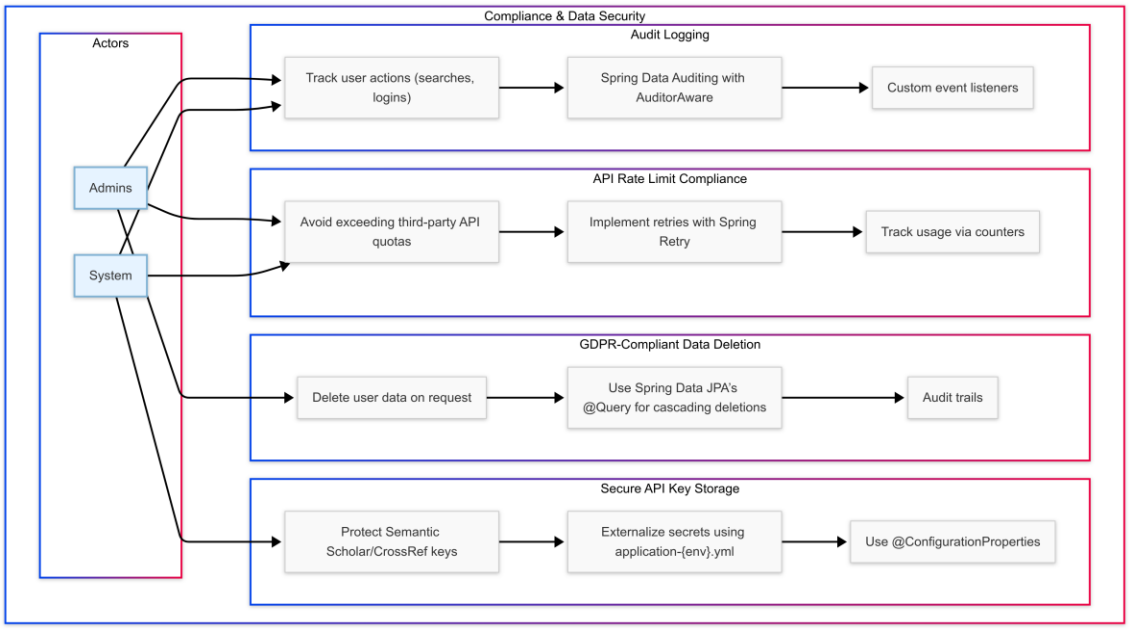
Entidad: GDPR Deletion Requests

- **Descripción:** Maneja solicitudes de eliminación de datos conforme al GDPR.
- **Atributos:**
  - `requestId` (UUID): Identificador único de la solicitud.
  - `userId` (UUID): Usuario que realizó la solicitud.
  - `status` (String): Estado actual de la solicitud (e.g., Pendiente, Completada).
  - `timestamp` (Datetime): Fecha y hora de la solicitud.
- **Restricciones:**
  - `userId`: Obligatoria y válida.
  - `status`: Obligatoria y debe ser coherente con el flujo definido.

Entidad: Audit Logs

- **Descripción:** Registra acciones realizadas dentro del sistema para auditorías.
- **Atributos:**
  - `logId` (UUID): Identificador único del registro.
  - `action` (String): Acción registrada (e.g., Inicio de sesión, Modificación).
  - `actorId` (UUID): Identificador del usuario o sistema que realizó la acción.
  - `timestamp` (Datetime): Fecha y hora del evento.
- **Restricciones:**
  - `action`: Obligatoria.
  - `timestamp`: Obligatoria

Diagrama



Recommendation Engine Documentation

1. Reglas de negocio asociadas a cada entidad

- **Articles**:
  - Los artículos deben ser analizados para calcular su similitud con otros utilizando modelos de lenguaje preentrenados (e.g., embeddings).
  - Cada artículo debe tener metadatos relevantes que faciliten la búsqueda y las recomendaciones (autor, tema, palabras clave).

    - Los artículos recomendados deben cumplir con un umbral de relevancia definido.
- **User Interactions**:
    - Las interacciones del usuario, como clics y búsquedas, deben registrarse para identificar patrones de comportamiento.
    - Las recomendaciones basadas en comportamiento deben adaptarse dinámicamente al historial y preferencias del usuario.
    - Los datos de interacción deben ser anonimizados y protegidos para cumplir con las normativas de privacidad.
- **Search Rankings**:
    - Los resultados de búsqueda deben ordenarse según los puntajes personalizados calculados para cada usuario.
    - Las preferencias del usuario deben influir en el ranking, utilizando datos como favoritos o historial de búsqueda.

2. Invariantes del sistema

- **Article Similarity Search**:
    - Los cálculos de similitud deben mantenerse consistentes al actualizar los modelos de embeddings.
    - Los artículos considerados como "similares" deben compartir al menos un conjunto de características clave (tema, palabras clave).
- **User Behavior Recommendations**:
    - Las recomendaciones deben actualizarse regularmente para reflejar los cambios en el comportamiento del usuario.
    - Solo las interacciones válidas deben ser consideradas en el análisis para evitar sesgos.
- **Personalized Search Ranking**:
    - Los rankings de búsqueda deben adaptarse exclusivamente a las preferencias del usuario autenticado.
    - Los puntajes personalizados deben ser recalculados si las preferencias del usuario cambian.

3. Diccionario de Datos

Entidad: Articles

- **Descripción:** Representa los artículos disponibles para búsquedas y recomendaciones.
- **Atributos:**
  - `articleId` (UUID): Identificador único del artículo.
  - `title` (String): Título del artículo.
  - `keywords` (Array<String>): Lista de palabras clave asociadas al artículo.
  - `embedding` (Vector): Representación en espacio vectorial para cálculos de similitud.
- **Restricciones:**
  - `title`: Obligatorio y único.
  - `embedding`: Debe ser generado con un modelo de lenguaje preentrenado.
- **Relaciones:** Relación N:M con User Interactions.

Entidad: User Interactions

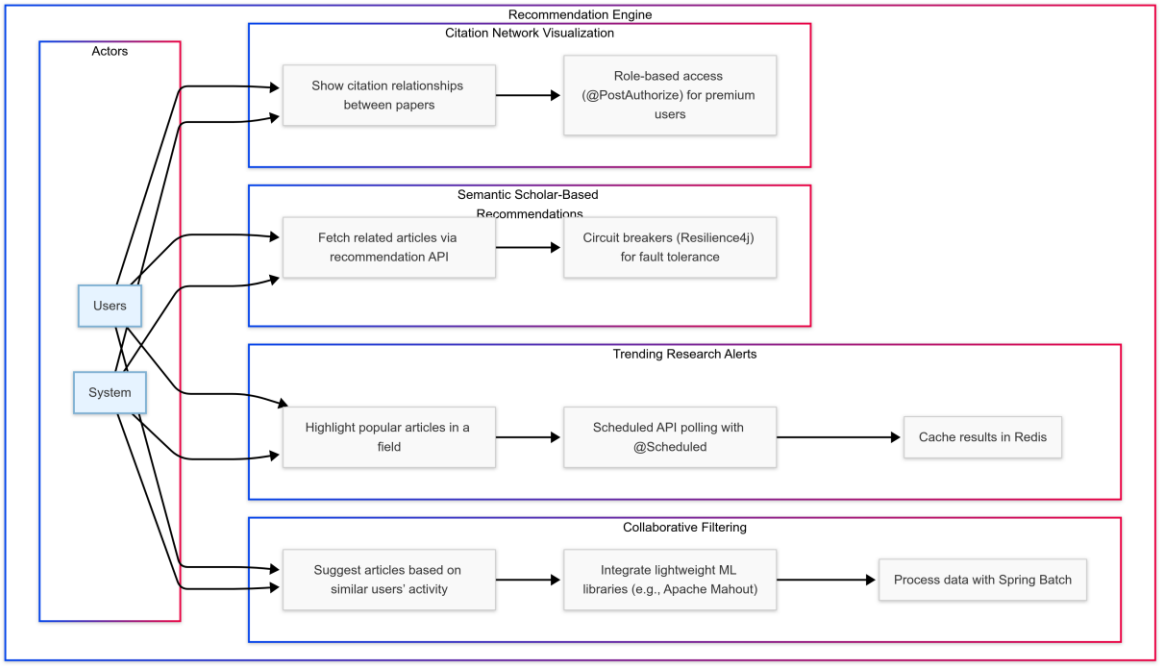
- **Descripción:** Representa las interacciones de los usuarios con los artículos y el sistema.
- **Atributos:**
  - `interactionId` (UUID): Identificador único de la interacción.
  - `userId` (UUID): Identificador único del usuario asociado.
  - `articleId` (UUID): Identificador del artículo con el que interactuó el usuario.
  - `actionType` (String): Tipo de interacción (e.g., clic, búsqueda).
  - `timestamp` (Datetime): Fecha y hora de la interacción.
- **Restricciones:**
  - `actionType`: Obligatorio.
  - `timestamp`: Obligatorio.
- **Relaciones:** Relación N:1 con Articles.

Entidad: Search Rankings

- **Descripción:** Representa el orden de los resultados de búsqueda para cada usuario.
- **Atributos:**
  - `rankingId` (UUID): Identificador único del ranking.
  - `userId` (UUID): Identificador único del usuario.
  - `articleId` (UUID): Identificador único del artículo.
  - `score` (Double): Puntaje calculado para el artículo en el ranking.

- **Restricciones:**
  - `score`: Debe ser válido y calculado usando preferencias del usuario.
- **Relaciones:** Relación N:1 con Articles.

Diagrama



Performance & Scalability

1. Reglas de negocio asociadas a cada entidad

- **API Responses**:
  - Las respuestas frecuentes deben ser almacenadas en caché para optimizar el rendimiento.
  - Los datos en caché deben ser válidos y eliminados cuando estén obsoletos.
  - Las políticas de caché deben cumplir con los límites de almacenamiento.
- **User Sessions**:
  - El sistema debe manejar concurrentemente al menos 1000 sesiones autenticadas activas.
  - Las sesiones deben ser seguras y almacenadas en una estructura distribuida.

    - Las sesiones inactivas se deben finalizar automáticamente tras un tiempo definido.
- **Authentication**:
    - El sistema debe gestionar picos de inicio de sesión sin afectar la disponibilidad.
    - Los filtros innecesarios deben ser deshabilitados para optimizar el proceso de autenticación.
- **Monitoring**:
    - El sistema debe registrar latencia, errores y tiempo de actividad en tiempo real.
    - Los puntos de monitoreo deben estar protegidos y sólo accesibles por roles autorizados.

2. Invariantes del sistema

- **API Responses**:
    - Todas las respuestas en caché deben cumplir con las condiciones de validez antes de ser utilizadas.
    - Las respuestas deben ser actualizadas periódicamente para mantener consistencia con los datos fuente.
- **User Sessions**:
    - Las sesiones deben estar activas para usuarios autenticados únicamente.
    - No se permiten sesiones duplicadas para el mismo usuario simultáneamente.
- **Authentication**:
    - Cada intento de inicio de sesión debe pasar por la cadena de filtros configurada.
    - El sistema no debe permitir autenticar usuarios sin cumplir con las políticas de seguridad.
- **Monitoring**:
    - Los puntos de monitoreo deben estar siempre disponibles y protegidos.
    - Los errores críticos deben ser reportados inmediatamente para evitar interrupciones.

3. Diccionario de Datos

Entidad: API Responses

- **Descripción:** Representa las respuestas frecuentes almacenadas en caché.
- **Atributos:**
  - `cacheKey` (String): Identificador único de la respuesta.
  - `response` (String): Contenido de la respuesta en caché.
  - `timestamp` (Datetime): Fecha y hora de la creación de la caché.
- **Restricciones:**
  - `cacheKey`: Único.
  - `response`: Obligatorio.
  - `timestamp`: Debe ser válido y representativo.
- **Relaciones:** No aplica directamente.

Entidad: User Sessions

- **Descripción:** Representa sesiones activas de usuarios autenticados.
- **Atributos:**
  - `sessionId` (UUID): Identificador único de la sesión.
  - `userId` (UUID): Identificador único del usuario asociado a la sesión.
  - `startTime` (Datetime): Hora de inicio de la sesión.
  - `lastActive` (Datetime): Última actividad registrada.
- **Restricciones:**
  - `sessionId`: Único.
  - `userId`: Debe existir en la tabla Users.
  - `startTime` y `lastActive`: Obligatorios.
- **Relaciones:** Relación N:1 con Users.

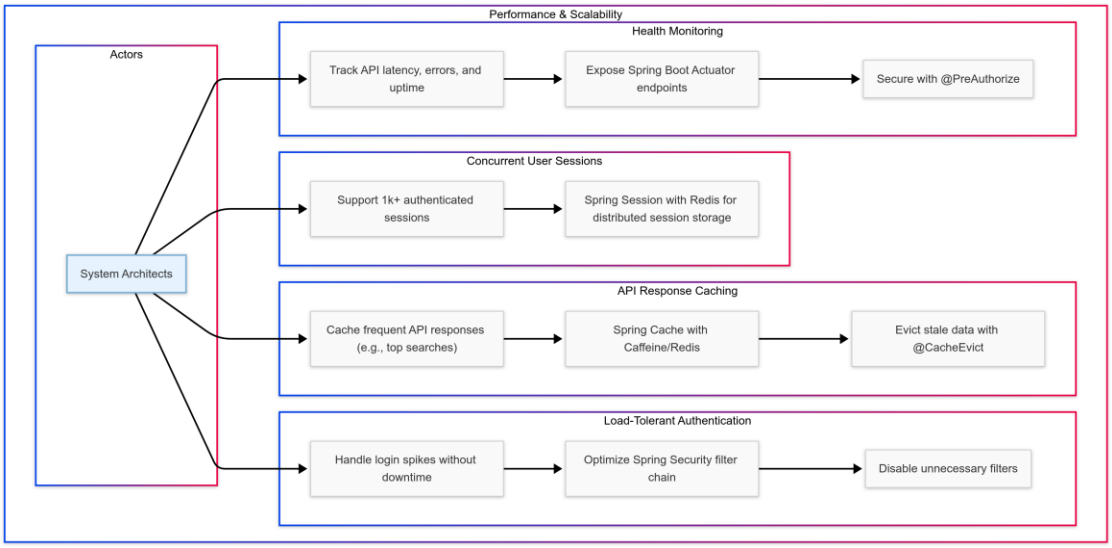
Entidad: Authentication Logs

- **Descripción:** Registra intentos de inicio de sesión y autenticación.
- **Atributos:**
  - `logId` (UUID): Identificador único del registro.
  - `userId` (UUID): Identificador del usuario que intentó iniciar sesión.
  - `attemptTime` (Datetime): Fecha y hora del intento.
  - `status` (String): Resultado del intento (Exitoso/Fallido).
- **Restricciones:**
  - `logId`: Único.
  - `userId`: Debe existir en la tabla Users.
- **Relaciones:** Relación N:1 con Users.

Entidad: Monitoring Metrics

- **Descripción:** Registra métricas relacionadas al rendimiento del sistema.
- **Atributos:**
  - `metricId` (UUID): Identificador único de la métrica.
  - `type` (String): Tipo de métrica (Latencia, Errores, Uptime).
  - `value` (String): Valor registrado.
  - `timestamp` (Datetime): Fecha y hora de registro.
- **Restricciones:**
  - `metricId`: Único.
  - `type` y `timestamp`: Obligatorios.
- **Relaciones:** No aplica directamente.

Diagrama



Accessibility & Integration

1. Reglas de negocio asociadas a cada entidad

- **Users**:
  - La interfaz debe ser completamente responsive para que los usuarios puedan acceder desde dispositivos móviles y desktop sin inconvenientes.
  - Los usuarios no deben tener acceso directo a datos sensibles a través de endpoints o integraciones de terceros.

    - Los errores en la UI o en las integraciones deben ser manejados de forma que no afecten la experiencia del usuario.
- **Developers**:
    - Los desarrolladores deben poder consumir los datos de la API mediante endpoints protegidos con claves API.
    - Todas las integraciones de terceros deben normalizar los datos antes de que se usen en el sistema.
    - Los desarrolladores tienen acceso limitado a las herramientas de gestión de metadatos.

2. Invariantes del sistema

- **Responsive Rendering**:
    - La UI debe adaptarse automáticamente a cualquier resolución de pantalla soportada.
    - Las plantillas Thymeleaf o frameworks de frontend deben proporcionar consistencia visual entre plataformas.
- **API Integration**:
    - Todas las consultas de API deben incluir claves API válidas.
    - Los datos obtenidos de terceros deben normalizarse según las especificaciones internas.
- **Error Handling**:
    - Los mensajes de error deben ser amigables para los usuarios y claros para los desarrolladores.
    - Los códigos de error HTTP personalizados deben seguir un estándar predefinido.

3. Diccionario de Datos

Entidad: Users

- **Descripción:** Representa a los usuarios del sistema.
- **Atributos:**
    - `id` (UUID): Identificador único del usuario.
    - `name` (String): Nombre del usuario.
    - `email` (String): Correo electrónico (único).
    - `preferences` (JSON): Configuraciones de accesibilidad específicas del usuario.

- **Restricciones:**
  - `email`: Obligatorio y único.
  - `preferences`: Deben ser válidas según las configuraciones definidas.

Entidad: API Integrations

- **Descripción:** Representa las integraciones con terceros para consumo de datos.
- **Atributos:**
  - `integrationId` (UUID): Identificador único de la integración.
  - `endpoint` (String): URL del endpoint del API externo.
  - `authKey` (String): Clave API para autenticar las consultas.
  - `lastSynced` (Datetime): Última fecha de sincronización exitosa.
- **Restricciones:**
  - `authKey`: Debe ser válida y obligatoria.
  - `endpoint`: Único y obligatorio.

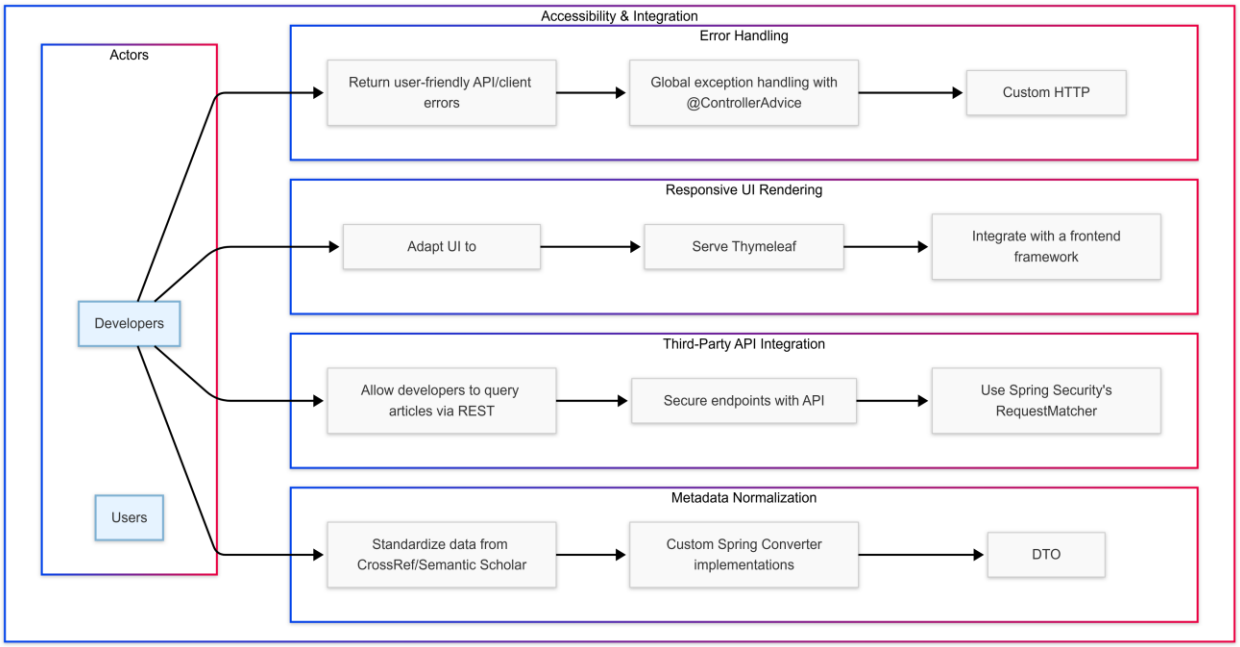
Entidad: Metadata Normalization

- **Descripción:** Representa los procesos de normalización de datos.
- **Atributos:**
  - `normalizationId` (UUID): Identificador único del proceso.
  - `source` (String): Fuente de los datos (CrossRef, Semantic Scholar).
  - `normalizedData` (JSON): Datos procesados y estandarizados.
- **Restricciones:**
  - `source`: Obligatoria.
  - `normalizedData`: Deben cumplir con las reglas de formato interno.

Entidad: Error Logs

- **Descripción:** Representa los errores detectados en el sistema.
- **Atributos:**
  - `logId` (UUID): Identificador único del registro.
  - `errorCode` (String): Código de error registrado.
  - `message` (String): Mensaje asociado al error.
  - `timestamp` (Datetime): Fecha y hora del registro.
- **Restricciones:**
  - `errorCode`: Obligatorio y único en el contexto.
  - `timestamp`: Obligatorio.

Diagrama



Advanced Security Documentation

1. Reglas de negocio asociadas a cada entidad

- **Admins**:
  - Los administradores son responsables de configurar políticas de seguridad, como la gestión de jerarquías de roles y la definición de contraseñas seguras.
  - Supervisan los eventos críticos de seguridad, incluidos los intentos de inicio de sesión fallidos y las auditorías de actividad.
  - Tienen acceso exclusivo a la configuración de tiempo de sesión y bloqueo de cuentas.
- **Users**:
  - Los usuarios deben autenticarse utilizando contraseñas que cumplan con los requisitos establecidos de longitud, complejidad y unicidad.
  - Las sesiones de usuario deben expirar automáticamente después de un período de inactividad configurado.

    - Las cuentas de usuario deben ser bloqueadas temporalmente si se detectan múltiples intentos de inicio de sesión fallidos consecutivos.

2. Invariantes del sistema

- **Prevención de Fuerza Bruta (Brute-Force Login Prevention)**:
    - Las cuentas deben ser bloqueadas temporalmente después de un número predefinido de intentos fallidos.
    - Los contadores de intentos se deben reiniciar automáticamente tras un inicio de sesión exitoso.
- **Gestión de Tiempos de Sesión (Session Timeout Management)**:
    - Las sesiones inactivas deben finalizar automáticamente después de un período configurado sin actividad.
    - Las sesiones activas no deben ser interrumpidas mientras haya interacción con el sistema.
- **Cumplimiento de Políticas de Contraseñas (Password Policy Enforcement)**:
    - Las contraseñas deben incluir una combinación de letras mayúsculas y minúsculas, números y caracteres especiales.
    - Las contraseñas deben ser cifradas utilizando un algoritmo seguro como bcrypt.
- **Jerarquías de Roles (Role Hierarchies)**:
    - Los roles deben estar organizados jerárquicamente para reflejar permisos acumulativos.
    - Las configuraciones de roles deben ser auditables y solo editables por administradores autorizados.

3. Diccionario de Datos

Entidad: Admins

- **Descripción:** Representa a los administradores responsables de la configuración y supervisión de la seguridad del sistema.

- **Atributos:**
  - `id` (UUID): Identificador único del administrador.
  - `name` (String): Nombre del administrador.
  - `email` (String): Correo electrónico único.
  - `role` (String): Rol asignado al administrador.
- **Restricciones:**
  - `email`: Obligatorio y único.
  - `role`: Obligatorio y reflejando permisos avanzados.
- **Relaciones:** Relación N:1 con `Role Hierarchies`.

Entidad: Users

- **Descripción:** Representa a los usuarios que interactúan con el sistema.
- **Atributos:**
  - `id` (UUID): Identificador único del usuario.
  - `email` (String): Correo electrónico único y obligatorio.
  - `password` (String): Contraseña cifrada en formato seguro.
  - `lockoutCounter` (Integer): Número de intentos fallidos consecutivos.
  - `lastLoginAttempt` (Datetime): Fecha del último intento de inicio de sesión.
- **Restricciones:**
  - `password`: Obligatoria y debe cumplir las políticas de seguridad.
  - `lockoutCounter`: No debe superar el límite permitido.
- **Relaciones:** Relación N:1 con `Role Hierarchies`.

Entidad: Roles

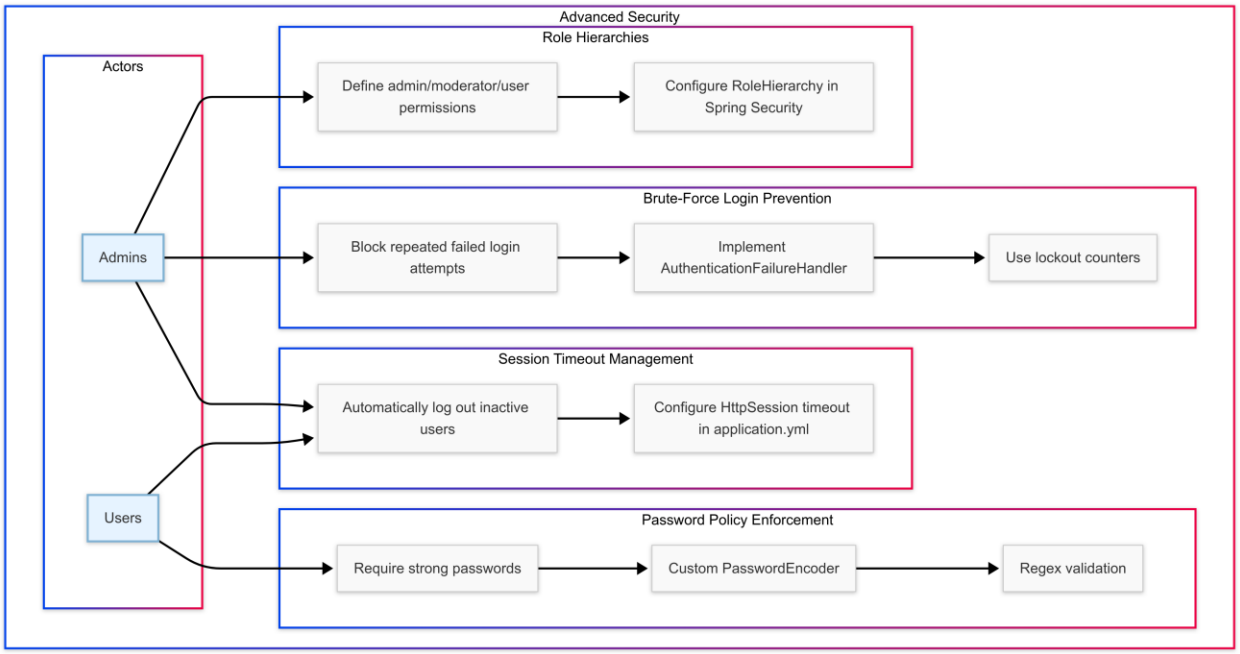
- **Descripción:** Representa la jerarquía de permisos de acceso en el sistema.
- **Atributos:**
  - `id` (UUID): Identificador único del rol.
  - `name` (String): Nombre del rol (e.g., Admin, Moderator, User).

    - `permissions` (JSON): Lista de permisos asignados al rol.
- **Restricciones:**
    - `name`: Obligatorio y único.
    - `permissions`: Obligatorios y válidos según los niveles definidos.
- **Relaciones:** Relación 1:N con `Admins` y `Users`.

Entidad: Audit Logs

- **Descripción:** Registra eventos críticos de seguridad como intentos fallidos de inicio de sesión.
- **Atributos:**
    - `logId` (UUID): Identificador único del registro.
    - `userId` (UUID): Identificador del usuario afectado.
    - `eventType` (String): Tipo de evento (e.g., Failed Login, Lockout).
    - `timestamp` (Datetime): Fecha y hora del evento.
- **Restricciones:**
    - `eventType`: Obligatorio.
    - `timestamp`: Obligatorio.
- **Relaciones:** Relación N:1 con Users

Diagrama



Endpoints

La siguiente tabla describe los endpoints de autenticación del sistema, permitiendo a los usuarios registrarse, iniciar sesión y cerrar sesión de forma segura. Se detallan los métodos HTTP, funcionalidades y validaciones aplicadas en cada endpoint para garantizar la integridad y seguridad del acceso al sistema.

Endpoint	Método	Funcionalidad	Validaciones	Rol por defecto
/registro	POST	Registro de nuevos usuarios	• Email único • Email con formato valido • Contraseña de mínimo 6 caracteres	ROLE_USER
/login	POST	Autenticación de usuarios	• Email con formato valido	NA
/logout	POST	Cierre de sesiones	NA	NA

La siguiente tabla presenta los endpoints protegidos del sistema, diseñados para restringir el acceso a funcionalidades específicas según el rol del usuario. Se detallan los roles autorizados para cada endpoint, asegurando que solo los usuarios con los permisos adecuados puedan acceder a las diferentes secciones y funcionalidades del sistema.

Ruta	Nombre	Descripción

/user/**	Dashboard de usuario	Acceso solo para usuarios autenticados incluyendo usuarios y administradores
/admin/**	Panel de administración	Acceso exclusivo para usuarios con rol ROLE_ADMIN
/api/admin/**	API de administración	• Acceso exclusivo para usuarios con rol ROLE_ADMIN • Endpoints REST para gestión de usuarios • Requiere autenticación y rol de administrador

Roles

El sistema implementa un modelo de control de acceso basado en roles para gestionar los permisos y funcionalidades disponibles para los usuarios. Se definen dos roles principales: ROLE_USER, que proporciona acceso a funcionalidades básicas y personalizadas, y ROLE_ADMIN, que otorga privilegios administrativos completos para la gestión del sistema.

Rol	Funcionalidades
ROLE_USER	• Acceso a dashboard personal. • Visualización de perfil. • Acceso limitado.
ROLE_ADMIN	• Acceso al panel de administración. • Gestión completa de usuarios. • Acceso a las funcionalidades de ROLE_USER.

Seguridad y Gestión de Sesiones

El sistema implementa una sólida estrategia de seguridad para proteger la información y el acceso de los usuarios. Se utilizan tecnologías y prácticas avanzadas para garantizar la confidencialidad, integridad y disponibilidad de los datos, incluyendo autenticación robusta, encriptación de contraseñas y control de acceso basado en roles.

Estrategias de Seguridad:

- **Autenticación:**
    - Spring Security para un marco de seguridad integral.

    - Validación de credenciales personalizada mediante CustomUserDetailsService, con consulta de usuarios en base de datos y comparación de contraseñas encriptadas.
- **Encriptación:**
    - BCrypt para el almacenamiento seguro de contraseñas.
- **Control de Acceso:**
    - Rutas públicas: /registro, /login, recursos estáticos.
    - Rutas protegidas: Requieren autenticación y control de acceso basado en roles.
- **Gestión de Sesiones:**
    - Inicio y cierre de sesión con redirección dinámica y URL de logout.
    - Opción de "Recordarme" para mayor comodidad.

Puntos clave adicionales:

- La seguridad se centra en la autenticación robusta, la encriptación de contraseñas y el control de acceso basado en roles.
- Se utilizan tecnologías estándar de la industria, como Spring Security y BCrypt, para garantizar la seguridad.
- Se distingue claramente entre rutas públicas y protegidas para controlar el acceso a los recursos del sistema.

Conexión a Base de Datos

La aplicación se conectará a una base de datos relacional MySQL para la persistencia y administración de los datos de usuarios. Se utilizará Spring Data JPA para abstraer las operaciones de acceso y manipulación de la información, permitiendo la ejecución de operaciones CRUD de manera eficiente y segura.

Uso de Spring Data JPA para la gestión de usuarios

Objetivo:
Utilizar Spring Data JPA para implementar las operaciones CRUD (crear, leer, actualizar y eliminar) sobre la entidad *Usuario*, que almacena información relevante como correo, contraseña, nombre y rol.

Detalles de la Implementación:

La entidad **Usuario** se define mediante anotaciones JPA, garantizando la integridad de los datos mediante validaciones como @Email para el correo y @NotEmpty para el nombre.

Se utiliza un repositorio basado en Spring Data para abstraer las operaciones de acceso a la base de datos, facilitando la interacción con la misma sin escribir consultas SQL manualmente.

La estrategia de generación del identificador (por ejemplo, GenerationType.IDENTITY) y la configuración de la tabla (@Table(name = "usuarios")) aseguran la correcta persistencia en la base de datos.

Configuración detallada en el archivo application.properties.

Propósito de la Configuración:
El archivo application.properties centraliza la configuración de la conexión a la base de datos y de otros componentes de la aplicación, facilitando ajustes sin necesidad de recompilar el código.

Parámetros Esenciales:

- **spring.application.name:** Define el nombre de la aplicación.
- **spring.datasource:** Parámetros de conexión a la base de datos (URL, usuario y contraseña).
- **spring.jpa:** Configuraciones específicas para Hibernate y el manejo del esquema de la base de datos (como ddl-auto, que en este caso se utiliza en modo *update* para sincronizar los cambios en la entidad).
- **Configuración Adicional:**
  Se incluyen parámetros para Thymeleaf (en caso de ser necesario) que gestionan la ubicación y el formato de las plantillas HTML.

Funcionalidad (F)

F-01 Registro y Autenticación de Usuario
• *Descripción:* Permite a nuevos usuarios registrarse, autenticarse y gestionar su perfil, incluyendo la opción de inicio de sesión mediante proveedores externos (Google, Facebook, etc.).
• *Criterios de Aceptación:* Email único, contraseña segura, validación de credenciales locales y externas, y asignación de un rol por defecto (ROLE_USER).

F-02 Inicio de Sesión
• *Descripción:* Los usuarios se autentican utilizando su email y contraseña.
• *Criterios de Aceptación:* Validación de credenciales contra la base de datos y redirección según el rol.

F-03 Panel de Administración
• *Descripción:* Los administradores pueden gestionar usuarios mediante un CRUD exclusivo.
• *Criterios de Aceptación:* Acceso restringido a ROLE_ADMIN.

F-04 Búsqueda de Artículos
• *Descripción:* Permite a los usuarios buscar artículos científicos por palabras clave, autor, año o tema.
• *Criterios de Aceptación:* Resultados relevantes basados en los criterios ingresados.

F-05 Filtros Avanzados
• *Descripción:* Implementación de filtros por idioma, revista o tipo de artículo.
• *Criterios de Aceptación:* Posibilidad de aplicar múltiples filtros y actualización en tiempo real.

F-06 Visualización de Artículos
• *Descripción:* Muestra detalles clave (título, autores, resumen, año, enlace) de cada artículo.
• *Criterios de Aceptación:* Presentación clara y organizada.

F-07 Visor Embebido
• *Descripción:* Permite la visualización de documentos PDF en acceso abierto dentro del sistema.
• *Criterios de Aceptación:* Activación solo si el artículo es de acceso abierto.

F-09 Recomendación Personalizada

• *Descripción:* Ofrece recomendaciones basadas en el historial de búsqueda, artículos marcados como favoritos y tendencias generales del sistema.
• *Criterios de Aceptación:* Actualización dinámica de las recomendaciones.

F-10 Historial y Favoritos

• *Descripción:* Registra el historial de búsqueda y permite marcar artículos como favoritos.
• *Criterios de Aceptación:* Acceso y gestión intuitiva del historial y favoritos.

F-11 Integración con APIs Externas

• *Descripción:* Conecta el sistema con APIs como CrossRef y Semantic Scholar para obtener metadatos en formato JSON.
• *Criterios de Aceptación:* Procesamiento correcto de las respuestas JSON y almacenamiento adecuado de la información.

F-12 Sistema de Endpoints REST

• *Descripción:* Exponer la información del sistema a través de endpoints REST que permitan:
  – Consultar artículos filtrados según criterios de búsqueda.
  – Acceder al historial y favoritos del usuario.
  – Integrar funcionalidades para terceros.
• *Criterios de Aceptación:* Respuestas en formato JSON, documentación técnica detallada y pruebas de integración.

Usabilidad (U)

• Diseño minimalista y responsivo, compatible con dispositivos móviles y de escritorio.
• Accesibilidad para usuarios con discapacidades, incluyendo soporte para lectores de pantalla y navegación por teclado.
• *Personalización de la Interfaz:* Opciones para ajustar el tamaño del texto y el contraste, facilitando la experiencia de usuarios con dificultades visuales.
• Experiencia de usuario optimizada mediante tutoriales interactivos y mensajes de error claros.
• Creación de prototipos funcionales que muestren las principales funcionalidades del sistema, siguiendo un enfoque de diseño centrado en el usuario.

Confiabilidad (R)

• Gestión robusta de errores y validación estricta de datos para evitar fallos.
• Seguridad mediante autenticación segura, encriptación de datos sensibles y prevención de vulnerabilidades (inyección SQL, XSS, fuerza bruta).
• *Disponibilidad:* Garantizar un tiempo de actividad del 99.9% mediante infraestructura en la nube.
• *Recuperación Automática:* Implementar mecanismos para la recuperación ante fallos, especialmente en las solicitudes a APIs externas.

Rendimiento (P)

• Tiempos de respuesta inferiores a 3 segundos para la autenticación y operaciones clave.
• *Búsquedas en APIs Externas:* Procesadas en menos de 5 segundos.
• Escalabilidad para soportar hasta 10,000 usuarios concurrentes, con capacidad de expansión según la demanda.
• Uso eficiente de recursos mediante técnicas de caché y procesamiento distribuido.

Soporte y Mantenibilidad (S)

• Código modular y documentado para facilitar actualizaciones y mantenimiento.
• *Documentación Técnica:* Completa para desarrolladores, con ejemplos de uso de los endpoints REST.
• Integración con herramientas de gestión (ej. ClickUp) para el seguimiento de tareas y sprints.
• *Compatibilidad:* Integración con APIs estándar de datos académicos (por ejemplo, CrossRef, CORE, Semantic Scholar) y capacidad para exportar datos en formatos JSON y CSV para análisis externo.
• Actualizaciones regulares basadas en el feedback de los usuarios.

Aspectos Adicionales (+)

Esta sección complementa los requerimientos previos, abordando aspectos opcionales y consideraciones estratégicas que pueden aportar valor agregado al sistema:

• **Integraciones Opcionales:**
  o *Autenticación vía OAuth 2.0:* Posible incorporación de inicio de sesión con Google,

Facebook u otros proveedores, ofreciendo mayor flexibilidad y simplificando el proceso de registro.

o *Integración con Servicios de Análisis:* Posibilidad de integrar módulos para analizar tendencias y comportamiento de usuarios que ayuden a mejorar las recomendaciones.

• **Consideraciones Legales y Normativas:**

o Cumplimiento de normativas internacionales (por ejemplo, GDPR) en la protección de datos personales.

o Implementación de políticas de privacidad y términos de uso claros, que aseguren la transparencia y el manejo adecuado de la información.

• **Consideraciones del Diseño:**

o Arquitectura modular que permita la incorporación de nuevas funcionalidades sin afectar la estabilidad del sistema.

o Diseño escalable y adaptable, considerando futuros crecimientos en la cantidad de usuarios y la integración con nuevos servicios externos.

o Uso de patrones de diseño que faciliten el mantenimiento y evolución del software.

• **Estrategias de Mejora Continua:**

o Establecimiento de ciclos de feedback y revisiones periódicas para la implementación de mejoras basadas en el comportamiento y la experiencia de los usuarios.

o Monitorización constante del rendimiento y la seguridad para anticipar y mitigar posibles vulnerabilidades.

• **Extensiones Adicionales:**

o *Análisis Avanzado:* Implementar algoritmos de aprendizaje automático para mejorar las recomendaciones y analizar tendencias en el uso del sistema.

o *Exportación de Datos:* Permitir a los usuarios exportar resultados de búsqueda y listas de favoritos para integrarlos en otros sistemas.

o *Integración con Ontologías:* Utilizar bases de conocimiento como DBpedia o Wikidata para enriquecer la información mostrada.