Universidad Zaragoza

Trabajo Fin de Grado

Grado en Ingeniería Informática

Monitorización de la calidad de los metadatos en portales de datos abiertos

Autor

Sergio Martínez Martín

Director

Javier Nogueras Iso

Escuela de Ingeniería y Arquitectura

Curso 2021-2022

Resumen

Para fomentar la transparencia y la participación ciudadana, numerosos gobiernos, tanto a nivel nacional como local, han promovido la creación de iniciativas de datos abiertos para diseminar los datos con los que trabajan. Estas iniciativas se implementan sobre la puesta en marcha de portales de datos abiertos, cuyo elemento principal son catálogos en línea interoperables que facilitan la búsqueda y publicación de los conjuntos de datos abiertos a partir de los metadatos que los describen.

Para fomentar la interoperabilidad y que se puedan encontrar fácilmente los datos abiertos, se debe utilizar un vocabulario común para definir estos metadatos y su contenido debe ser de alta calidad. Respecto al vocabulario, existe un consenso generalizado en el uso de DCAT, un esquema de metadatos propuesto basado en RDF y propuesto por W3C para la descripción de catálogos de datos. En cuanto a la evaluación de la calidad de los metadatos existen varias propuestas emergentes, destacando entre ellas la metodología MQA (Metadata Quality Assesment) elaborada por la Comisión Europea para el Portal de Datos Europeo y la metodología que adapta la norma ISO 19157 de calidad de información geográfica al contexto de los metadatos de portales de datos abiertos.

El objetivo de este Trabajo de Fin de Grado ha sido construir un portal web para monitorizar la calidad de los metadatos de portales de datos abiertos en base a los métodos existentes de evaluación de la calidad de los metadatos. A través de este portal se pretende proporcionar una interfaz de usuario que facilite y mejore la experiencia de usuario a la hora de aplicar estos métodos de evaluación de la calidad, añadiendo además características adicionales que puedan resultar de utilidad como puede ser la monitorización automática de un portal de datos abiertos, la presentación gráfica de los resultados de la evaluación, y la exportación de estos resultados con el vocabulario DQV (Data Quality Vocabulary), un vocabulario basado en RDF que se usa en la actualidad para informar sobre la calidad.

Para el desarrollo de la aplicación web que da soporte al portal de monitorización se ha utilizado el stack tecnológico MEAN: Angular para el Frontend, Node.js y Express para el Backend, y MongoDB para el almacenamiento de los resultados de la evaluación. En cuanto al cálculo automático de las medidas de evaluación de la calidad, este se ha desarrollado en Python y se han integrado el lenguaje SPARQL para hacer consultas sobre el contenido de los metadatos del portal de datos abiertos evaluado.

Agradecimientos

A mi familia y a mi pareja por haberme apoyado especialmente durante toda mi etapa universitaria.

A mis amigos por haber permitido que me distraiga cuando más lo necesitaba.

En especial a Daniel, con quien he coincidido la mayor parte de mis estudios universitarios y hemos podido contar el uno con el otro siempre que ha hecho falta

A Javier Nogueras, director de este proyecto por haber tenido la paciencia de mostrarme una parte desconocida de mi campo de trabajo y acompañarme a lo largo de todo el proyecto.

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación

Con el gran aumento del interés por parte de gobiernos y entidades públicas de ser más transparentes, han surgido numerosas iniciativas de datos abiertos que persiguen que determinados tipos de datos estén disponibles de forma libre para todo el mundo, sin restricciones de derechos de autor, de patentes o de otros mecanismos de control. El elemento más visible de estas iniciativas es la creación de portales web de datos abiertos, los cuales se construyen sobre la base de catálogos de metadatos interoperables describiendo los conjuntos de datos.

Para que los catálogos sean interoperables, es necesario que se cumplan dos elementos fundamentales: por un lado, los metadatos deben seguir un esquema común; por otro lado, las interfaces de acceso a los catálogos deben ser estándar.

Cuando hablamos de esquemas de metadatos, nos referimos al establecimiento del conjunto de propiedades (título, temática, descripción , autoría, fecha de publicación, etc.) que vamos a utilizar para describir los datos abiertos. En el contexto de los catálogos de datos abiertos existe un consenso generalizado en la utilización del vocabulario DCAT [8]. DCAT es el acrónimo de W3C's Data Catalog vocabulary, una recomendación de W3C para describir datos abiertos, desarrollado a través de un perfil de metadatos de Dublin Core basado en vocabularios RDF [7].[1] La Figura 1.1 muestra un diagrama

[1]RDF (Resource Description Framework) es una familia de especificaciones propuestas por W3C, que fue diseñada orignalmente como un modelo de datos para metadatos, pero que actualmente se utiliza extensivamente para describir información semántica que sea procesable. El modelo de representación de RDF se basa en el uso de tripletas de información del tipo *subject-predicate-object*: un recurso x (*subject*) tiene una propiedad y (*predicate*) con un valor z (*object*).

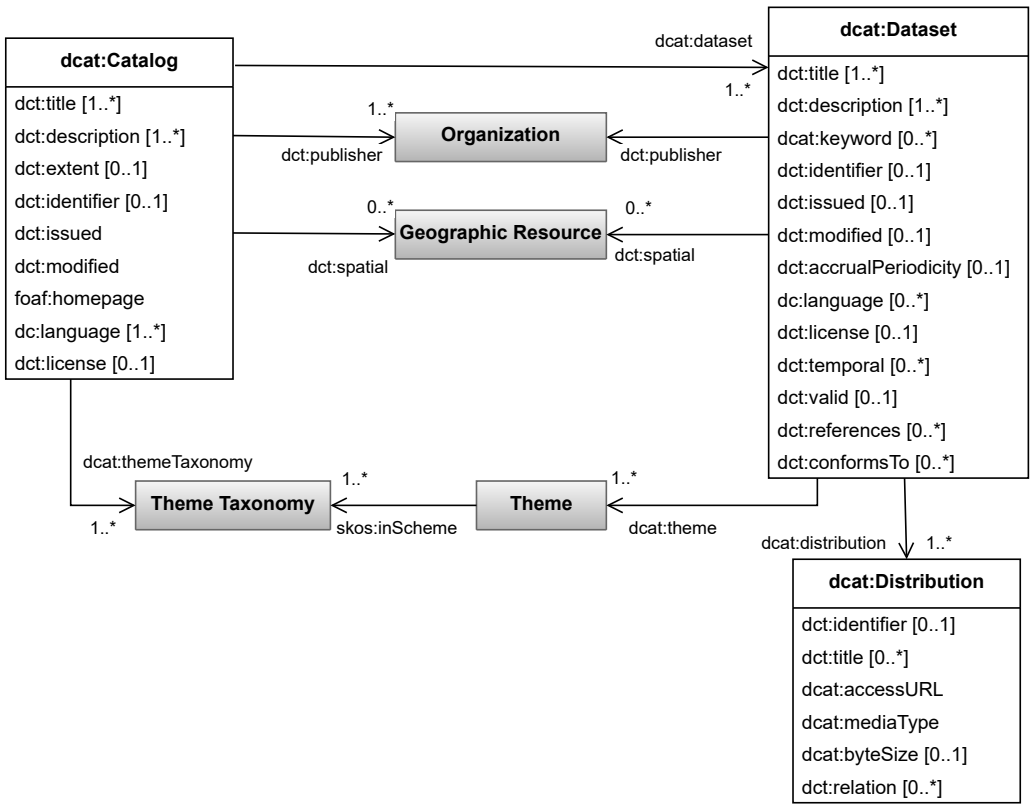

Figura 1.1: Principales clases, propiedades y relaciones del vocabulario DCAT en su adaptación al contexto español [19]

UML con las entidades y propiedades principales incluidas en DCAT, las cuales fueron adoptadas en España para la descripción de datos abiertos a través de la Norma Técnica de Interoperabilidad de Reutilización de Recursos de la Información [19]. Los vocabularios basados en DCAT se centran en proporcionar información acerca de tres entidades principales: catálogos (Catalog), conjuntos de datos (Dataset) y distribuciones (Distribution). Las propiedades de un catálogo informan acerca de la organización a cargo de la publicación de conjuntos de datos abiertos. Por otra parte, las propiedades de un conjunto de datos proporcionan la información principal para su búsqueda y caracterización. Finalmente, las propiedades de una distribución se centran fundamentalmente en informar de los mecanismos para solicitar o descargar los conjuntos de datos.

Respecto a las interfaces proporcionadas por los catálogos, estos suelen implementarse mediante soluciones software como CKAN [6]. CKAN es la plataforma Open Source más utilizada para dar soporte a portales de da-

tos abiertos e incluye los complementos necesarios para poder intercambiar metadatos basados en RDF, como es el caso de DCAT. En particular, este software ofrece una API REST que proporciona operaciones concretas para filtrar metadatos de conjuntos de datos que cumplen ciertos criterios y descargarlos; o para recolectar metadatos de otros catálogos de datos abiertos (o que los propios metadatos de un catálogo sean recolectados por otros). También suele ser habitual que los portales de datos abiertos ofrezcan una almacenamiento alternativo de los metadatos a través de un almacén de tripletas RDF y que este almacén sea accesible a través de un punto SPARQL. Un punto SPARQL proporciona una interfaz de servicio que permite procesar consultas en lenguaje SPARQL que dotan de una mayor flexibilidad que las API REST para consultar los grafos RDF de los metadatos.

A pesar de que técnicamente está resuelta la interoperabilidad entre catálogos y estos facilitan la indexación de los contenidos en la web de forma que puedan ser encontrados a través de motores de búsqueda, en muchas ocasiones la mala calidad del contenido de los metadatos dificulta que se puedan encontrar conjuntos de datos relevantes, o acceder a ellos de la forma apropiada. Por ejemplo, para ilustrar los problemas que pueden surgir en los metadatos publicados por los portales de datos abiertos, la Figura 1.2 muestra un registro de metadatos en formato Turtle (una de las posibles serializaciones de RDF sobre texto) donde se describe de forma errónea un hipotético conjunto de datos sobre observaciones de calidad del aire en la ciudad de Zaragoza:

- Problema de comisión de completitud: El conjunto de datos (*Dataset*) contiene 2 identificadores (*identifier*), pero solo tendría que tener 1 como máximo.
- Problema de omisión de completitud: El conjunto de datos (*Dataset*) contiene la propiedad obligatoria *publisher* para referenciar al proveedor de los datos.
- Problema de consistencia de dominio: El idioma (*language*) debería ser un valor de un sistema lingüístico bien conocido (por ejemplo, código "en" de ISO 639-1) en lugar de texto libre ("English").
- Problema de consistencia conceptual: La URL de acceso (*accessURL*) de la distribución (*Distribution*) enlaza a un potencial fichero RDF, pero se ha indicado que el formato (*format*) es JSON.
- Problema de consistencia temporal: La fecha de creación (propiedad *issued*) es más reciente que la fecha de modificación (propiedad *modified*).

- Problema de clasificación temática: El conjunto de datos está más relacionado con el tema de medio-ambiente que con el tema de educación (valor *educación* en la propiedad *theme*).
- Problema de corrección posicional: Si el conjunto de datos recopila observaciones de la calidad del aire en la ciudad de Zaragoza, su cobertura espacial debería ser la provincia de Zaragoza (en lugar de la provincia de Huesca).

```
<http://datos.gob.es/catalogo/real-time-air-quality-observations>
  a  dcat:Dataset ;
  dct:title "Real time air quality observations in the urban area of
      Zaragoza"@en ;
  dct:identifier "https://opendata.aragon.es/datos/catalogo/dataset/oai-
      zaguan-unizar-es-96845" ;
  dct:identifier "https://opendata.aragon.es/datos/catalogo/dataset/oai-
      zaguan-unizar-es-96846" ;
  dct:description "This dataset provides real time air quality observation
      data, for all the pollutants,
    at the location of each sensor."@en ;
  dc:language "English" ;
  dct:issued "2020-12-11T05:43:39.043550"^^xsd:dateTime ;
  dct:modified "2020-05-15T12:36:38.291865"^^xsd:dateTime ;
  dct:spatial <http://datos.gob.es/recurso/sector-publico/territorio/
      Provincia/Huesca> ;
  dcat:distribution <http://datos.gob.es/catalogo/real-time-air-quality-
      observations/resource/7d6a569502a8> ;
  dcat:theme <http://datos.gob.es/kos/sector-publico/sector/educacion> .

<http://datos.gob.es/catalogo/real-time-air-quality-observations/resource/7
    d6a569502a8>
  a  dcat:Distribution ;
  dct:format <http://datos.gob.es/catalogo/real-time-air-quality-
      observations/resource/7d6a569502a8/format>;
  dct:identifier
    "http://zaguan.unizar.es/record/96845/files/open_data%3
        Areal_time_air_quality_observations.rdf" ;
  dct:title "Real time air quality observations"@en ;
  dcat:accessURL
    "http://zaguan.unizar.es/record/96845/files/open_data%3
        Areal_time_air_quality_observations.rdf" .

<http://datos.gob.es/catalogo/real-time-air-quality-observations/resource/7
    d6a569502a8/format>
  a  dct:IMT ;
  rdf:value  "application/json" .
```

Figura 1.2: Registro de metadatos en formato Turtle conteniendo varios problemas (Fuente: [23]).

Para intentar detectar estos problemas de baja calidad en los contenidos de los catálogos de metadatos, se han planteado estos últimos años distintas metodologías para evaluar la calidad de los portales de datos abiertos. En

particular, dentro del grupo de investigación de Sistemas de Información Avanzados (IAAA) de la Universidad de Zaragoza donde se enmarca este TFG se ha decidido apostar por el estudio e implementación de dos de ellas: la metodología MQA (Metadata Quality Assesment) elaborada por la Comisión Europea [18]; y la metodología propuesta por miembros del grupo IAAA y otros investigadores de la Universidad de Jaén que adapta la norma ISO 19157 de calidad de información geográfica al contexto de los metadatos de portales de datos abiertos [23]:

- La metodología MQA es la base del cuadro de mandos que se ha integrado dentro del Portal Europeo de Datos [26] para monitorizar los contenidos recolectados de los diferentes catálogos de datos abiertos (o portales de datos abiertos) que contribuyen al portal europeo. La metodología MQA está inspirada en los principios FAIR [33], los cuales proporcionan guías para mejorar la facilidad de localización (*findability*), la accesibilidad (*accessibility*), la interoperabilidad (*interoperability*), y la reutilización de recursos digitales (Reusability). En particular, MQA propone 23 dimensiones distribuidas en 5 categorías que chequean el contenido de los metadatos que describen los conjuntos de datos publicados .

- Por otro lado, la metodología basada en la norma ISO 19157 propone 6 categorías, 12 dimensiones y 16 métricas (que generan 45 medidas cuantitativas y cualitativas) para evaluar un conjunto de registros de metadatos en términos de completitud (seguimiento de reglas del esquema de metadatos relativas a la definición de propiedades o elementos obligatorios y su multiplicidad), consistencia (adecuación de los metadatos respecto al dominio de los elementos y coherencia con el esquema propuesto) y exactitud (correlación entre las descripciones y el contenido de los datos). Además, se pone especial atención en evaluar la exactitud de los elementos de metadatos que hacen referencia a fechas y localizaciones, o la legibilidad del texto libre.

Aunque la definición de las metodologías está clara (se describen con mayor detalle en el capítulo 2), en el momento en el que se planteó este TFG no existía una implementación disponible de ambas metodologías que se pudiese aplicar sobre un conjunto de registros de metadatos seleccionado de forma dinámica de un portal de datos abiertos. En el caso de MQA, el Portal de Datos Europeo proporciona una API Rest para consultar un informe de calidad con los resultados de la evaluación MQA de un registro de metadatos individual o de un catálogo, pero no facilita el código fuente para el cálculo del mismo. Dentro del grupo IAAA se había diseñado cómo

calcular las métricas utilizando como elemento principal consultas SPARQL sobre un almacén de tripletas conteniendo los metadatos para la metodología MQA y la metodología basada en ISO 19157 [31, 23], pero no se disponía de un programa que facilitase de forma integrada y repetible la ejecución de estas métricas sobre un conjunto variable de registros de metadatos.

1.2. Objetivos

El objetivo de este TFG ha sido construir un portal web para monitorizar la calidad de los metadatos de portales de datos abiertos en base a los métodos existentes de evaluación de la calidad de los metadatos. A través de este portal se pretende proporcionar una interfaz de usuario que facilite y mejore la experiencia de usuario a la hora de aplicar estos métodos de evaluación de la calidad, añadiendo además características adicionales que puedan resultar de utilidad como puede ser la monitorización automática de un portal de datos abiertos, la presentación gráfica de los resultados de la evaluación, y la exportación de estos resultados con el vocabulario DQV (Data Quality Vocabulary) [9], un vocabulario basado en RDF que es el que más se usa en la actualidad para crear informes de evaluación de la calidad de datos en cualquier dominio.

La nueva aplicación Web permitirá a la persona que esté haciendo uso de ella introducir fácilmente la información necesaria para la realización de la evaluación, utilizar una única metodología o ambas simultáneamente y visualizar los datos de una forma más gráfica y amigable. Además de este tipo de usuario corriente que podrá realizar una evaluación de la calidad sobre un portal en cualquier momento, existe la figura del administrador. El administrador podrá realizar las mismas funciones que un usuario anónimo y además, contará con la característica adicional de poder programar tareas para que estas realicen una evaluación de la calidad en repetidas ocasiones de forma automática.

La descripción de los casos de uso, así como las tablas con los diferentes requisitos funcionales, no funcionales y restricciones que deben contemplarse en la aplicación se muestran en el apéndice A.

1.3. Tecnologías utilizadas

Para implementar la aplicación Web que sirve de interfaz gráfica interactiva para el portal web de monitorización se ha elegido el stack de tecnologías conocido como MEAN[17]. Esta decisión ha sido tomada principalmente de-

bido a la familiarización del autor del Trabajo de Fin de Grado (TFG) con las diferentes tecnologías que se ven implicadas, y la existencia de numerosas librerías que facilitan la implementación y cohesión de las nuevas funcionalidades.

MEAN es un framework basado en JavaScript para el desarrollo de aplicaciones web, cuyo nombre son las siglas de las distintas tecnologías que conforman las capas del framework: MongoDB, Express, Angular y Node. La elección de este framework permite que la implementación de la aplicación web sea sencilla y eficaz, permitiendo que cada una de las tecnologías se especialice en un apartado diferente de la aplicación.

MongoDB[30] es una base de datos NoSQL basada en "documentos" encargada de almacenar de forma persistente los datos generados por la aplicación. Esta tecnología es especialmente interesante debido a la fácil integración con la otra parte del Backend de la aplicación Web y la necesidad de almacenar distintos tipos de datos, como son los usuarios, resultados de análisis, programación de tareas periódicas o ficheros de texto de una longitud indeterminada de forma eficiente.

Express[22] y Node.js[21] son dos tecnologías diferentes pero fuertemente ligadas entre sí. Mientras que Node.js es un entorno de ejecución para JavaScript, puede llegar a ser realmente tedioso implementar una aplicación Web puramente con esta solución. Express permite al desarrollador delegar en el framework tareas relativamente costosas y que se repiten en numerosas ocasiones a lo largo de un proyecto. Así pues, Express posee mecanismos para gestionar las cabeceras de las peticiones entrantes y de salida, enviar respuestas al cliente o realizar el enrutamiento de peticiones que llegan al Backend para ejecutar distintas funciones dentro del servidor, permitiendo además realizar operaciones intermedias como podría ser la autenticación de usuarios en apenas unas pocas líneas de código. Además, desde Node.js se puede integrar el código en Python que realiza el cálculo de las métricas que evalúan la calidad de los metadatos. Como se ha comentado previamente, este proyecto parte de una base existente en la que habían definido consultas SPARQL para el cálculo de algunas de las métricas de las metodología de evaluación de la calidad de los metadatos que se habían integrado a modo de prototipo dentro de scripts elaborados con Python3.7[27].

Angular[2] es a su vez un framework para Frontend que extiende las capacidades de desarrollo de HTML, CSS y JavaScript, permitiendo crear páginas web dinámicas e interactivas que de desarrollarse únicamente mediante código HTML sería imposible.

1.4. Estructura de la memoria

El resto de las secciones de esta memoria están divididas de la siguiente forma:

- El Capítulo 2 describe las dos metodologías de evaluación de la calidad de los metadatos y su representación en DQV, un concepto clave para entender la funcionalidad del portal web de monitorización de la calidad que se ha desarrollado.
- El Capítulo 3 describe el diseño del sistema desarrollado, explicando a su vez, el comportamiento del mismo.
- El Capítulo 4 describe el despliegue y puesta en marcha de la solución propuesta. Para probar la viabilidad se han configurado además un punto SPARQL para la carga local de los metadatos a evaluar, y un catálogo con tecnología CKAN.
- El Capítulo 5 expone cómo la gestión del proyecto y las conclusiones obtenidas por la elaboración del mismo. Además se detallan propuestas de trabajo futuro.

Además, se han incluido dos apéndices: el apéndice A, el cual contiene el catálogo de requisitos y los diagramas de casos de uso asociados; y el apéndice B, el cual contiene las gráficas de algunos resultados obtenidos de la evaluaciones descritas en el apartado 4.3.

Capítulo 2

Metodologías de evaluación de la calidad de los metadatos

Uno de los requisitos principales de este TFG ha sido poder evaluar la calidad de los metadatos de portales de datos abiertos siguiendo distintas metodologías de evaluación. Por ello, era fundamental que los distintos métodos de evaluación de la calidad se pudiesen expresar con un mismo vocabulario y también que los informes de la evaluación se representasen y serializasen de la misma forma. Para conseguir este objetivo, se decidió utilizar el vocabulario Data Quality Vocabulary (DQV) [9].

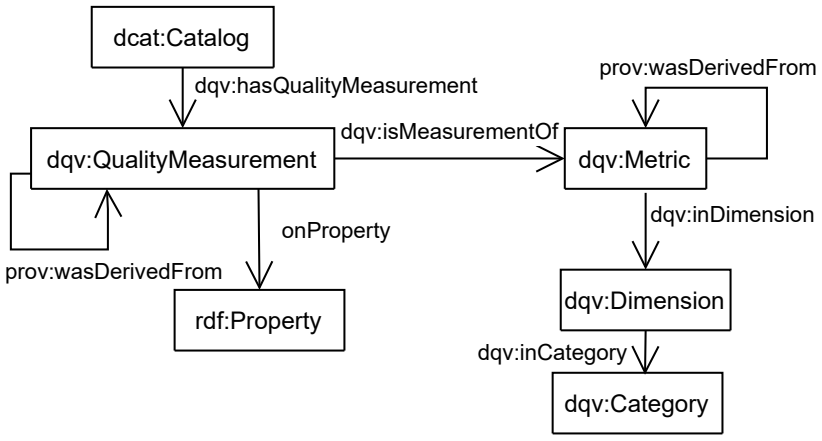

Figura 2.1: Subconjunto del vocabulario DQV necesario para describir la calidad de un catálogo (Fuente:[23]).

DQV es un vocabulario RDF propuesto por W3C que extiende el vocabulario DCAT con propiedades y clases para anotar la calidad de conjuntos de datos (*dcat:Dataset*) y distribuciones (*dcat:Distribution*). En particu-

lar, en este TFG hemos explotado la posibilidad de anotar medidas (clase *dqv:QualityMeasurement*) que hemos asociado directamente a un catálogo (*dcat:Catalog*). La Figura 2.1 muestra un resumen de las principales clases y propiedades del vocabulario DQV. Cada medida es el resultado obtenido después de aplicar una métrica para una dimensión concreta de la calidad. A su vez, cada dimensión está vinculada a una categoría de la calidad.

En las siguientes secciones se explica la metodologías de evaluación de la calidad de los metadatos MQA y la metodología basada en la norma ISO 19157. Además de explicar cómo se calculan las medidas asociadas a las métricas consideradas en cada dimensión y categoría de la calidad, se indica cómo representar estas medidas mediante el vocabulario común de DQV.

2.1. Metodología Metadata Quality Assessment (MQA)

La metodología MQA chequea el contenido de un catálogo de metadatos en base a 23 dimensiones de la calidad distribuidas en 5 categorías:

- Facilidad de localización: se comprueba la disponibilidad de palabras clave, categorías, cobertura espacial y cobertura temporal.
- Accesibilidad: se comprueba la disponibilidad de URL de descarga, y el estado de accesibilidad de las URL de acceso y descarga.
- Interoperabilidad: se comprueba la conformidad de los metadatos respecto a la especificación DCAT-AP (perfil de DCAT para datos abiertos a nivel europeo) [11], la disponibilidad información sobre el formato de distribución, la utilización de formatos bien conocidos, la utilización de formatos no propietarios y la utilización de formatos procesables.
- Reusabilidad: se comprueba la disponibilidad de información sobre licencias, restricciones de acceso, puntos de contactos y editores. También se comprueba la utilización de licencias y restricciones de acceso internacionalmente reconocidas.
- Contextualidad: se comprueba la disponibilidad de información relativa a los derechos de distribución, el tamaño de las distribuciones, y las fechas de creación y distribución de los conjuntos de datos.

En el contexto de este TFG para establecer una correspondencia con el vocabulario DQV se puede asumir que cada una de estas 23 dimensiones tiene asociadas dos métricas y sus correspondientes medidas. La primera métrica

consiste en calcular el porcentaje de registros del catálogo que cumplen con el objetivo de cada dimensión de la calidad. La segunda métrica asociada consiste en calcular un valor de puntos por cada dimensión de calidad cuyo valor máximo por dimensión ha sido establecido por el MQA y que se prorratea en función de la medida de la métrica de porcentaje asociada. En la Tabla 2.1 se puede ver un resumen de esas métricas, así como la puntación máxima asociada a cada una. El objetivo final del MQA es proporcionar una valoración global de la calidad de un catálogo de metadatos sumando las puntuaciones por dimensión y clasificar estos catálogos en 4 categorías en función de su valoración: excelente (entre 351 y 405 puntos), buena (entre 221 y 350 puntos), suficiente (entre 121 y 220 puntos) y mala (entre 0 y 120 puntos).

Categoría	**Dimensión**	**Puntos**
Facilidad de localización	Palabra clave disponible (Dataset/keyword)	30
	Categoría disponible (Dataset/theme)	30
	Cobertura espacial disponible (Dataset/spatial)	20
	Cobertura temporal disponible (Dataset/temporal)	20
Accesibilidad	URL de acceso activa (Distribution/accessURL)	50
	URL de descarga disponible (Distribution/downloadURL)	20
	URL de descarga activa (Distribution/downloadURL)	30
Interoperabilidad	Formato disponible (Distribution/format)	20
	Información de formato MIME disponible (Distribution/mediaType)	10
	Utilización de formato conocido (Distribution/format o Distribution/mediaType)	10
	Utilización de formato no propietario (Distribution/format o Distribution/mediaType)	20
	Utilización de formato procesable (Distribution/format o Distribution/mediaType)	20
	Conformidad con especificación DCAT-AP (todas las entidades y propiedades)	30
Reusabilidad	Licencia disponible (Distribution/license)	20
	Utilización de licencia conocida (Distribution/license)	10
	Información sobre restricciones de acceso disponible (Dataset/accessRights)	10
	Utilización de tipo de restricciones de acceso conocido (Dataset/accessRights)	5
	Punto de contacto disponible (Dataset/contactPoint)	20
	Editor disponible (Dataset/publisher)	10
Contextualidad	Información sobre derechos de distribución disponible (Distribution/rights)	5
	Tamaño de la distribución disponible (Distribution/byteSize)	5
	Fecha de creación disponible (Dataset/issued o Distribution/issued)	5
	Fecha de modificación disponible (Dataset/modified o Distribution/modified)	5

Tabla 2.1: Categorías, dimensiones y puntuaciones máximas de la metodología MQA

Respecto al detalle del cálculo de las métricas de porcentajes en cada dimensión, se puede resumir de la siguiente forma:

- Para las métricas que chequean la disponibilidad de una propiedad, se

debe contabilizar el número de entidades (*Dataset* o *Distribution*) que contienen esa propiedad y calcular el porcentaje respecto al número total de entidades que se están evaluando.

- Para las métricas que chequean la utilización de valores concretos dentro de una propiedad, se deben contabilizar los valores concretos de cada propiedad que se corresponden con los vocabularios específicos recomendados por el Portal Europeo de Datos.[1]

- Para las métricas que chequean el acceso de las URL, se deben contabilizan las direcciones alcanzables después de realizar una petición HTTP cuya respuesta devuelve un código de estado mayor o igual de 200 y estrictamente menor de 400.

- Para chequear la conformidad con la especificación DCAT-AP, se debe contabilizar el número de registros de metadatos cuyo RDF valida frente a los ficheros con restricciones SHAPE de la versión 2.0.1 de DCAT-AP elaborados por la Unión Europea.[2]

Respecto a la representación en formato DQV de los resultados de la evaluación según el vocabulario DQV, la Figura 2.2 muestra un ejemplo en Turtle [28] (serialización de RDF sobre texto) de las métricas y resultados asociados a la dimensión "Palabra clave disponible" dentro de la categoría "Facilidad de localización".

2.2. Metodología de la calidad basada en la norma ISO 19157

La metodología de evaluación de la calidad basada en la norma ISO 19157 [23] analiza un amplio rango de aspectos relacionados con la completitud, la consistencia y la exactitud de acuerdo a los elementos de la calidad (dimensiones según DQV) propuestas por ISO 19157. Otra característica especial de esta propuesta de evaluación de la calidad es que está centrada en realizar controles de calidad. Un control de calidad determina sin un parámetro (dimensión de calidad) de un producto satisface un requisito específico expresado como un nivel de calidad, por ejemplo, no más de un 5 % de errores. En el control de la calidad se pueden dar dos escenarios diferentes. El primer escenario ocurre cuando la automatización de un proceso de control es posible

[1] `https://gitlab.com/european-data-portal/edp-vocabularies`

[2] `https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/dcat-application-profile-data-portals-europe/release/201-0`

y se puede chequear la población completa para el tipo de error que pueda existir. El segundo escenario ocurre cuando la automatización no es posible y hay que realizar un control basado en muestreo para derivar una decisión que implique riesgos limitados. El mecanismo para implementar los controles de calidad en la norma ISO 19157 es definir dos medidas relacionadas: una primera medida conteniendo un valor cuantitativo asociado a una métrica de las propuestas por ISO 19157 para una dimensión de calidad específica; y una segunda medida derivada de la primera que contiene un resultado de conformidad de la primera indicando si se satisface el nivel de calidad esperado o no.

En la Tabla 2.2 se muestra la jerarquía de categorías (*Quality category*) y dimensiones (*Quality element* en la terminología de ISO 19157) consideradas para los metadatos junto a las métricas (*Measure* en la terminología de ISO 19157) y una breve descripción de las mismas (*Measure description*). La mayor parte de las métricas se corresponden con métricas incluidas en el anexo D de la norma ISO 19157. También hay algunas medidas denominadas "similar to" para denotar que la formulación matemática de la métrica referenciada es idéntica. También se hace notar que las métricas definidas en términos de tasas (*rates*) se corresponden con controles automáticos que se aplican sobre toda la población. Por el contrario, las métricas definidas en términos del número de items correctos/incorrectos se corresponden con evaluaciones manuales sobre una muestra de la población que deben ser realizada por expertos.

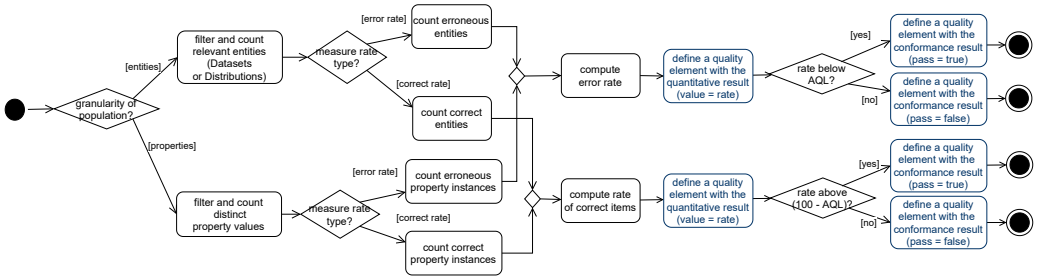

Figura 2.3: Diagrama de actividad para el computo de las medidas asociadas a métricas que se pueden calcular de forma automática (Fuente: [23])

Dentro de este TFG, ya que el objetivo es automatizar la evaluación de la calidad de los metadatos, nos hemos centrado en implementar aquellas dimensiones de la calidad y medidas asociadas que se puedan calcular de forma automática sobre toda la población de registros de metadatos. En la Figura 2.3 se puede ver un diagrama de actividad para el computo de las medidas asociadas a métricas automáticas. En este diagrama se puede observar que

hay dos tipos de métricas: métricas centradas en calcular tasas/porcentajes de errores, y métricas centradas en calcular tasas/porcentaje de aciertos. En ambos casos se menciona el uso del término AQL. AQL es el acrónimo de *Acceptance Quality Limit* o limite de calidad de aceptación, que sería el porcentaje de errores que podemos aceptar para pasar un control de calidad. En esta metodología se aplica un AQL de 4.16 %, que suele ser un parámetro habitual en este tipo de controles de calidad. Además, se puede observar también que las métricas trabajan o bien sobre entidades (*Dataset* o *Distribution*) o sobre propiedades concretas de estas entidades. La posibilidad de aplicar las dimensiones y métricas automáticas de la calidad de la Tabla 2.2 sobre distintas entidades y propiedades permite calcular 38 medidas con valores de porcentajes y 38 medidas de conformidad.

La Figura 2.4 muestra un ejemplo en Turtle de la representación de las dimensiones , métricas y medidas de esta metodología de evaluación de la calidad sobre DQV. La medida con el valor cuantitativo (*DQ_ComComDat_QR*) hace referencia a la métrica D.3 de ISO 19157 (*D.3.ISO.19157*). Esta es una medida asociada a la dimensión de comisión de completitud, la cual pertenece a la categoría de completitud (*DQ_Completeness*) de ISO 19157. Además, se indica la propiedad de los metadatos sobre la que se aplica: la propiedad *dcat:dataset* que enlaza un catálogo con sus conjuntos de datos. También se puede observar como el resultado de conformidad (*DQ_ComComDat_CR*) hace referencia a la métrica que se ha creado para indicar si la tasa de error D.3 de ISO 19157 está por debajo del AQL (*D.3.ISO.19157_conformance*). La propiedad *prov:wasDerivedFrom* indica que tanto la medida de conformidad como la métrica de conformidad se derivan de las correspondientes versiones cuantitativas.

```
#Quality report
:myCatalog a dcat:Catalog ;
    dcterms:title "datos.gob.es" ;
    dqv:hasQualityMeasurement
        :KeywordAvailability_rate_value ,:KeywordAvailability_points_value.

:KeywordAvailability_rate_value a dqv:QualityMeasurement ;
    dqv:computedOn :myCatalog ;
    dqv:isMeasurementOf :KeywordAvailability_rate ;
    dqv:value "100.0"^^xsd:double ;
    dct:date "2020-03-01"^^xsd:date ;
    :onProperty dcat:dataset, dcat:keyword .

:KeywordAvailability_points_value a dqv:QualityMeasurement ;
    dqv:computedOn :myCatalog ;
    dqv:isMeasurementOf :KeywordAvailability_points ;
    prov:wasDerivedFrom :KeywordAvailability_rate_value ;
    dqv:value "30.0"^^xsd:double ;
    dct:date "2020-03-01"^^xsd:date ;
    :onProperty dcat:dataset, dcat:keyword .

#definition of categories, dimensions and metrics
:Findability a dqv:Category ;
    skos:prefLabel "Findability"@en ;
    skos:definition "Metrics that help people and machines in finding
        datasets."@en .

:KeywordAvailability a dqv:Dimension ;
    dqv:inCategory :Findability ;
    skos:prefLabel "Keyword usage"@en ;
    skos:definition "Keywords directly support the search and thus increase
        the findability of the data dataset."@en .

:KeywordAvailability_rate a dqv:Metric ;
    skos:definition "Rate metric: The system checks whether keywords are
        defined. The number of keywords has no impact to the score."@en ;
    dqv:inDimension :KeywordAvailability ;
    dqv:expectedDataType xsd:double .

:KeywordAvailability_points a dqv:Metric ;
    prov:wasDerivedFrom :CategoryAvailability ;
    skos:definition "Points metric (0-30): The system checks whether
        keywords are defined. The number of keywords has no impact to the
        score."@en ;
    dqv:inDimension :KeywordAvailability_rate ;
    dqv:expectedDataType xsd:double .

#Parameters of the measurements (derived from the DQV specification)
:onProperty
    a qb:DimensionProperty, rdf:Property ;
    rdfs:comment        "Property on which the quality measure is assessed."
        @en ;
    rdfs:domain         dqv:QualityMeasurement ;
    rdfs:label          "Label assessment property"@en ;
    rdfs:range          rdf:Property.
```

Figura 2.2: Fragmento de una evaluación de calidad según la metodología MQA en formato DQV (turtle).

Tabla 2.2: Categorías, dimensiones de la calidad y métricas de la metodología basada en ISO 19157.

Quality category	**Quality element**	**Measure**	**Measure description**
DQ_Completeness	DQ_CompletenessCommission	D.3 (ISO 19157)	Rate of records with excess items.
	DQ_CompletenessOmission	D.7 (ISO 19157)	Rate of records with missing items.
DQ_LogicalConsistency	DQ_ConceptualConsistency	D.13 (ISO 19157)	Rate of records compliant with the conceptual schema.
		Similar to D.22 (ISO 19157)	Number of records with inconsistent information in metadata elements.
	DQ_DomainConsistency	D.17 (ISO 19157)	Value domain conformance rate.
	DQ_FormatConsistency	D.21 (ISO 19157)	Physical structure conflict rate.
	DQ_TopologicalConsistency	D.23 (ISO 19157)	Rate of records having faulty relationships with other records in the catalog.
		Topological contradiction	Rate of records having faulty relationships between two metadata elements of the same record.
DQ_TemporalQuality	DQ_TemporalConsistency	Similar to D.62 (ISO 19157) using a rate	Rate of records with conflict time sequences.
	DQ_TemporalValidity	D.18 (ISO 19157)	Value domain non-conformance rate.
DQ_ThematicAccuracy	DQ_ThematicClassificationCorrectness	D.63 (ISO 19157)	Number of incorrectly classified records.
	DQ_NonQuantitativeAttribute Correctness	D.69 (ISO 19157)	Rate of incorrect attribute values.
		D.68 (ISO 19157)	Number of records with incorrect attribute values.
	DQ_PositionalCorrectness	Similar to D.33 (ISO 19157)	Rate of records with positional errors: no overlapping between direct and indirect georeferences.
	DQ_QualityOfFreeText	Overall quality of free text	Number of records using text values with a bad quality level.
		Readability of free text	Rate of records using text values considered readable with a readability index (e.g. Flesch) above a threshold.

```
#Quality report
:myCatalog a dcat:Catalog ;
    dcterms:title "datos.gob.es" ;
    dqv:hasQualityMeasurement :DQ_ComComDat_CR, :DQ_ComComDat_QR .

:DQ_ComComDat_QR a dqv:QualityMeasurement ;
    dqv:computedOn :myCatalog ;
    dqv:isMeasurementOf :D.3.ISO.19157 ;
    dqv:value "0.0"^^xsd:double ;
    dct:date "2020-03-01"^^xsd:date ;
    :onProperty dcat:dataset .

:DQ_ComComDat_CR a dqv:QualityMeasurement ;
    dqv:computedOn :myCatalog ;
    dqv:isMeasurementOf :D.3.ISO.19157_conformance ;
    prov:wasDerivedFrom :DQ_ComComDat_QR ;
    dqv:value true ;
    dct:date "2020-03-01"^^xsd:date ;
    :onProperty dcat:dataset .

#definition of categories, dimensions and metrics
:DQ_Completeness a dqv:Category ;
    skos:prefLabel "Completeness"@en ;
    skos:definition "Completeness refers to the degree in which the metadata
         elements are present or absent."@en .

:DQ_CompletenessCommission a dqv:Dimension ;
    dqv:inCategory :DQ_Completeness ;
    skos:prefLabel "Completeness commission"@en ;
    skos:definition "Completeness commission refers to the degree in which
        there are excess instances of metadata elements in a metadata record
        ."@en .

:D.3.ISO.19157 a dqv:Metric ;
    skos:definition "Rate of records with excess items."@en ;
    dqv:inDimension :DQ_CompletenessCommission ;
    dqv:expectedDataType xsd:double .

:D.3.ISO.19157_conformance a dqv:Metric ;
    prov:wasDerivedFrom :D.3.ISO.19157 ;
    skos:definition "Checks if the rate of records with excess items is
        below AQL (statistical error level)."@en ;
    dqv:inDimension :DQ_CompletenessCommission ;
    dqv:expectedDataType xsd:boolean .
```

Figura 2.4: Fragmento de una evaluación de calidad según la metodología basada en la norma ISO 19157 en formato DQV (turtle)

Capítulo 3

Desarrollo

3.1. Arquitectura

La aplicación web desarrollada en este proyecto está basada en una arquitectura en tres capas, a partir del stack tecnológico MEAN y su integración con los analizadores de calidad de metadatos desarrollados en código Python. La Figura 3.1 muestra esta arquitectura en en capas mediante un diagrama de paquetes UML donde se representan los diferentes subsistemas. La capa de presentación ha sido, por tanto, desarrollada mediante Angular[2]. La capa de negocio se compone tanto de NodeJS[21] y Express[22] como del uso de las librerías de Python que realizan los análisis de calidad de metadatos. Por último, la capa de datos está compuesta por MongoDB[30] para asegurar la persistencia de los resultados obtenidos de los análisis y alguna librería adicional para determinadas funciones como Agenda[1] o Passport[25].

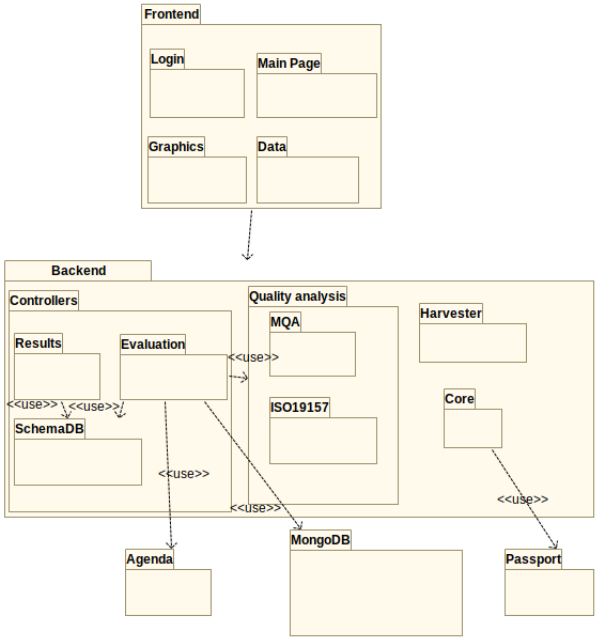

Figura 3.1: Diagrama de paquetes

La capa de negocio también centraliza todas las conexiones con librerías para la gestión de usuarios y la programación de tareas junto con la base de datos, ya que como se explicará posteriormente, solo un tipo de usuario puede programar y almacenar tareas para ejecutar de forma periódica dentro de la base de datos.

La Figura 3.2 representa las diferentes partes lógicas del software desplegado en su versión final mediante un diagrama de componentes UML. También se representan los accesos a los distintos puntos SPARQL y CKAN que pueden no estar controlados en el mismo entorno.

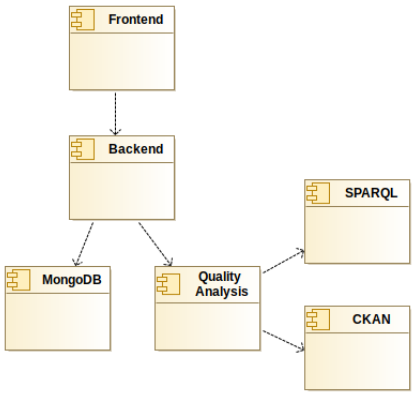

Figura 3.2: Diagrama de componentes

En los siguientes apartados se detalla el funcionamiento de la aplicación web separando, en primer lugar, de la capa de negocios y datos, y posteriormente de la capa de presentación.

3.2. Subsistema Backend

El Backend tiene dos objetivos principales: por un lado, facilitar la evaluación de un catálogo concreto de un portal de datos abiertos; por otro lado, también facilita la posibilidad de crear tareas periódicas de evaluación de la calidad.

A continuación, se explica cómo se han diseñado estas dos funcionalidades principales a través de los distintos subsistemas involucrados en el Backend.

3.2.1. Evaluación

En la Figura 3.3 se muestra un diagrama de actividad con el flujo de trabajo seguido al recibir una petición de evaluación. Cuando se recibe una petición de evaluación, el sistema debe diferenciar entre los distintos tipos de peticiones que se pueden realizar, ya que en función de la metodología o forma de evaluación, existen diferencias que deben ser tenidas en cuenta.

Cuando se debe evaluar un catálogo de datos que se encuentra accesible de manera pública, se recibe el enlace a la dirección correspondiente, de forma que dependiendo de si es un modelo de calidad ISO 19157 o MQA se ejecutarán las evaluaciones de la calidad implementadas en diferentes programas escritos en Python.

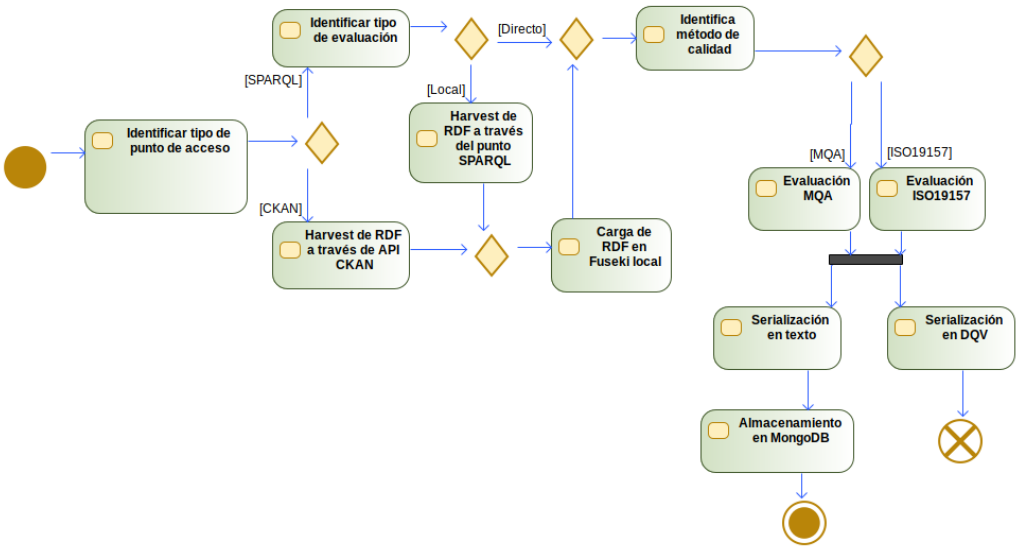

Figura 3.3: Diagrama de actividad del proceso de evaluación

Debido a que los analizadores realizan numerosas peticiones a un mismo portal, lo cual puede demorarse durante horas, es posible que estos portales lleguen incluso a bloquear al al programa de evaluación de la calidad justificando un posible ataque al servicio. Para poder evitar estos casos, así como otros posibles escenarios que puedan ser convenientes para el usuario, existe la posibilidad de descargar localmente el fichero RDF con los metadatos del portal haciendo uso de procedimientos de harvest (recolección) a través de las API disponibles de los portales. De esta forma, el sistema, de forma automática y transparente al usuario, descarga dicho fichero y lo carga en un servidor local de Fuseki [3] controlado por el administrador de la aplicación web. De esta forma, los datos son evaluados sobre este servidor en lugar del portal original de forma directa. Así pues, es menos probable que se bloqueen las peticiones, pues aun con todo, existen pruebas de acceso a direcciones web a las cuales se deben enviar peticiones para comprobar su disponibilidad.

3.2.1.1. QualityAnalysis

La Figura 3.4 muestra las clases principales que realizan la evaluación de la calidad para las dos metodologías ISO19157 y MQA. Además se muestran las clases de IndicePerspicuidad y SPARQLWrapper que se utilizan para realizar las evaluaciones correctamente, así como el uso de la librería rdflib para almacenar los resultados en un fichero DQV.

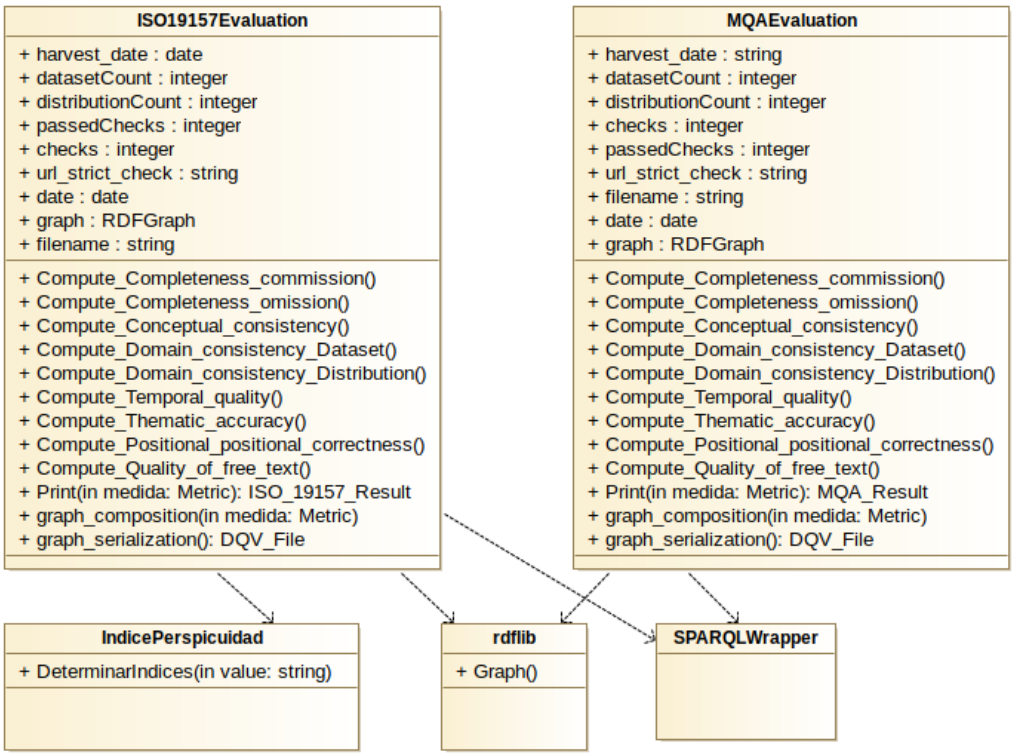

Figura 3.4: Descripción subsistema QualityAnalysis

Cuando el subsistema QualityAnalysis es invocado a través del controlador desarrollado en NodeJS, se realiza por medio de la librería llamada Python-Shell [12], que permite una ejecución en paralelo a la ejecución principal del servidor, aspecto de vital importancia, pues los análisis de algunos catálogos pueden llegar demorarse durante varias horas.

Los evaluadores de la calidad se comportan de forma similar tanto si se trata del modelo de calidad ISO19157 o MQA. Cuando estos son invocados, se inician los datos necesarios para realizar las diferentes medidas y el grafo RDF[20] que las almacenará. Este grafo se genera a partir de una plantilla en formato Turtle que contiene la definición de las métricas que se evaluarán, en función del modelo de calidad correspondiente. Una vez inicializado el proceso, se realizan las peticiones al portal a través de SPARQLWrapper de forma secuencial. Esto hace que, especialmente cuando se comprueban las medidas de accesibilidad, la ejecución del programa pueda demorarse durante un largo periodo de tiempo; siempre en función de la cantidad de datos evaluados.

Cuando se obtienen los datos necesarios, se calcula la métrica en función de estos, teniendo en cuenta que unas calculan el porcentaje de aciertos

mientras que otras el de errores. Por tanto, se realiza una corrección para devolver todos los datos en porcentaje de aciertos y que estos puedan ser comparables entre sí. Una vez realizadas dichas correcciones, se deben realizar dos trabajos. Por un lado, los resultados finales de la métrica son devueltos al controlador que invocó al evaluador. Por otro lado, se genera un grafo RDF mediante la librería rdflib [20] de forma que se agregan las tripletas generadas por cada una de las medidas de calidad efectuadas a lo largo de la ejecución de la evaluación. Una vez que han terminado de evaluarse todas las métricas, el grafo serializa las tripletas almacenadas en un fichero DQV en formato Turtle finalizando así su ejecución.

3.2.1.2. Almacenamiento de los resultados de la evaluación en MongoDB

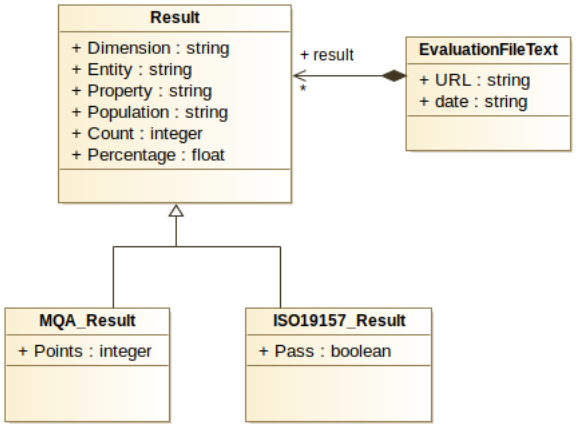

Figura 3.5: Estructura de los datos almacenados en MongoDB

Una vez resuelta la evaluación de forma exitosa, los resultados generados por el subsistema QualityAnalysis, son devueltos al controlador que inició el analizador. Estos datos son tratados para que puedan ser identificados posteriormente y representados de forma gráfica tal y como se muestra en la Figura 3.5.

El fichero DQV generado por el analizador ha sido creado en una carpeta común con el controlador que lo invocó, de forma que ahora el controlador, únicamente debe identificar el fichero correspondiente a su evaluación y almacenarlo en la base de datos para que este pueda ser exportado por los usuarios en un futuro a través de dos colecciones distintas. Esta separación en dos colecciones para cada uno de los ficheros se realiza para evitar que un fichero excesivamente grande produzca un desbordamiento en la base de

datos MongoDB debido al tamaño máximo que admite este para cada uno de los objetos almacenados[10]. Eso provoca que la información que identifica a un fichero se almacena en una colección llamada "fs.files", mientras que este en realidad queda dividido en N fragmentos dentro de la colección "fs.chunks" que son agrupados y devueltos como un único fichero cuando debe ser exportado. De esta forma, se garantiza, independientemente del tamaño del fichero generado, que se podrá almacenar y exportar correctamente.

3.2.2. Programación de tareas de evaluación

Actualmente, se espera que el sistema completo esté desplegado en un entorno controlado, sin una gran afluencia de peticiones realizadas por usuarios ajenos al mismo. Por este motivo, solamente se contempla que un administrador deba autenticarse para realizar acciones especiales como la programación de tareas. Si esto sucede, el controlador que invoca a los evaluadores de la calidad, también programa las tareas en función de los criterios del administrador para que se vuelvan a ejecutar las evaluaciones descritas previamente según corresponda.

La autenticación de este tipo de usuario se realiza por medio de la librería Passport[25]. Actualmente, los usuarios se autentican de forma local al sistema, esto es, un usuario debe estar almacenado en la base de datos de la propia aplicación web.

Para gestionar la programación de tareas, la librería llamada Agenda [1] permite integrarse con la base de datos de forma sencilla. Gracias a esto, cuando una tarea es programada, se almacena el periodo en el que debe ejecutarse nuevamente así como cierta información a criterio del desarrollador para limitar el uso de estas en función de las previsiones que se tengan de las mismas, como puede ser el número de ejecuciones concurrentes de tareas programadas. Sin embargo, la característica más importante para el ámbito de este proyecto es el hecho de detectar las tareas que se encontraban en ejecución en caso de una parada inesperada del servicio y las arranca de nuevo automáticamente cuando este vuelve a estar operativo, de forma que estas tareas son persistentes.

3.3. Subsistema Frontend

El subsistema encargado de proporcionar la interfaz gráfica interactiva al usuario se ha implementado mediante Angular[2]. La interfaz web se compone de cuatro pantallas para dar servicio al usuario.

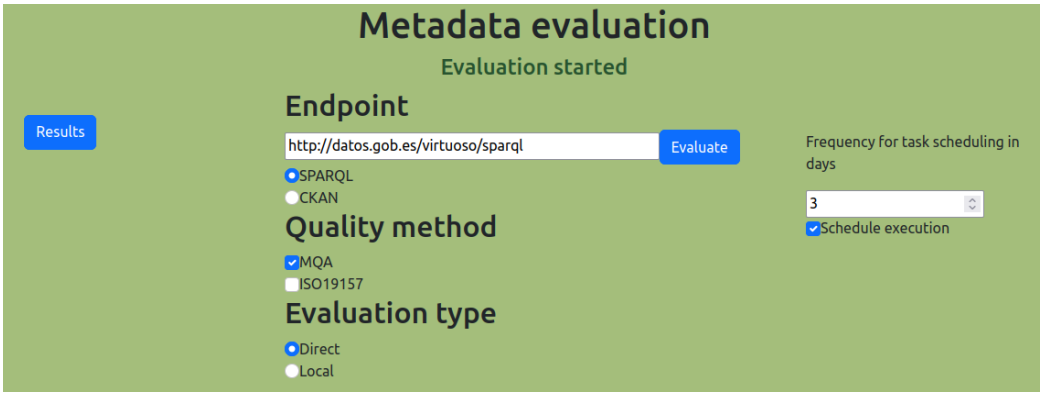

Figura 3.6: Pantalla principal

La pantalla principal, mostrada en la Figura 3.6, es la primera página a la que accede un usuario. En esta pantalla se puede realizar la evaluación de un catálogo de metadatos permitiendo al usuario elegir distintas opciones. Estas pueden ser la selección del modelo de calidad a utilizar, el tipo de portal al que se accederá, así como la indicación de si se debe evaluar la dirección indicada o bien, se trata de un enlace a un catálogo de datos RDF el cual se debe descargar y cargar en un portal controlado por el administrador del servicio para su posterior evaluación.

Debido a que la ejecución de las evaluaciones puede demorarse durante horas, el portal únicamente indica si se ha comenzado la tarea correctamente, pero no si se produce algún fallo a lo largo de la misma.

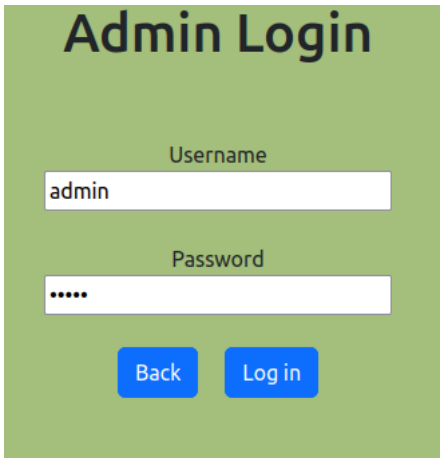

Figura 3.7: Pantalla de login del administrador

La pantalla de login (Figura 3.7) habilita a un usuario registrado previamente en el sistema a realizar programaciones de tareas de evaluación

para que estas se ejecuten de forma periódica.

URL	Date	Method
http://datos.gob.es/virtuoso/sparql	2022-12-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-12-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-15-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-15-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-17-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-17-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-17-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-17-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-17-11	MQA
http://datos.gob.es/virtuoso/sparql	2022-12-11	ISO19157

Figura 3.8: Pantalla con las evaluaciones disponibles para consulta

La Pantalla de resultados (Figura 3.8) está disponible para consultar por cualquier usuario los resultados que se han obtenido anteriormente, independientemente del usuario que haya realizado la evaluación. Estos resultados son todos los que se encuentran almacenados en la base de datos y que, por tanto, han sido ejecutados a lo largo del tiempo de vida de la aplicación web. De esta forma, antes de realizar la evaluación de un portal de datos, un usuario puede acceder a esta página para comprobar si existen evaluaciones previas sobre dicho portal y su antigüedad por si le resultaran útiles y evitar la espera que puede suponer realizar una nueva evaluación.

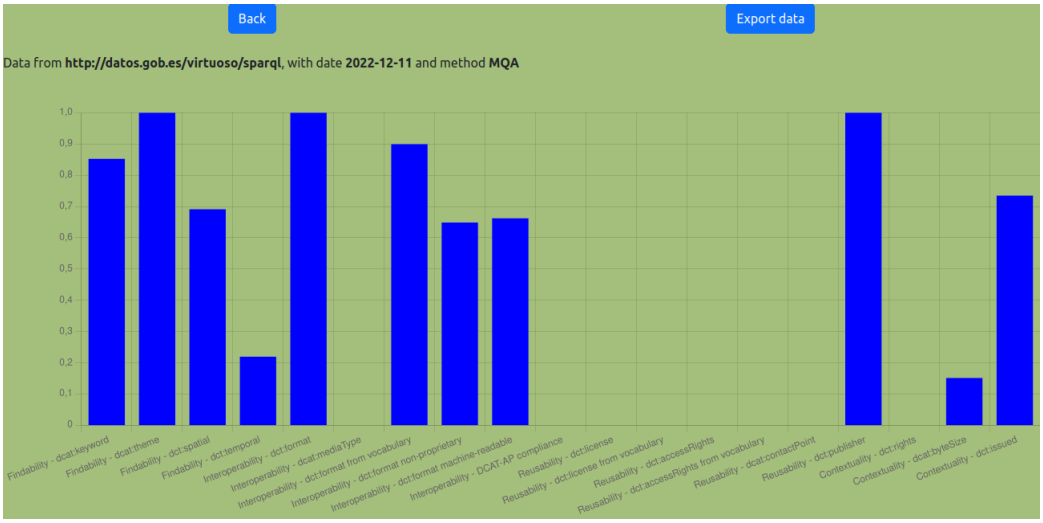

Figura 3.9: Pantalla con la gráfica de los resultados obtenidos

La Figura 3.9 muestra un ejemplo de las gráficas que se pueden obtener con los resultados de la evaluación de la calidad previamente realizada y seleccionada. Estos resultados se muestran de forma gráfica de forma que el usuario pueda interpretar el estado general del portal en función del porcentaje de datos que ha superado el criterio de calidad según el modelo. Además, en esta pantalla se permite al usuario exportar en un fichero DQV en formato Turtle los datos que se están consultando de forma libre, sin necesidad de estar registrado en la aplicación.

Un ejemplo de una gráfica proporcionada por el modelo de calidad ISO19157 puede ser observado en la Figura 7 del apéndice B. En ella se puede observar cómo se representa en color rojo aquellas medidas que no alcanzan el umbral de conformidad establecido.

Capítulo 4

Puesta en marcha

La aplicación web ha sido diseñada para poder desplegar todos los componentes necesarios (Frontend, Backend y base de datos MongoDB) sobre la misma máquina. Sin embargo, esto no es un requisito y se podrían instalar los componentes en nodos de procesamiento separados dependiendo de la carga de trabajo exigida. Siguiendo el concepto de construcción de la aplicación en tres capas y niveles (tier), el despliegue de la aplicación queda representado en el el diagrama de despliegue que se muestra en la Figura 4.1.

Respecto al detalle concreto de los artefactos desplegados en cada nodo, el proyecto se desarrolla con dos ficheros llamados "package.json", uno para la parte de la interfaz web desarrollada en Angular[2] y el otro para el servidor desarrollado con NodeJS[21]. Suponiendo que existe una instancia de MongoDB[30] ejecutándose de forma accesible, se facilita la tarea de inicio de la aplicación web, pues únicamente se deben instalar las dependencias y ejecutar el script que comienza la ejecución en cada uno de los ficheros.

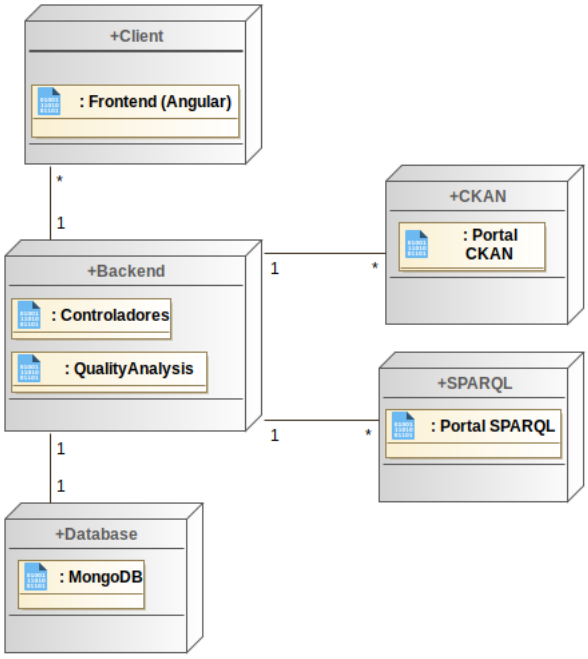

Figura 4.1: Diagrama de despliegue

Debido a que existen varias librerías que dependen entre sí en versiones concretas, se recomienda crear un entorno virtual de Python[32]. Este módulo permite la creación de un entorno aislado y completamente independiente del entorno de la máquina en la que se ejecute la aplicación web y las posibles versiones de Python que esta pueda tener previamente instaladas. En este entorno se debe instalar la versión Python3.7 junto con las dependencias definidas para cada uno de los analizadores que se ejecutarán posteriormente para realizar las evaluaciones de calidad de los metadatos.

Por último, en la Figura 4.1 se muestran dos nodos correspondientes a catálogos desarrollados con la tecnología CKAN y a catálogos accesibles como un punto SPARQL. En ambos casos, y tal como se explica en las siguientes secciones, se han desplegado portales de prueba con contenedores Docker. Además, en el caso del punto SPARQL este tipo de nodo se utiliza como repositorio temporal de los contenidos de un portal de datos abiertos que se quiere evaluar localmente sin hacer peticiones directas al portal.

4.1. Fuseki

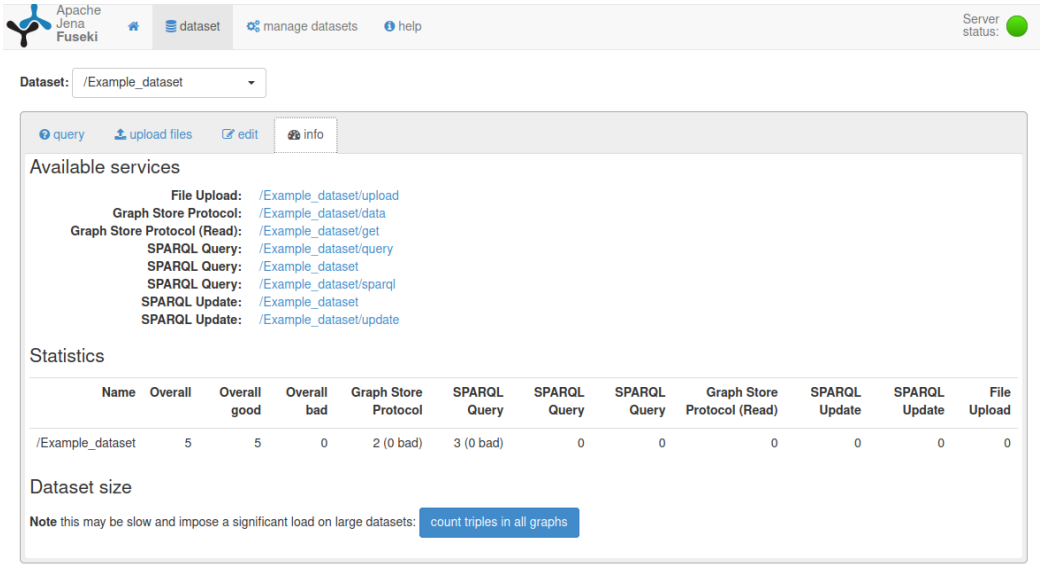

Figura 4.2: Portal de Fuseki sobre docker

Para realizar consultas sobre un punto SPARQL de forma local durante el periodo de pruebas, se ejecuta un portal de Fuseki sobre docker[29]. Esta solución permite que el portal pueda ser accedido y controlado de forma sencilla al encontrarse dentro de la misma máquina que el resto de la aplicación web. Además, este portal permite cargar los catálogos a través de una API y acceder a los mismos posteriormente.

Este portal proporciona una interfaz web que facilita determinadas tareas como puede ser el comprobar que el proceso automático de descarga y carga de un catálogo de metadatos se haya realizado con éxito. Esto se puede realizar gracias a que el mismo portal muestra información sobre el catálogo, tal como aparece en la figura 4.2 y permite realizar consultas SPARQL directamente.

4.2. CKAN

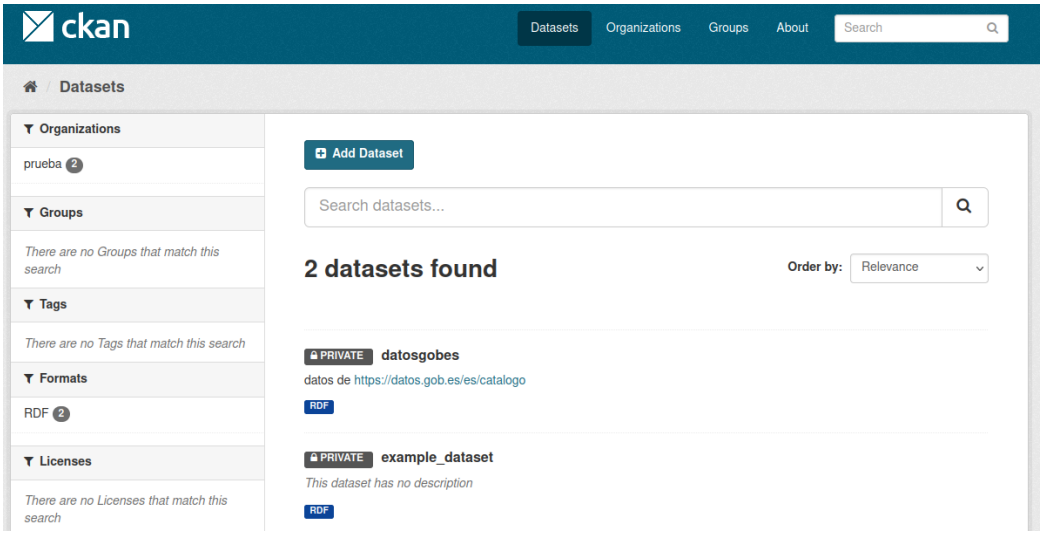

Figura 4.3: Portal de CKAN sobre docker

La instalación de un punto CKAN mediante docker (Figura 4.3) se ha llevado a cabo a través de la guía oficial[13]. La versión utilizada a lo largo de este proyecto ha sido la versión CKAN 2.9.4, instalado mediante la utilización de Python 3.7 en una máquina Ubuntu 20.04 64-bit. Además, se han instalados dos extensiones adicionales:

- En primer lugar, se ha instalado ckanext-spatial[14]. Esta extensión permite soportar campos espaciales particulares, la indexación de estos campos espaciales y la recolección de catálogos CSW. Debido a la compatibilidad de la extensión con Python2, es necesario asegurarse que la instalación de librerías requeridas se realiza mediante "pip3 install -r pip-requirements.txt" en el momento indicado por la documentación.

- Por otro lado, se ha instalado ckanext-dcat[5]. Esta extensión permite consumir metadatos de catálogos usando documentos RDF[7] serializados mediante DCAT[8]. De forma similar que la anterior extensión, tal y como se indica en la guía de instalación, cuando se instalan las librerías necesarias para su funcionamiento, se debe hacer a través del fichero "requirements-py2-py36.txt" en lugar del indicado por defecto.

4.3. Configuraciones probadas

Aunque a lo largo de la elaboración del proyecto, se han realizado numerosas pruebas, algunos ejemplos de evaluaciones más relevantes se exponen a continuación:

- La figura 6 del apéndice B muestra una evaluación del método de calidad MQA para una API SPARQL evaluado directamente sobre el portal de datos https://datos.gob.es/es/catalogo, que contiene 63.863 conjuntos de datos.
- La figura 7 del apéndice B muestra una evaluación del método de calidad ISO 19157 para una API SPARQL evaluado directamente sobre el portal de datos https://datos.gob.es/es/catalogo, que contiene 63.863 conjuntos de datos.
- La figura 8 del apéndice B muestra una evaluación del método de calidad MQA para una API CKAN evaluado de forma local sobre Fuseki del portal de datos https://datos.santiagodecompostela.gal/catalogo, que contiene 60 conjuntos de datos.
- La figura 9 del apéndice B muestra una evaluación del método de calidad ISO 19157 para una API CKAN evaluado de forma local sobre Fuseki del portal de datos https://datos.santiagodecompostela.gal/catalogo, que contiene 60 conjuntos de datos.

Capítulo 5

Gestión, conclusiones y trabajo futuro

5.1. Gestión

En el diagrama de gantt de la Figura 5.1 se puede observar las dos fases principales del proyecto. En primer lugar se dedicó un tiempo aproximado de un mes y medio para la familiarización del proyecto. Este apartado incluye el estudio de la terminología y las herramientas que se iban a emplear, así como la configuración de herramientas que se utilizarían más adelante. También se realiza un primer boceto de un prototipo de la aplicación Web sobre el que apoyarse a la hora de comprender los distintos aspectos del proyecto y el impacto que cada uno de ellos podría suponer.

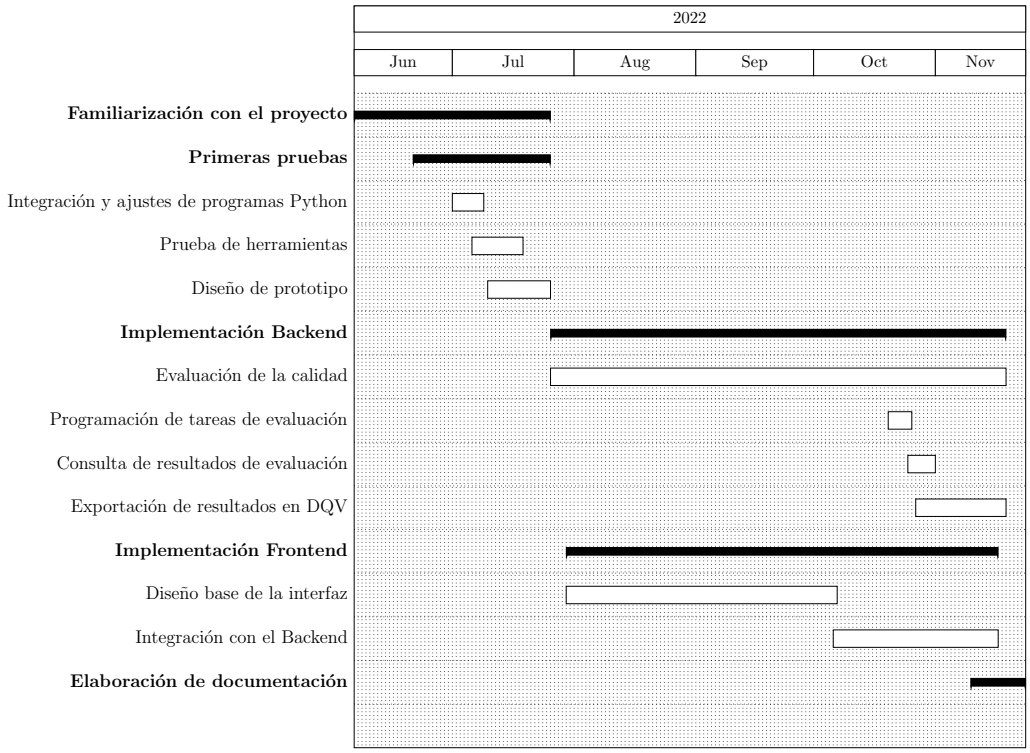

Figura 5.1: Diagrama Gantt sobre la dedicación de esfuerzos al proyecto

Por otra parte, se encuentra el diseño y la implementación tanto del software que ejecuta la parte del servidor, como la del cliente de una forma más o menos paralela según el grado de desarrollo del proyecto. La mayor parte del tiempo ha sido invertida en realizar la integración del código de los evaluadores de la calidad de metadatos con el resto de la aplicación Web para poder asegurar su automatización y consistencia ante fallos, así como la paralelización de tareas sin que estas supongan un riesgo para la ejecución de otras. Una vez se tiene desarrollada esta parte del proyecto, la ampliación de características como la programación periódica de tareas, la identificación de usuarios o la exportación de datos ha supuesto un esfuerzo menor, aunque ello haya requerido replantear el diseño previo del código.

En total se han invertido 375 horas en la elaboración de este proyecto, cuyo desglose de horas aproximado para cada una de las tareas ha sido el siguiente:

- Familiarización con el proyecto: 10 horas
- Primeras pruebas

  Integración y ajustes de programas Python: 15 horas

Prueba de herramientas: 25 horas

Diseño de prototipo: 5 horas

- Implementación Backend

  Evaluación de la calidad: 135 horas

  Programación de tareas de tareas de evaluación: 15 horas

  Consulta de resultados de evaluación: 30 horas

  Exportación de resultados de evaluación en DQV: 20 horas

- Implementación Frontend

  Diseño base de la interfaz: 30 horas

  Integración con el Backend: 50 horas

- Elaboración de documentación: 40 horas

Para realizar el seguimiento del estado de las tareas y planificación de las mismas según su importancia se ha utilizado la herramienta Microsoft To-Do [4]. Esta herramienta permite la organización de tareas en distintas listas permitiendo a su vez, crear subtareas y ubicarlas en diferentes apartados. Logrando de esta forma una metodología similar a Kanban [15].

El código ha sido guardado en todo momento en un repositorio ubicado en GitHub[16], permitiendo de esta forma despreocuparme de la posibilidad de perder parte o todo el código y tener un control de versiones sobre los progresos que iba realizando, ya que al hacer uso de herramientas y tecnologías desconocidas hasta ahora por mí, en ocasiones se han producido fallos que he debido revertir.

Para la documentación realizada expuesta en este documento, se ha hecho uso de la herramienta Overleaf[24]. Esta herramienta online ha permitido desarrollar un documento de forma que el director del proyecto podía revisar y realizar comentarios sobre el mismo en cualquier momento, sin necesidad de concertar una reunión para ello y agilizando de esta forma las reuniones presenciales.

5.2. Conclusiones

Se ha podido implementar con éxito una aplicación web que facilita el uso de herramientas de evaluación de la calidad de los metadatos de portales de datos abiertos, permitiendo que el usuario pueda seleccionar los portales a

evaluar junto con otros parámetros de la evaluación. Por otro lado, las funcionalidades de visualización de la aplicación permiten realizar un mejor análisis sobre la calidad de los metadatos en portales de datos abiertos, haciendo que los resultados de la evaluación de la calidad sean fácilmente accesibles y legibles por los usuarios. Además, la implementación de nuevas características como puede ser la exportación de resultados en un fichero DQV, permitirá a los usuarios evaluar y utilizar dichos datos de forma rápida, pues según sus necesidades, podrían utilizar resultados ya almacenados sin necesidad de esperar a que se realice una evaluación de la calidad que podría llegar a demorarse durante varias horas.

5.3. Trabajo futuro

Aparte de posibles mejoras en la interfaz de usuario y inclusión de nuevas alternativas para la visualización gráfica de resultados, como trabajo futuro se plantea ampliar la aplicación web para dar soporte a medidas que evaluación que requieren la inspección de muestras de registros de metadatos por parte de expertos. La norma de evaluación de la calidad basada en ISO 19157 propone varias de estas métricas que se deben realizar manualmente para algunas dimensiones de la calidad. El soporte futuro de la aplicación sería la posibilidad de seleccionar una muestra de registros aleatoriamente y facilitar que los expertos anoten manualmente si han detectado errores o aciertos en relación a la métrica evaluada para un registro de metadatos o los valores de una propiedad específica de los metadatos.

Otra posibilidad de extensión en el futuro sería la posibilidad de configurar a través de la aplicación web los elementos de metadatos sobre los que se aplican las medidas. Por ejemplo, la metodología basada en la norma ISO 19157 propone varias dimensiones (por ejemplo, la consistencia de dominio) y métricas asociadas que se pueden aplicar sobre varios elementos de metadatos.

Bibliografía

[1] Agenda. *Agenda/agenda: Lightweight job scheduling for node.js*. URL: https://github.com/agenda/agenda.

[2] *Angular*. URL: https://angular.io/.

[3] *Apache Jena*. URL: https://jena.apache.org/index.html.

[4] *Aplicación de Listas de Tareas Pendientes y Administración de Tareas*. URL: https://www.microsoft.com/es-es/microsoft-365/microsoft-to-do-list-app?rtc=1.

[5] Ckan. *Ckan/ckanext-DCAT: CKAN dcat*. URL: https://github.com/ckan/ckanext-dcat#installation.

[6] CKAN Association. *The CKAN website*. 2021. URL: https://ckan.org/.

[7] R. Cyganiak, D. Wood y M. Lanthaler. *RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014*. 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[8] *Data Catalog Vocabulary (DCAT) - version 3*. URL: https://www.w3.org/TR/vocab-dcat-3/.

[9] *Data on the web best practices: Data Quality Vocabulary*. URL: https://www.w3.org/TR/vocab-dqv/.

[10] *Data Partitioning with Chunks*. URL: https://www.mongodb.com/docs/manual/core/sharding-data-partitioning/.

[11] *DCAT-AP*. URL: https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/dcat-application-profile-data-portals-europe/release/201-0.

[12] Extrabacon. *Extrabacon/python-shell: Run python scripts from node.js with simple (but efficient) inter-process communication through Stdio*. URL: https://github.com/extrabacon/python-shell.

[13] *Full table of contents*. URL: https://docs.ckan.org/en/2.9/contents.html.

[14] *Installation and setup*. URL: https://docs.ckan.org/projects/ckanext-spatial/en/latest/install.html.

[15] *Kanban (desarrollo)*. 2022. URL: https://es.wikipedia.org/wiki/Kanban_(desarrollo).

[16] *Let's build from here*. URL: https://github.com/.

[17] *Mean (solution stack)*. 2022. URL: https://en.wikipedia.org/wiki/MEAN_(solution_stack).

[18] *Metadata quality*. URL: https://data.europa.eu/mqa/methodology.

[19] Ministerio de Hacienda y Administraciones Públicas (MINHAP). *ANEXO III. Metadatos de documentos y recursos de información del catálogo. Resolución de 19 de febrero de 2013, de la Secretaría de Estado de Administraciones Públicas, por la que se aprueba la Norma Técnica de Interoperabilidad de Reutilización de recursos de la información. Boletín Oficial del Estado, lunes 4 de marzo de 2013*. 2013. URL: http://www.boe.es/boe/dias/2013/03/04/pdfs/BOE-A-2013-2380.pdf.

[20] *Navigating graphs*. URL: https://rdflib.readthedocs.io/en/stable/intro_to_graphs.html.

[21] Node.js. URL: https://nodejs.org/.

[22] *Node.js web application framework*. URL: https://expressjs.com/.

[23] Javier Nogueras-Iso et al. "Quality of Metadata in Open Data Portals". En: *IEEE Access* 9 (2021), págs. 60364-60382. DOI: 10.1109/ACCESS.2021.3073455.

[24] *Overleaf, online latex editor*. URL: https://www.overleaf.com/.

[25] *Passport.js*. URL: https://www.passportjs.org/.

[26] *Portal oficial de datos europeos*. URL: https://data.europa.eu/es.

[27] *Python 3.7.14 documentation*. URL: https://docs.python.org/3.7/.

[28] *RDF 1.1 turtle*. URL: https://www.w3.org/TR/turtle/.

[29] *stain/jena-fuseki*. URL: https://hub.docker.com/r/stain/jena-fuseki/#!.

[30] *The developer Data Platform*. URL: https://www.mongodb.com/.

[31] Manuel Antonio Ureña-Cámara, Javier Nogueras-Iso y Javier Lacasta. "Revisión de la calidad de los conjuntos de datos abiertos sobre presupuestos". En: *Revista cartográfica* 103 (2021), págs. 145-163.

[32] *Venv - creation of virtual environments*. URL: https://docs.python.org/3.7/library/venv.html.

[33] Mark D. Wilkinson et al. “The FAIR Guiding Principles for scientific data management and stewardship”. En: *Scientific data* 3.1 (2016), págs. 1-9.

A. Requisitos

A.1. Catálogo de requisitos

El catálogo de requisitos se compone de un conjunto de requisitos funcionales (Tabla 1), un conjunto de requisitos no funcionales (Tabla 2) y algunas restricciones (Tabla 3).

Identificador	Descripción
RF-1	El portal permite la evaluación de los metadatos de un portal de datos abiertos.
RF-2	La calidad se podrá evaluar utilizando uno o varios métodos de calidad.
RF-3	La evaluación sobre un punto CKAN se hará mediante la descarga y carga de los datos sobre un punto SPARQL local.
RF-3	Se debe poder evaluar un punto SPARQL accesible de forma pública.
RF-4	Se podrán descargar los datos de un catálogo y cargarlos sobre un punto SPARQL local para ser evaluados sobre el mismo posteriormente.
RF-5	Los administradores pueden programar tareas periódicas de monitorización.
RF-6	Tras finalizar una evaluación, los resultados deben ser accesibles por cualquier usuario.
RF-7	El portal incluirá un cuadro de mandos con los resultados de la evaluación.
RF-8	Los resultados de la evaluación de la calidad se podrán exportar por cualquier usuario.

Tabla 1: Tabla de los requisitos funcionales

Identificador	Descripción
RNF-1	El fichero de exportación de una evaluación será en formato DQV.
RNF-2	Los metadatos que se deben evaluar estarán basados en el vocabulario DCAT sobre RDF.
RNF-3	Los métodos de evaluación de la calidad serán el método basado en ISO 19157 y la metodología MQA (Metadata Quality Assessment) del portal de datos europeo.

Tabla 2: Tabla de los requisitos no funcionales

Identificador	Descripción
R-1	El Backend debería ser capaz de integrar código en Python para la evaluación de la calidad.
R-2	El servidor que proporciona el acceso a un punto SPARQL de forma local debe ser independiente al resto de la aplicación.

Tabla 3: Tabla de restricciones

A.2. Casos de uso

Para poder entender mejor los requisitos se han elaborado distintos diagramas de casos de uso según las distintas acciones que un usuario puede realizar en la aplicación web desarrollada.

Así pues se describe el caso de uso de las funciones completas que puede realizar la aplicación web en la Figura 2.

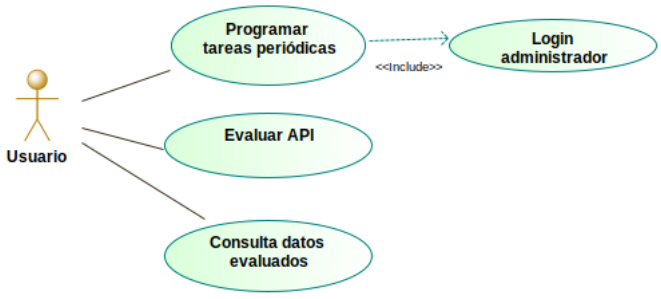

Figura 2: Diagrama de caso de uso de la aplicación general

En la Figura 3 se muestra las acciones que implican la evaluación de un catálogo de datos.

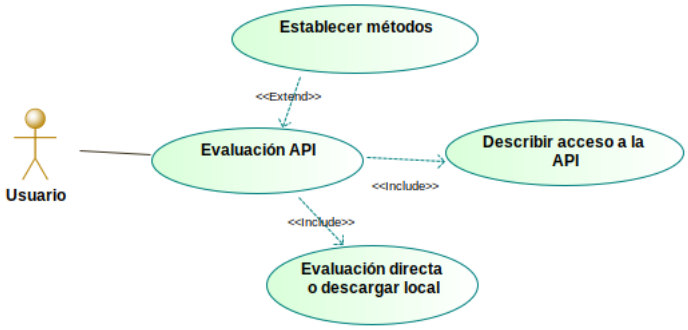

Figura 3: Diagrama de caso de uso de una evaluación

En la Figura 4 se muestra las acciones que puede realizar un administrador para programar una tarea de evaluación.

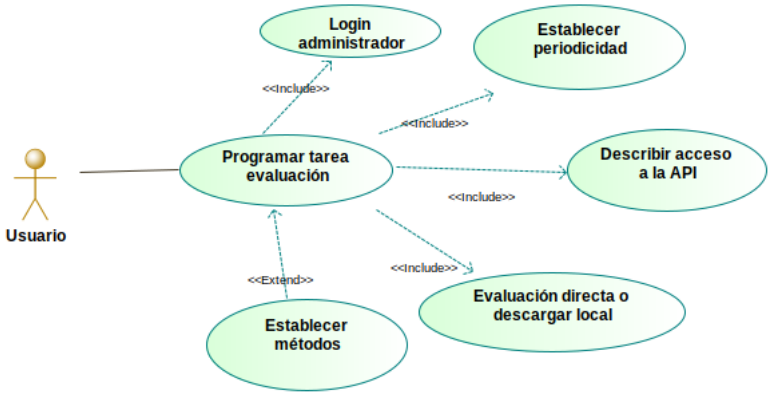

Figura 4: Diagrama de caso de uso de programación de tareas

En la Figura 5 se muestra las acciones que se pueden realizar para la consulta de resultados de una evaluación.

Figura 5: Diagrama de caso de uso de consulta de resultados

B. Resultados de evaluaciones

En las siguiente figuras se muestran los resultados obtenidos de distintas evaluaciones que se detallan en el apartado de configuraciones probadas (ver sección 4.3).

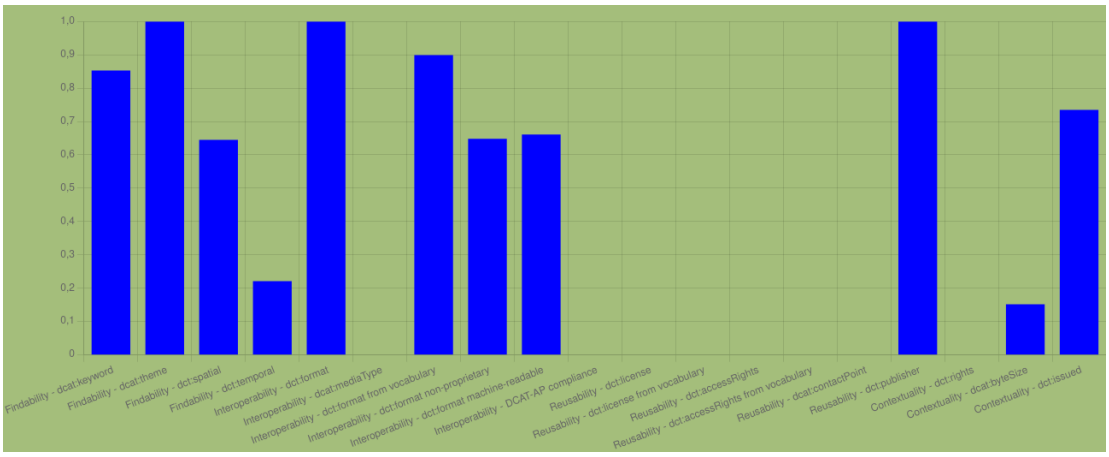

Figura 6: Evaluación con el modelo de calidad MQA sobre datos de https://datos.gob.es/es/catalogo

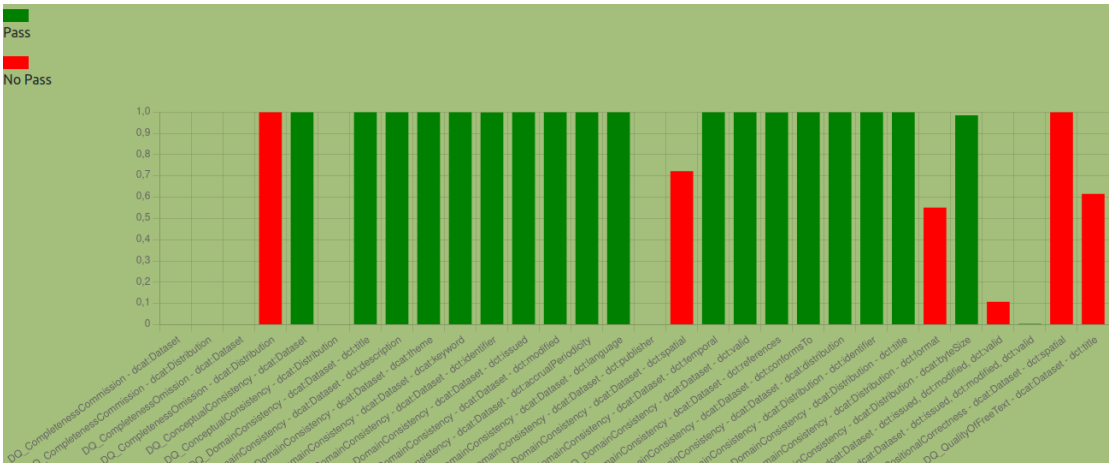

Figura 7: Evaluación con el modelo de calidad ISO 19157 sobre datos de https://datos.gob.es/es/catalogo

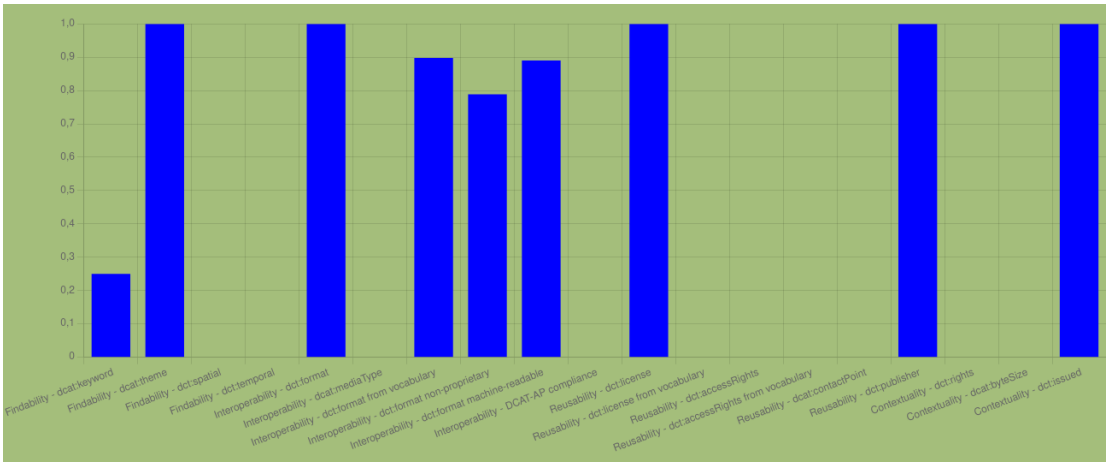

Figura 8: Evaluación con el modelo de calidad MQA sobre datos de https://datos.santiagodecompostela.gal/catalogo

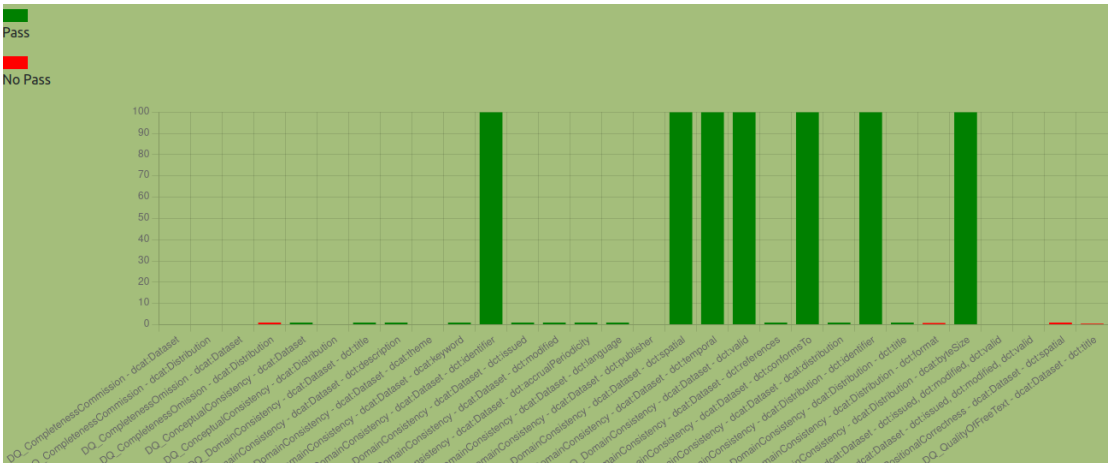

Figura 9: Evaluación con el modelo de calidad ISO 19157 sobre datos de https://datos.santiagodecompostela.gal/catalogo