Subject Index

Guía de usuario

Abril 2025

1. Introducción

`SubjectIndex` es una biblioteca diseñada para el almacenamiento, indexado y consulta de datos asociados a un *Subject* (sujeto), identificado por un par `(id, type)`. A diferencia de estructuras orientadas a la presentación o modelado estático, `SubjectIndex` se centra en *cómo* se categoriza y segmenta la información mediante pares clave-valor (tokens), con el objetivo de habilitar búsquedas rápidas y consultas flexibles sobre un conjunto heterogéneo de sujetos.

Para ilustrar su utilidad, imaginemos una aplicación de organización de proyectos académicos. Cada proyecto puede estar representado por un sujeto, como por ejemplo `12345:project`, y puede contener atributos como `manager`, `departamento`, `titulo`, `descripcion`, etc. Estos atributos se almacenan como tokens, permitiendo que el proyecto sea posteriormente localizado a través de múltiples criterios de filtrado.

Diferentes fuentes de información (como formularios, servicios web o análisis automáticos) pueden ir aportando metadatos a cada sujeto, sin necesidad de definir una estructura de datos global. Por ejemplo, un mismo proyecto puede tener múltiples usuarios asignados, descripciones actualizadas con el tiempo o accesos restringidos mediante expresiones regulares. `SubjectIndex` permite agregar o eliminar estos tokens dinámicamente, conservando un historial compacto y fácilmente consultable.

El verdadero potencial de `SubjectIndex` reside en su capacidad de realizar consultas expresivas sobre los sujetos, permitiendo filtrar por tokens exactos, buscar por coincidencia parcial o evaluar patrones mediante expresiones regulares. De esta forma, se pueden extraer conjuntos de sujetos que

cumplan determinadas condiciones, sin necesidad de recorrer manualmente todo el espacio de datos.

Gracias a su diseño enfocado en la simplicidad operativa y la potencia consultiva, `SubjectIndex` se adapta tanto a contextos simples (como clasificación de usuarios o agrupación de documentos) como a entornos más complejos donde múltiples actores enriquecen progresivamente la información asociada a un conjunto creciente de sujetos.

2. Prerrequisitos

Antes de comenzar a utilizar `SubjectIndex` en tu proyecto, es necesario cumplir con los siguientes requisitos básicos:

- **Java 19 o superior**: Asegúrate de contar con una versión compatible del JDK instalada en tu entorno de desarrollo. Puedes verificar tu versión ejecutando:

  ```
  java -version
  ```

- **Apache Maven**: Es necesario disponer de Maven como gestor de dependencias y herramienta de compilación para integrar fácilmente la biblioteca en tu proyecto. Puedes comprobar si está instalado con el siguiente comando:

  ```
  mvn -version
  ```

- **Conocimientos básicos sobre series temporales y atributos categóricos**: Dado que `SubjectStore` maneja datos temporales, es recomendable entender conceptos elementales sobre series de tiempo y datos categorizados.

3. Instalación

Puedes instalar `SubjectIndex` en tu proyecto Java utilizando un gestor de dependencias como Maven o Gradle.

Para incorporar `SubjectIndex` mediante Maven, añade la siguiente dependencia al archivo `pom.xml` de tu proyecto:

```
<dependency>
    <groupId>systems.intino.datamarts</groupId>
    <artifactId>subject-index</artifactId>
    <version>1.0.0</version>
</dependency>
```

4. Indexado

El proceso de indexado en `SubjectIndex` consiste en asociar uno o varios tokens (pares clave-valor) a un sujeto identificado por un `id` y un `type`. Esta operación se realiza mediante una API que permite agregar múltiples tokens y confirmarlos mediante un commit explícito.

Para indexar un sujeto, se utiliza el método `on(id, type)`, seguido de una o más llamadas a `set(key, value)`, y finalizando con `commit()` para aplicar los cambios al índice.

```
SubjectIndex index = new SubjectIndex(file);

index.on("anchor-map", "project")
     .set("descripcion", "libreria")
     .set("programador", "jose")
     .set("programador", "mario")
     .set("departamento", "TIC")
     .commit();
```

Cada operación de indexado afecta únicamente al sujeto especificado, y no sobrescribe el resto de sus tokens a menos que se indique explícitamente. Además, múltiples valores pueden coexistir para la misma clave, permitiendo un modelo multivaluado por campo.

```
index.on("quassar", "project")
     .set("descripcion", "entorno de modelado")
     .set("programador", "octavio")
     .commit();
```

También es posible eliminar tokens específicos de un sujeto utilizando el método `unset(key, value)`, seguido de `commit()` para confirmar la operación:

```
index.on("123456", "project")
     .unset("departamento", "TIC")
     .commit();
```

Para eliminar completamente un sujeto del índice, junto con todos sus tokens, se utiliza el método `drop(id, type)`. Esta operación es inmediata y no requiere un `commit` posterior. Una vez eliminado, el sujeto ya no forma parte del índice y no será devuelto por ninguna consulta, incluso si previamente tenía tokens asignados.

```
index.drop("anchor-map", "project");
```

5. Consultas

5.1. Tokens de un sujeto

Es posible recuperar todos los tokens asociados a un sujeto específico. Esta operación permite inspeccionar el contenido completo de un sujeto a partir de su identificador y tipo. El resultado es un conjunto de pares clave-valor que representan los atributos actuales del sujeto. Esta operación es útil para depuración, visualización o exportación de metadatos asociados a una entidad.

```
TokenSet tokens = index.tokens().of("intino", "project");
```

5.2. Sujetos cuyos tokens cumplan ciertas condiciones

El índice permite construir consultas complejas para recuperar únicamente aquellos sujetos que cumplan una o más condiciones sobre sus tokens. Estas condiciones pueden especificarse de distintas formas:

- Presencia exacta de un token. Para recuperar todos los sujetos que tengan asignado un token con una clave y un valor específicos. Es decir, se excluyen del resultado aquellos sujetos que contengan exactamente ese par clave-valor.

  ```
  SubjectSet set = index.subjects()
                      .with("email", "jose@example.com")
                      .toSet();
  ```

- Ausencia de un token. Para recuperar todos los sujetos que no tengan asignado un token con una clave y un valor específicos. Es decir, se excluyen del resultado aquellos sujetos que contengan exactamente ese par clave-valor.

```
SubjectSet set = index.subjects()
                    .with("email", "jose@example.com")
                    .toSet();
```

- Coincidencia parcial del valor de un campo. Para encontrar sujetos cuyo valor asociado a una clave contenga una subcadena específica. La búsqueda no requiere coincidencia exacta; basta con que el valor registrado incluya, en cualquier parte, la cadena proporcionada.

```
SubjectSet set = index.subjects()
                    .where("name")
                    .contains("sim")
                    .toSet();
```

- Coincidencia por expresión regular. Para encontrar sujetos cuyo valor asociado a una clave sea una expresión regular que coincida con un texto de entrada. A diferencia de una búsqueda convencional, aquí el contenido del token actúa como patrón, y el valor proporcionado es evaluado contra ese patrón.

```
SubjectSet set = index.subjects()
                    .where("access")
                    .matches("juan@ulpgc.es")
                    .toSet();
```