VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Incorporación de la tecnología QR en el almacenamiento y visualización de metadatos geográficos en el marco de una IDE Corporativa

Ing. German Giovanni Vargas Velásquez

Universidad de Salamanca

Escuela Politécnica Superior de Ávila

Departamento de Ingeniería Cartográfica y del Terreno

Ávila, España

2022

Incorporación de la tecnología QR en el almacenamiento y visualización de metadatos geográficos en el marco de una IDE Corporativa

Ing. German Giovanni Vargas Velásquez

TFM presentado como requisito final para optar al título de:

Máster Universitario en Geotecnologías Cartográficas en Ingeniería y Arquitectura

Directores:

PhD. Ángel Luis Muñoz Nieto

PhD. José Antonio Martín Jiménez

Universidad de Salamanca

Escuela Politécnica Superior de Ávila

Departamento de Ingeniería Cartográfica y del Terreno

Ávila, España

2022

A mis seres queridos, por haberme educado en sus valores y con su ejemplo de vida para ser lo que soy y llegar hasta donde he llegado. Por ser la inspiración de mis actos y el tesoro más preciado de mi existencia.

Agradecimientos

Agradezco en primera instancia a la Universidad de Salamanca por la oportunidad de permitirme acceder a esta prestigiosa y emblemática casa de estudios, en cuyo seno científico, adquirí un amplio espectro de saberes y habilidades que coadyuvaron al crecimiento y maduración de mi perfil académico y profesional.

Así mismo, agradezco profundamente a mi asesor, el docente del máster Ángel Luis Muñoz Nieto por su orientación e invaluables aportes conceptuales y metodológicos que permitieron la conclusión satisfactoria de éste.

Finalmente, este trabajo no hubiese sido posible sin la colaboración desinteresada de algunos de mis colegas quienes aportaron soluciones a los constantes retos que implicó el desarrollo de este trabajo. A todos mil gracias.

Contenido

Resumen

La gestión de metadatos geográficos es un conjunto de procesos por los cuales se controla el ciclo de vida de estos datos desde su creación hasta su disposición final. El objetivo consiste en exponer información descriptiva sobre el contexto, la calidad, la condición, entre otras características, para identificar completamente un recurso de información geográfica.

Su utilidad en el mundo digital adquiere absoluta relevancia por cuanto permiten mejorar la estructuración de tales recursos (describen y catalogan), así como su recuperación (identificar la relevancia del recurso buscado). Esto se traduce en eficiencias operativas y soporte a la toma de decisiones de negocio, suscitando ventajas competitivas para una organización.

En este sentido, este trabajo pretende hacer una aproximación conceptual sobre la gestión de metadatos geográficos como epicentro en procesos de gestión de información espacial y gobierno de datos al interior de una infraestructura de datos espaciales (IDE) corporativa. Así mismo, proponer un modelo de gestión incorporando la tecnología QR y exponer su implementación mediante una solución informática construida sobre software libre.

Aunque los metadatos incluyen información valiosa y decisiva al momento de utilizar adecuadamente un recurso, en la práctica ésta no es del todo accesible para el usuario final, impactando negativamente sus flujos de trabajo. Por esta razón, una iniciativa de este tipo no sólo facilitará acceder a los metadatos de un mapa sino además la experiencia de usuario en cuanto a su manipulación, lectura e interpretación. En consecuencia, se habrá mejorado la productividad de las operaciones en donde el mapa que describen es empleado como un recurso de información.

Palabras claves: *gestión, metadatos geográficos, códigos QR, mapas digitales, IDE corporativa*

1. Introducción

Ciertamente la información geográfica es cada vez más importante en las sociedades del mundo moderno. Tanto las organizaciones como los proveedores de tecnología han advertido el potencial que representa una interrelación entre los procesos operativos y activos de información, de forma transversal e intersectorial, siendo ésta la principal motivación de los nuevos diseños arquitectónicos y estratégicos de gestión.

Bajo este contexto, estas estructuras corporativas han venido valorando progresivamente dichos activos y especialmente aquellos que poseen geolocalización. En efecto, se constituyen un recurso esencial en la toma de decisiones inteligentes basadas en criterios de tipo geográfico. Dada esta propiedad, pueden ser socializados fácilmente mediante el uso de mapas, entendiendo éste como un producto informativo, particularmente útil para mejorar la comprensión de los resultados de negocio.

En consecuencia, durante los últimos años la industria geoespacial ha experimentado un importante dinamismo y, por tanto, una proliferación en la creación y publicación de datos espaciales, servicios interoperables de información geográfica, productos cartográficos digitales y aplicaciones que los consumen, ofreciendo valor agregado a los usuarios que demandan este tipo de recursos.

En esa misma dirección, actualmente se lideran procesos de cambio e innovación que, facilitando la transformación de instituciones públicas y privadas en este mundo digitalizado, se explora constantemente múltiples alternativas para la creación, administración y diseminación optimizada de estos recursos.

Dichos procesos responden, por un lado, a mejorar el desempeño de las operaciones diarias que recurren a su consulta para el cumplimiento de objetivos de negocio; y por otro, garantizar su disponibilidad oportuna para consolidar un crecimiento estratégico y sostenible de la organización, que le asegure continuidad en el mercado cualquiera sea su competencia misional.

Tal escenario ha motivado que hoy cada vez más profesionales, adscritos a un amplio rango de disciplinas, estén interesados en producir, modificar y utilizar productos de información geográfica.

Conforme crece el número, complejidad y diversidad de estos recursos, también estriba la necesidad de disponer de un mecanismo que permita su catalogación, organización y gestión a partir de unas descripciones claves, las cuales han de brindar un contexto para el uso adecuado y eficiente de éstos. Esto supone que, para asegurar su correcta utilización, las presunciones y limitaciones que han afectado su creación deben ser documentadas.

En este orden de ideas, según Caplan (1995) los metadatos se construyen a la luz de ofrecer información sobre el contenido, licenciamiento, condiciones de aprovechamiento y control de un recurso. De este modo, se alcanzan objetivos como su descripción, identificación, recuperación, evaluación y preservación.

Por esta razón, los metadatos permiten a un productor describir completamente sus recursos, de manera que sus potenciales usuarios puedan localizarlos, evaluar su adecuación para un determinado propósito, y conocer las presunciones y condiciones de uso.

De hecho, ofrecer la posibilidad que un usuario acceda a este tipo de descripciones estructuradas sobre los productos de información geográfica que requiere, en general se concede entre otras ventajas, según lo expresa Bernabé & López (2012), organizar y mantener la inversión que implicó su adquisición. Esto supone conocer qué recursos existen, dónde se encuentran disponibles y bajo qué condiciones. Luego, la incorporación de metadatos fomenta la reusabilidad y mantenimiento de los recursos sin tener que recurrir constantemente a sus creadores originales.

Así mismo, la existencia de metadatos geográficos ofrece, a aquellos que no están familiarizados con el producto informativo, su entendimiento. De esta manera, se posibilita su aprovechamiento óptimo, lo cual repercute en eficiencias operativas al interior de los procesos de negocio de una organización.

A pesar del reconocimiento general de la importancia que goza los metadatos al interior de un modelo de gestión de información, según Wayne (2005), su generación dista de ser una tarea sencilla y atractiva.

Diferentes obstáculos se interponen en la gestión de metadatos geográficos. Una de las principales limitaciones obedece precisamente a problemas prácticos asociados a su accesibilidad y lectura semántica (visto desde la óptica de usuario) y la complejidad

conceptual que subyace propiamente a su creación (visto desde la óptica del productor), especialmente en la descripción de mapas, limitando su efectividad en tareas como el descubrimiento o valoración de tales documentos.

De modo semejante, se atribuye que la extensión y complejidad de los estándares existentes, en este universo de discurso, otorga cierto grado de incomprensión y dificultad en su implementación (Bodoff, 2006). De hecho, Anaya et al., (2002) afirma que todavía no existe un consenso sobre cuáles son los metadatos más apropiados para describir, por ejemplo, mapas digitales.

En la práctica, Beltrán Fonollosa (2013) sostiene que la creación de metadatos geográficos ha ocupado en general un papel secundario al interior de las organizaciones, especialmente del sector privado. Esta situación es verificable con facilidad si se considera que, dichos metadatos son opcionalmente construidos con un tiempo posterior a la producción de los recursos. Por ende, es percibido con frecuencia como un coste adicional en tanto su elaboración estandarizada demanda tiempo y personal cualificado (Najjar et al., 2004), relegándose en tareas de gestión por sus pocos beneficios tangibles e incentivos para su generación.

Más aún, en la actualidad se exige la creación de estos metadatos siguiendo un variado corpus de estándares, perfiles y directrices de diverso alcance, con el consiguiente problema de interpretación y acople para luego compartir y utilizar esta información. A medida que el número, tamaño y nivel de detalle de los estándares de metadatos sea mayor, la tarea de crearlos y mantenerlos bajo estos lineamientos resulta ser más larga, difícil y tediosa.

No obstante, es importante señalar que, si bien a corto plazo la documentación de productos cartográficos puede parecer una tarea costosa y dispendiosa para una organización, cuya misión no es la de producir cartografía oficial. A largo plazo las ventajas superan estos dilemas, dado que el coste inicial de documentar este tipo de recursos es muy inferior al eventual coste de adquirir o generar nuevamente geodatos y sus productos derivados, previniendo un incremento de ficheros duplicados y redundantes.

Por consiguiente, la gestión de metadatos se advierte como una actividad que contribuye al conocimiento de los recursos existentes, evita la duplicación de esfuerzos en los procesos de producción, potencia su explotación y reduce el riesgo de que los datos

espaciales, servicios interoperables de información geográfica y productos informativos se desconozcan por falta de mecanismos para su administración (Bernabé-Poveda & López-Vázquez, 2012).

En virtud de lo anterior, este trabajo busca mejorar el proceso de creación, administración y consumo de metadatos geográficos al interior de una IDE de nivel corporativo. Mediante una solución informática más flexible y eficiente, se aspira soportar la descripción y evaluación de documentos cartográficos a través del uso de códigos de respuesta rápida QR.

Se pretende entonces, a partir de la incorporación de esta tecnología, almacenar información descriptiva de un mapa en una matriz de puntos bidimensional. Esta matriz es leída fácilmente desde un dispositivo móvil, ofreciendo así una experiencia de consumo de información útil y precisa al usuario, de manera más fácil e intuitiva.

Así, se procede a simplificar las búsquedas por consulta parametrizada requeridas en catálogos y herramientas de gestión de metadatos, implementados en nodos IDE de nivel superior, pues basta con escanear el código para acceder al contenido informativo de un documento cartográfico.

En la última década, el uso de códigos QR ha crecido exponencialmente en una amplia gama de aplicaciones. Sin embargo, en el campo de la Geoinformática no existe una implementación rigurosa, pese a ofrecer múltiples beneficios, especialmente en la gestión de metadatos geográficos. Su implementación resulta bastante prometedora en estos contextos al mejorar la productividad y optimizar tareas de gestión que demandan normalmente la documentación de este tipo de recursos.

Teniendo en cuenta que, actualmente, productores y usuarios disponen de un dispositivo móvil con capacidades de escaneo para este tipo de códigos, y si a esta práctica se incluye la inserción de éstos como un elemento que complemente la información marginal de mapas digitales, indudablemente permitirá que la experiencia de acceso y lectura de metadatos geográficos se vuelva mucho más ágil, sencilla y amigable para todos los miembros que hacen uso de este tipo de recursos en el cumplimiento de sus actividades institucionales.

Por lo anterior, este TFM se estructura en 9 capítulos. El primero corresponde a la exposición de aspectos y consideraciones que motivaron el planteamiento del proyecto.

El capítulo 2 formula los objetivos del TFM, los cuales exponen las metas concretas que se espera alcanzar para la consecución del diseño e implementación del modelo de gestión de metadatos geográficos incorporando la tecnología QR.

El capítulo 3 plantea la hoja de ruta o el conjunto de actividades y su secuencia lógica para alcanzar los objetivos propuestos en el capítulo anterior.

El capítulo 4 corresponde al primer resultado del TFM. Este contiene una síntesis conceptual sobre los procesos de gestión de información espacial, los componentes y arquitectura de una IDE en un entorno institucional y las generalidades de los metadatos geográficos y la metodología para su creación.

El capítulo 5 describe el estudio de los estándares de metadatos geográficos y una presentación sucinta de las herramientas más populares de gestión.

El capítulo 6 contiene el segundo resultado del TFM. Este capítulo detalla la forma cómo se organizan y combinan los elementos requeridos para implantar el modelo de gestión de metadatos geográficos incorporando la tecnología QR a partir del análisis de requerimientos y la especificación de la arquitectura del sistema.

El capítulo 7 presenta el tercer resultado del TFM. En este se describe el proceso de desarrollo del sistema de gestión de metadatos geográficos, articulado fundamentalmente en dos aplicaciones. Una aplicación de escritorio construida sobre el lenguaje de programación Python encargada de las operaciones de gestión. Y una aplicación móvil para la lectura y visualización del metadato, construida sobre el lenguaje Java para el sistema operativo Android.

El capítulo 8 expone la implementación y acople de los ejecutables del sistema y un proceso de validación y pruebas de funcionamiento. En este se presenta un ejemplo de creación, modificación y eliminación de un metadato desde la aplicación de escritorio y su despliegue en la aplicación móvil.

El capítulo 9 finaliza el TFM con las conclusiones, aportaciones y trabajos futuros sugeridos para el proyecto.

2. Objetivos

2.1. Objetivo general

Definir una propuesta metodológica para la incorporación de la tecnología QR en el almacenamiento y visualización de metadatos geográficos asociados a mapas digitales en el marco de una infraestructura de datos espaciales corporativa.

2.2. Objetivos específicos

- Diagnosticar y conceptualizar la gestión de metadatos geográficos en el contexto de una IDE corporativa.

- Definir los elementos determinantes para el diseño del modelo de gestión de metadatos geográficos basado en tecnología QR.

- Construir un prototipo que permita el acceso, almacenamiento y visualización de metadatos geográficos mediante el uso de códigos QR.

- Validar el funcionamiento del prototipo mediante la generación de un código QR asociado a un metadato geográfico y su despliegue en un dispositivo móvil.

3. Metodología

La metodología de este proyecto se concibe en 4 fases (ver figura 1). Una primera de conceptualización y diagnóstico. Posteriormente una fase de diseño, que incorpora actividades orientadas a obtener el modelamiento del sistema para la gestión de metadatos geográficos incorporando la tecnología QR, el modelo de persistencia y el diseño de las interfaces gráficas de usuario.

Una fase de desarrollo que pretende la codificación del prototipo funcional que satisfaga los requerimientos. En este apartado se propone la utilización de la metodología ágil de desarrollo de software SCRUM.

Finalmente, una fase de implementación y pruebas donde se pretende validar, por un lado, el acople de los componentes de la solución informática, y por otro, el rendimiento de la aplicación y la coherencia de sus resultados, mediante la creación de un registro de metadato para un mapa digital y su posterior lectura y despliegue en pantalla mediante un dispositivo móvil.

En seguida, se presenta una descripción de cada una de las fases consideradas para el desarrollo del TFM.

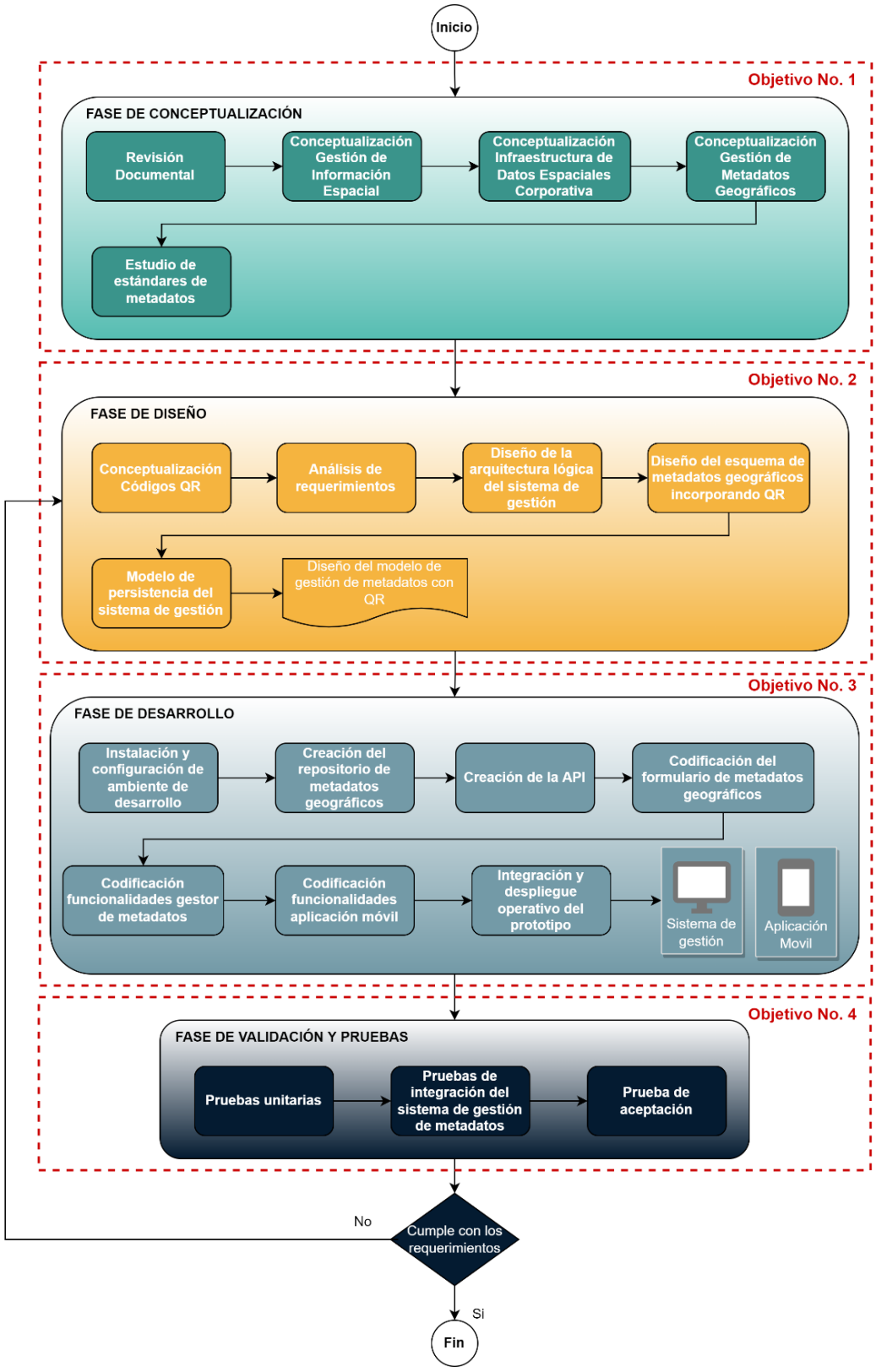


***Figura 1**. Esquema metodológico del TFM. Fuente: Elaboración propia.*

3.1. Fase de conceptualización

En esta fase se pretende hacer una revisión documental que permita construir un estado del arte sobre el universo de discurso del TFM. Las principales tareas son la recolección, lectura y síntesis de información técnica obtenida de artículos de revistas, ponencias, capítulos de memoria, libros, disertaciones académicas, normatividad y otra información relevante obtenida en diversas bases de datos científicas.

La conceptualización se efectúo sobre tres temas claves: la gestión de información espacial como marco referencial, las infraestructuras de datos espaciales corporativas vistas como un instrumento para la implementación exitosa de un modelo de gestión, y la gestión de metadatos propiamente dicha, como epicentro de los procesos informacionales al interior de una organización.

Adicionalmente, desde el enfoque geomático, se presenta una discusión que se vertebra sobre dos ejes, por un lado, una descripción de la estandarización de metadatos geográficos y sus actualizaciones, y por otro, la indagación sobre las herramientas de edición y publicación disponibles en el mercado para adelantar esta tarea.

3.2. Fase de diseño

En esta fase se propone una línea base del proyecto que inicia por el levantamiento de requerimientos para la definición del alcance. Acto seguido, se bosqueja el modelo conceptual de metadatos para la documentación de mapas digitales a partir del núcleo del estándar ISO19115-1.

La fase finaliza con el diseño arquitectónico de la solución informática basada en tecnología QR mediante el modelado comportamental y estructural empleando el lenguaje de especificación UML.

Esta actividad de modelado comprende la identificación de la estructura principal del sistema, sus componentes, sus relaciones y cómo se distribuirán desde el punto de vista de la ejecución en el hardware.

Así mismo, se contemplan tareas orientadas al diseño de las interfaces gráficas de usuario, mediante la técnica de prototipado de maqueta digital o muckup (sobre las

herramientas de desarrollo), atendiendo estándares y estrategias de diseño (estilos y metáforas) para su construcción.

3.3. Fase de desarrollo

En línea con las actividades fundamentales del ciclo de vida del software se propone un conjunto de tareas necesarias para la construcción de la solución informática que permita la gestión de metadatos geográficos a nivel corporativo.

En este sentido, se plantea en primera instancia la instalación de herramientas de software requeridas para la implementación (Motor de base de datos PostgreSQL, IDE Spyder, Qt Designer, Android Studio) y configuración del entorno de desarrollo para atender este fin.

Posteriormente, se procede a la creación de la base de datos relacional que almacenará los registros de metadatos de los productos informativos (mapas digitales). En seguida se procede con la codificación (utilizando los lenguajes de programación Python y Java) de las funcionalidades para la herramienta de gestión de metadatos y la aplicación móvil, de conformidad a los casos de uso identificados en la fase anterior.

La fase finaliza con el alojamiento y despliegue operativo del prototipo funcional (programas ejecutables) para su validación y pruebas.

Es conveniente mencionar que el enfoque de desarrollo de software más idóneo según las necesidades y tiempos de ejecución del proyecto corresponde a la metodología Scrum. Esta metodología se caracteriza por brindar un marco de trabajo ágil en ambientes de requerimientos imprecisos o cambiantes con foco en el desarrollo iterativo.

En esencia, Scrum define tres roles: el scrum máster o líder del proyecto, es quien coordina el proyecto y la implementación de la metodología. El dueño del producto quien representa a los stakeholders y es el responsable de maximizar el valor del producto de software; tiene entre sus funciones gestionar la lista ordenada de tareas requeridas o Product Backlog (unidades básicas de trabajo).

Por último, el rol de desarrollador, quien tiene la responsabilidad de convertir los requerimientos funcionales del Product Backlog en iteraciones funcionales del producto de

software a implementar en la organización. Para el desarrollo de este proyecto, los tres roles son asumidos por el autor de este documento.

Adicionalmente, Scrum define un evento principal o Sprint que corresponde a una ventana de tiempo donde se crea el prototipo funcional a partir de un trabajo iterativo con incrementos hechos en la medida en que se van atendiendo los requerimientos funcionales del product backlog.

Un Sprint a su vez se compone de los siguientes elementos (ver figura 2): Planeación del sprint (se define un plan de trabajo, qué se va a entregar y cómo se logrará). Daily scrum (reunión diaria breve del equipo de desarrollo para revisar avances y los obstáculos que se han presentado). El trabajo de desarrollo. La revisión del sprint (se muestra el producto y su funcionamiento). Y la retrospectiva del sprint (se analiza cómo fue la comunicación, el proceso y las herramientas; qué estuvo bien, qué no, y se crea un plan de mejoras para el siguiente sprint).

Es importante mencionar que para el desarrollo de este TFM no serán considerados el daily scrum y la retrospectiva del sprint, ya que no existe un equipo de desarrollo y no se pretende hacer escalable el prototipo funcional con nuevas funcionalidades, para acudir a otro Sprint.

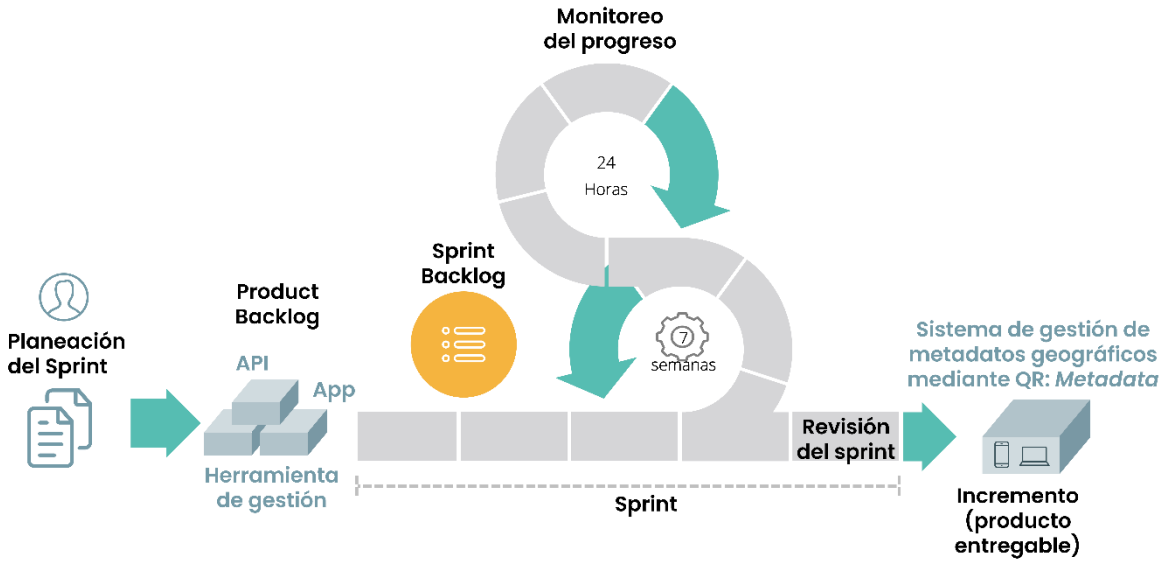


***Figura 2**. Metodología scrum para el desarrollo del proyecto. Fuente: Elaboración propia.*

En la metodología existen también los artefactos. Estos son subproductos de las actividades del marco de trabajo (ver figura 2). Entiéndase por artefactos los siguientes: el

Product Backlog, en donde se han establecido 3 tareas o producto backlog para el proyecto: La API, la herramienta de escritorio para gestión de metadatos geográficos y la aplicación móvil para su consumo.

El Sprint Backlog, el cual es construido con los requerimientos del product backlog, es decir, cada funcionalidad que se codificará mediante un lenguaje de programación o cada customización que se hará de interfaz gráfica, documentación, etc., y que constituyen los incrementos del producto.

El Monitoreo de Progreso para controlar cómo se avanza frente al cronograma de tareas. El Incremento o producto entregable que corresponde a la suma de todos los ítems terminados del sprint backlog.

La iteración o Sprint tiene una duración de 7 semanas, en donde se entregan productos parciales de cada tarea, que son perfectamente funcionales y articulables en el desarrollo de todo el sistema gestor de metadatos geográficos y su despliegue operativo.

3.4. Fase de validación y pruebas

Finalmente se hace entrega del prototipo funcional del sistema para la gestión de metadatos geográficos incorporando tecnología QR, en cumplimiento de los lineamientos planteados en las fases anteriores.

De esta manera, se confronta los requerimientos con la solución informática en un proceso iterativo y se somete a un conjunto de pruebas los componentes del sistema (pruebas unitarias), el sistema en general y su integración para depurar y ajustar errores.

Se concluye con las pruebas de aceptación en virtud del objetivo general de este trabajo. Estas actividades pretenden mostrar que el sistema informático diseñado para la gestión de metadatos geográficos es conforme a la especificación y da alcance a los requerimientos de este proyecto.

4. Diagnóstico y conceptualización

4.1. Gestión de la información espacial

En términos generales, la gestión de información es un concepto reciente en el campo de las ciencias de la información y las comunicaciones. Se orienta a la generación de estrategias de negocio apoyadas en tecnología y activos de información, para encontrar soluciones que generan alto valor en las organizaciones.

De acuerdo con Dante (2011), desde la década de los 80, ha ganado un espacio importante en las instituciones en general, y en particular, en aquellas que han adoptado como misión el desarrollo de servicios y productos de información.

Algunos autores como Salas (2002) citando a Woodman (1985) coinciden en que la gestión de información, bajo este marco de trabajo, se refiere a la obtención de la información adecuada, para la persona indicada, a su costo esperado, en el tiempo y lugar oportuno, para tomar la decisión correcta.

En este sentido, se trataría entonces de un proceso articulado, que pretende maximizar el valor y los beneficios derivados del uso de la información, minimizar el coste de adquisición, procesamiento y disposición, determinar responsabilidades para su uso eficiente y efectivo, y asegurar un suministro o flujo de información constante. Esto supone que su explotación deberá servir a una amplia variedad de objetivos en ambientes de trabajo endógenos o exógenos de una corporación.

Por otro lado, para Ponjuán (2004) cuando se hace alusión a la gestión de información, esta se refiere a la gestión que se lleva a cabo en un sistema de información, considerando que este provee productos informacionales.

Así pues, consiste en un proceso mediante el cual se obtienen, despliegan o utilizan recursos humanos y físicos para manejar información. Esto precisa un conjunto de procesos por los cuales se controla el ciclo de vida de los datos, desde su obtención (por creación o captura), hasta su disposición final (archivo o eliminación).

La finalidad de la gestión de la información, según Morales Flores (2004), es ofrecer mecanismos que permitan a la organización adquirir, producir y transmitir, al menor coste

posible, datos e informaciones con una calidad, exactitud y actualidad suficientes para servir a sus objetivos de negocio de manera competitiva.

Los elementos involucrados con la gestión de información se pueden resumir en tres aspectos: (a) Los que competen a la información como fuente/recurso, es decir, asociada a procesos productivos al interior de las organizaciones; (b) los relacionados con el usuario de productos y servicios de información; (c) los que conforman el canal de comunicación y acceso entre el usuario y la fuente (Salas, 2002).

Un aspecto interesante es la connotación que adquiere la información bajo este contexto, ya que puede ser considerada: Como un recurso, al tener un coste y por tanto debe ser posible recuperar un beneficio o rendimiento. Como un producto, y por consiguiente deberá cumplir unas exigencias de consistencia y calidad para la generación de valor. Como un activo, lo cual implica que la institución se ocupe insistentemente por una gestión coherente y articulada con el direccionamiento estratégico y las metas que persigue la corporación.

De esta manera, la gestión de información está estrechamente relacionada con la generación y aplicación de estrategias, políticas, procedimientos y en general una cultura organizacional que promueve la extracción, combinación, depuración y distribución de información garantizando integridad, disponibilidad y confidencialidad a los miembros de la misma, siempre que la requieran. En cuanto al paradigma geoespacial, su conceptualización puede ser perfectamente extrapolada.

En este orden de ideas, la gestión de información espacial comprende la planificación, organización, dirección y control de recursos, sistemas y acciones asociadas con la información que posee una referencia geográfica.

Figura 3. *Cadena de valor de la información geográfica en una organización. Fuente: Elaboración propia.*

A través de su gestión es posible minimizar costos y maximizar beneficios derivados de su uso y tratamiento, al alinear la estrategia corporativa, las geotecnologías y los datos espaciales; al tiempo que se asignan responsabilidades informativas para asegurar una circulación constante de productos informativos. (Rodríguez Cruz, 2008)

Se entiende este tipo de productos, bajo este contexto, como el conjunto de datos transformados en información útil para responder a los objetivos de negocio, generalmente en la forma visual de mapas (Tomlinson, 2008).

A tenor de esta definición, uno de los propósitos que es posible inferir de la gestión de información espacial es precisamente el de plantear e implementar estrategias que coadyuven a mejorar y fortalecer los procesos que se desarrollan en las fases de planificación, producción, conservación y distribución (Bernabé & López, 2012) de recursos de información geográfica.

Con ello, se contribuye al cumplimiento de la misión institucional y a atender adecuadamente las demandas informacionales de los usuarios, como se ilustra a continuación:

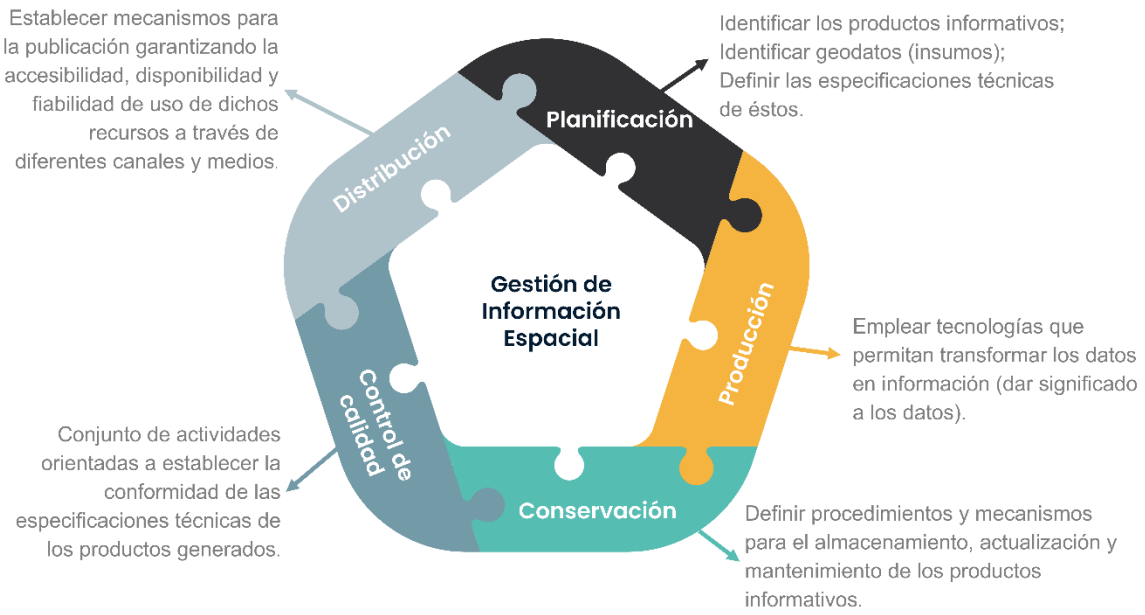


Figura 4. *Esquema de la Gestión de Información Espacial a partir del ciclo de vida de la información. Fuente: Elaboración propia.*

4.1.1. Objetivos de la gestión de información espacial

Considerando los procesos expuestos en la figura 4, los objetivos para una adecuada gestión se enmarcan en la necesidad de proveer instrumentos que mejoren el ciclo de vida de los recursos de información espacial mediante un conjunto de estrategias, de conformidad a unas políticas y estándares institucionales o del orden local-regional-nacional. Algunos de ellos son:

- Propender por establecer un marco de coordinación y cooperación entre productores y usuarios de estos recursos.
- Dar cumplimiento a los lineamientos vigentes derivados de procesos de normalización y estandarización respecto a datos, formatos y tecnologías, que facilite la comunicación entre los involucrados (stakeholders) mediante un lenguaje común (criterios comunes) para producir, compartir y hacer uso de los recursos informacionales.
- Identificación de los requerimientos y expectativas de los usuarios de información geográfica e integración oportuna de ésta en correspondencia a las funciones y actividades de una organización.

- Establecer mecanismos de colaboración de distintos participantes y roles en los procesos de intercambio e interoperabilidad de los recursos.
- Documentar el ciclo de vida de los productos informativos, garantizando la calidad, integridad y disponibilidad de éstos, y asegurando su dimensión efectiva.
- Implementar acciones de evaluación con el fin de que los procesos tengan resultados de costo-beneficio positivos, sean mejor controlados y respondan a objetivos específicos, para incrementar su productividad o el rendimiento de la inversión.

4.1.2. Principios de la gestión de información espacial

Se atribuye como principios de la gestión de información espacial, los siguientes:

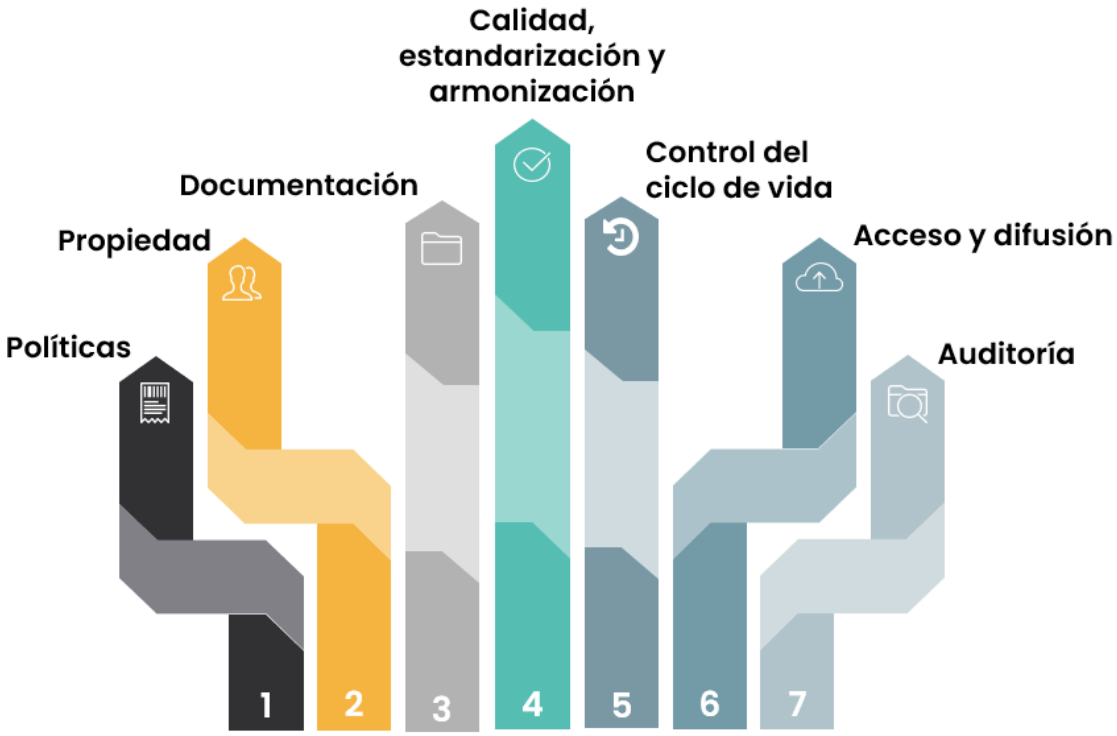


Figura 5. *Principios de la gestión de información espacial. Fuente: Elaboración propia.*

Tabla 1. *Descripción de los principios de gestión*

Principios de la gestión de información espacial	
Principio	**Descripción**
Políticas	Conjunto de principios de alto nivel y de amplia cobertura que forman el marco de acción en el cual opera la gestión de información. Buscan orientar el comportamiento de actores individuales y colectivos para el mejoramiento de la gestión de información espacial. Se requiere la aprobación y respaldo de alto nivel para formalizar la política de información en la organización.
Propiedad	Consiste en la identificación clara de los derechos de propiedad intelectual sobre la información geográfica. Implica el derecho a explotar los datos y aplica tanto a un ítem de información, como a conjuntos de datos combinados y de valor agregado. Es importante que se produzcan acciones para establecer y documentar: los derechos de propiedad intelectual que deben salvaguardarse; las políticas de seguridad, publicación, precios y distribución; acuerdos firmados con los usuarios sobre las condiciones de uso, antes de su publicación.
Documentación	Todos los conjuntos de datos y productos informativos derivados deben documentarse para facilitar su uso adecuado y evitar la duplicación de esfuerzos y recursos en la producción. Con el fin de suministrar un inventario preciso de los datos de la corporación, implementar un catálogo de metadatos que permita a los usuarios conocer el contenido, la calidad, la precisión, la actualidad y el punto de contacto para su adquisición.
Calidad y estandarización	Conjuntos de datos y recursos informacionales conformes a las necesidades actuales y a su futura explotación. La capacidad de integrar información facilita la generación de valor agregado, promueve el uso de los datos y la recuperación de los costos de producción. Para maximizar el potencial de los conjuntos de datos y productos informativos se debe: - Usar definiciones y formatos estándares (facilita la comprensión de los datos para garantizar la integración e interoperabilidad). - Definir criterios de calidad (describen su capacidad para satisfacer necesidades establecidas e implícitas) y aplicar procesos de validación adecuados. - Promover el uso de estándares nacionales e internacionales.

Control del ciclo de vida	El ciclo de vida completo de los datos se debe manejar cuidadosamente, teniendo en cuenta: -La justificación de aspectos tales como: por qué producir nuevos datos o productos informativos, en cambio de mantener los existentes, cómo se puede dar el máximo uso a éstos, por qué los costos de manipulación, almacenamiento y mantenimiento de éstos son aceptables y recuperables. La especificación y modelamiento de los datos, el procesamiento, el mantenimiento la seguridad de los datos, para asegurar cumplirán con su propósito. La auditoría de los datos para monitorear y evaluar su uso y efectividad. Las copias de seguridad.
Acceso y difusión	Pese a que estas actividades dependen de las políticas de la organización, se debe tener en cuenta que el acceso a los datos y productos informativos debe suministrarse a los usuarios de conformidad con las estrategias y políticas de la organización y los derechos de propiedad intelectual sobre los mismos.
Auditoría	Evaluar el alcance de la gestión y los procesos de implantación, dando especial relevancia a los siguientes objetivos: -Determinar el grado de conformidad de los procesos de producción, almacenamiento y distribución de los datos, con las políticas y lineamientos institucionales. -Revisar el alcance de los procedimientos en el mejoramiento de la calidad y el acceso a recursos informacionales. -Verificar los acuerdos establecidos con los usuarios para asegurar que las condiciones de uso de los datos cumplen con las políticas institucionales.

Fuente: Elaboración propia, adaptado de IDESC, 2010[1].

4.1.3. Beneficios de la gestión de información espacial

Hoy, uno de los principales beneficios obtenidos de la gestión de la información espacial se ha materializado en la mejora de los procesos involucrados en la producción recursos de información geográfica, así como el intercambio de éstos en diferentes canales de comunicación, especialmente a través de la web. Dichos beneficios se sintetizan en el siguiente esquema:

[1] Recuperado de https://idesc.cali.gov.co/download/capacitacion_geoservicios_idesc/taller_nivelacion_ide/04_gestion_de_informacion_geografica.pdf

***Figura 6**. Beneficios de la gestión de información espacial. Fuente: Elaboración propia.*

La siguiente tabla presenta comparativamente algunas de las consideraciones a la hora de implementar un modelo de gestión bajo este contexto en el ámbito organizacional.

***Tabla 2.** Consideraciones asociadas a la gestión de información espacial*

Antes de implementar	Después de implementar
Cada dependencia produce la información geográfica que requiera para sus análisis	Productores de información identificados y responsables de sus datos misionales
Datos escasos, de difícil localización, acceso y utilización	Posibilidad de integrar la información, es decir, existe una estructura común de tal manera que se pueden incorporar datos de diferentes productores bajo un único esquema de almacenamiento
Dificultad para encontrar los datos y productos informativos requeridos	Información sobre disponibilidad de los datos debidamente publicada y puesta a disposición de los usuarios
Demora en la consecución de información sobre los datos y su calidad	Información sobre los datos disponibles como: precisión, grado de actualización, entre otros.
Conocimiento y aprendizaje reservado a nivel de una dependencia o individual	Facilidad en la trasferencia del conocimiento al interior de la organización
Recursos limitados tanto económicos, tecnológicos y humanos	Optimización de recursos a todo nivel

Duplicidad de información por necesidad de satisfacer sus objetivos	Definición una base de información de uso común (datos fundamentales), que permita la generación de productos o servicios con propósitos específicos. Producción de recursos de información según misión de la organización.
Procedimientos complejos al momento trabajar con información proveniente de otras fuentes	Facilidad en el uso de información proveniente de otras fuentes, posibilidad de interoperar información y servicios
Definición de calidad de información de acuerdo a necesidades propias o según requerimientos sin estándares aplicables a nivel general	Existencia de estándares de información geográfica aplicados en la producción de información
Software disponible complejo, potente pero poco estandarizado	Posibilidad de articular herramientas de software comercial y libre, tecnología abierta
Esfuerzos aislados de integración de información	Respaldo legal y normativo para la integración e interoperabilidad

Fuente: Elaboración propia, adaptado de (IDECA, 2013).

Una implementación exitosa de un modelo de gestión de información espacial se efectúa mediante la construcción de un entorno que permita la articulación de capacidades y esfuerzos de una organización para afinar los procesos asociados al ciclo de vida de la información geográfica.

Esto se logra a partir de 5 componentes claves: Recursos de Información, Servicios TIC, Cultura y fortalecimiento organizacional, Marco de políticas y estándares y el colectivo de individuos (Comunidad) que interactúa con los procesos antes mencionados (ver figura 7).

Por excelencia, este tipo de entornos se logra mediante la implantación de infraestructuras de datos espaciales. El desarrollo de una IDE en cualquier organización[2], supone la estandarización, la interoperabilidad, la explotación y la publicación de información geoespacial para soportar las operaciones de negocio.

[2] Información del desarrollo de la solución informática diseñada para para el Administrador de Infraestructura Ferroviaria de España IDEADIF construida por la compañía INDRA, recuperado de www.indracompany.com/sites/default/files/ideadif_0.pdf

Figura 7. *Modelo de gestión de información espacial corporativo. Fuente: Elaboración propia.*

Entre sus principales objetivos están: Proporcionar una plataforma orientada a servicios trasversal que responda a las necesidades y demandas de productos informacionales estandarizados. Establecer las directivas de calidad de dichos productos. Facilitar la interoperabilidad de recursos geográficos dentro y fuera de la organización, así como integrar distintos sistemas de información y plataformas. Integrar la información geográfica con el resto de los procesos de negocio asegurando su disponibilidad en todos los niveles de la organización.

Un óbice a la hora de tomar decisiones inteligentes basadas en criterios geográficos es que los recursos de información requeridos no están preparados para ser utilizados por diferentes motivos: desconocimiento de la existencia de éstos, la falta de capacidades técnicas para su uso, el insuficiente respaldo documental disponible e incluso la inconsistencia y falta de integridad en los datos que lo componen.

Al respecto, los usuarios de estos recursos no encuentran lo que necesitan y terminan optando por construirlos a medida. Análogamente, resulta bastante habitual encontrar que, al interior de las propias organizaciones, se tiene desconocimiento de los recursos con los que se cuenta, incluso de una dependencia a otra. Esta falencia en los procesos de gestión y sincronización provoca que se realicen consecutivas reconstrucciones de tales productos bajo las mismas características (Anaya et al., 2002).

Una iniciativa IDE a nivel organizacional proporciona el marco de acción para la optimización de la producción, el mantenimiento y la distribución de dichos recursos. Sin embargo, uno de los requisitos principales para el establecimiento de una iniciativa de este tipo es justamente contar con metadatos que describan tales recursos disponibles, siendo este un requisito esencial para localizarlos, descubrirlos y evaluarlos (Criado et al., 2007).

4.2. Infraestructuras de datos espaciales corporativas

El concepto de infraestructura de datos espaciales (IDE) institucionales o de nivel corporativo nace de la necesidad interna de las organizaciones de integrar y unificar modelos e interfaces de sistemas empresariales, cuando advierten diferentes soluciones basadas en sistemas de información geográfica (GIS, por sus siglas en inglés) que, pese a simplificar flujos de trabajo y soportar procesos de negocio a partir de productos informativos (Tomlinson, 2008), el enfoque sigue centrado en una arquitectura empresarial orientada a aplicaciones.

De hecho, uno de los grandes problemas de los GIS durante mucho tiempo ha sido el acceso y la compartición de información geográfica de interés entre múltiples usuarios. Así mismo, un GIS no soluciona problemas, por ejemplo, asociados a duplicidad y redundancia en la captura y almacenamiento de datos que ocurren en las diferentes dependencias al interior de una institución. Tal falta de comunicación y colaboración conlleva a gastos innecesarios de recursos económicos y humanos, encareciendo esta tecnología.

Con base en lo anterior, las IDE corporativas emergen como una posibilidad de gestionar los activos de información de una corporación de manera distribuida. Esto sugiere que dichos activos no residen en una única máquina, sino que se encuentran físicamente separados en múltiples servidores, conectados entre sí por una red formando un sistema distribuido.

Bajo este modelo de operación, se ofrecen recursos de información geográfica que pueden ser intercambiados en tiempo real por cualquier funcionario que navegue en la internet/intranet.

Este beneficio se materializa a través de una plataforma de productos informativos y servicios interoperables que permite de forma eficiente la integración de los distintos sistemas GIS como una única plataforma abierta, escalable e interoperable (Masser, 2005).

Existe cierto consenso en considerar que el concepto tuvo su origen en una orden ejecutiva por el entonces mandatario de los Estados Unidos de América, Bill Clinton, quien firmó en 1994 la declaración para crear el FGDC (comité federal de datos geográficos, por sus siglas en inglés), concediendo una importancia estratégica a este tipo de iniciativas, promoviendo el desarrollo coordinado para el uso, difusión y estandarización de la información geográfica (Rojas Guerrero, 2014; Clinton, 1994).

4.2.1. Conceptualización

De acuerdo con Gallego Priego (2017), el concepto de infraestructura de datos espaciales es, a menudo, utilizado para referirse a la colección de tecnologías de base, políticas y acuerdos institucionales que facilitan la disponibilidad y el acceso a la información geográfica.

Se emplea el término “infraestructura”, teniendo en cuenta que, por definición según la RAE, corresponde a un conjunto de elementos o servicios que se consideran necesarios para la creación y funcionamiento de una organización cualquiera.

De este modo, el desarrollo de una infraestructura bajo el paradigma espacial requiere poner en comunicación coherente y colaborativa diferentes sistemas de información GIS que operan de manera independiente.

En particular, Warnest (2005) asegura que el tratamiento de la información espacial como infraestructura surge con la necesidad de regulación de las actividades de acceso y uso de ésta tanto del sector público como del privado.

Dicha situación sugiere que, mediante este tipo de proyectos, se proporcionan diferentes mecanismos para que las partes implicadas (proveedores, consumidores, etc.) alcancen acuerdos consensuados que garanticen una manera homogénea de operar.

Por lo tanto, es posible ofrecer un acceso estandarizado a los recursos que se producen individualmente para que sean incorporados correctamente al flujo de procesos

medulares de una organización de manera conjunta, es decir, sean interoperables en una infraestructura común.

La directiva europea INPIRE (Infraestructura de Información Espacial de Europa, por sus siglas en inglés) afirma que los problemas relativos a la disponibilidad, calidad, organización y accesibilidad de información espacial son comunes a un gran número de políticas y de temáticas, y se hacen sentir en los diferentes niveles de la administración.

La resolución de estos problemas requiere medidas que atiendan al intercambio, puesta en común, acceso y utilización de datos espaciales interoperables y de servicios en dicho ámbito, medidas que conciernen a los diferentes niveles de la administración y a los diferentes sectores. Es necesario, por todo ello, instituir una infraestructura de información espacial en la Comunidad.

Wytzisk & Sliwinski (2004) precisan que una IDE puede ser entendida como una colección escalable y adaptable de servicios técnicos y humanos, interconectados más allá de los límites organizacionales, administrativos y de sistemas, mediante el uso de interfaces estándar.

Estos servicios permiten a los usuarios de diferentes dominios de aplicación participar en las cadenas de valor mediante el acceso transparente a la información espacial y recursos de geoprocesamiento.

En esa misma línea, Valencia (2008), puntualiza una IDE como un conjunto de recursos técnicos, informáticos y humanos para integrar información geográfica (servicios, datos y metadatos) con el objetivo de permitir su difusión a cualquier usuario, fundamentalmente a través de internet, para poder realizar distintas acciones sobre dicha información (localización, identificación, selección, tratamiento, análisis, etc.) y que permita, a su vez, el desarrollo de nueva información, la toma de decisiones, el desarrollo de protocolos de actuación, etc., para el propio estamento encargado de su gestión.

Dicho esto, vale la pena agregar que, para hacer funcional una IDE, también se debe incluir los acuerdos organizativos necesarios para coordinarla y administrarla a escala institucional, local, regional, nacional y transnacional.

Finalmente se invita a esta disertación, la definición de Warnest (2005) al plantear que una IDE se conforma como un sistema holístico formado por los acuerdos

institucionales, la coordinación, las políticas, los datos y las normas, las redes de acceso y distribución, y el conjunto de usuarios y proveedores de información espacial.

Gestionar información espacial mediante una IDE puede resultar una tarea compleja de acometer, especialmente si sobre ésta recae la responsabilidad de custodiar y administrar información de toda una nación a nivel detallado (escalas grandes).

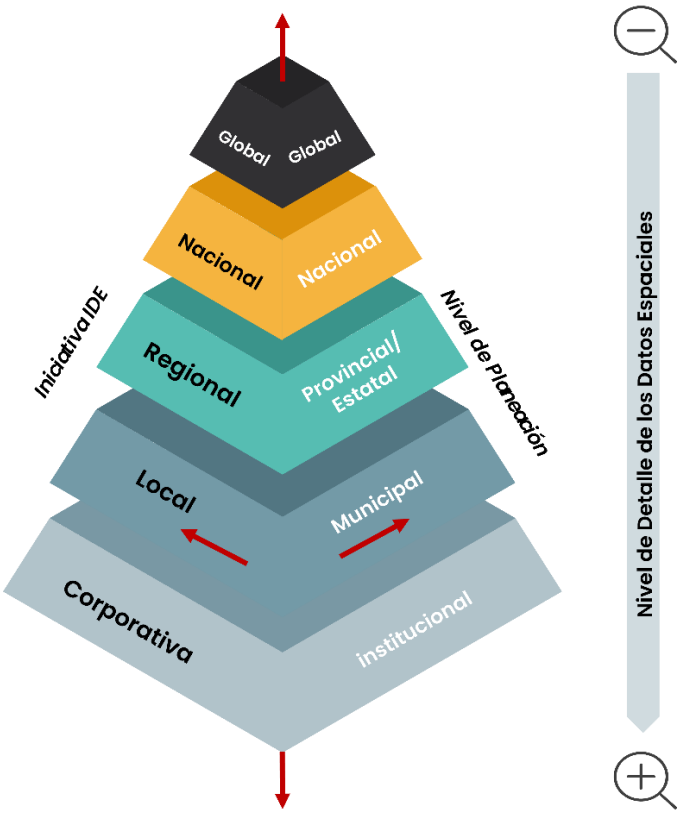


Figura 8. *Jerarquía de las IDE. Fuente: Elaboración propia.*

Evidentemente, en la práctica, aunque pudiese ser posible, demandaría una basta cantidad de recursos para ser funcional. Una alternativa viable obedece a una jerarquización (ver figura 8) considerando niveles de implementación, con base al nivel de detalle de los datos que se producen y consumen a esa escala de trabajo y el nivel de planeación que implica su gestión (Rajabifard & Williamson, 2001).

Nótese en la figura 8 que las IDE de distintos niveles poseen dos tipos de relaciones (representadas en flechas rojas), una en dirección vertical exponiendo relaciones de

dependencia y complementariedad, y otra en dirección horizontal, que pretende armonizar iniciativas del mismo nivel.

Cabe mencionar que el objetivo es que los datos espaciales se adquieran una única vez en el nivel que sea más eficiente según su propósito, y se compartan con otros niveles.

Así, en lo que respecta a una IDE de nivel corporativo, se trata entonces de un proyecto colaborativo de planeación institucional, el cual responde al papel clave y estratégico que adquiere la información geográfica en una organización. Esta proporciona los mecanismos y el ecosistema para disponer y compartir recursos informacionales para apoyar procesos de toma de decisiones de negocio.

Por esta razón, puede atribuirse dos ventajas para su implementación: Por un lado, maximiza la gestión de este tipo de recursos de escala grande. Por otro, cubre el amplio abanico de necesidades y requerimientos de información detallada a nivel de interdependencias e incluso trascendiendo al ámbito intersectorial.

4.2.2. Componentes

Una IDE en general está formada por una colección de componentes, algunos institucionales (intangibles) y otros tecnológicos (tangibles). De acuerdo con Bernabé-Poveda & López-Vázquez (2012), los componentes o marcos básicos o de alto nivel para la implementación exitosa de una IDE (ver figura 9), son:

- **Marco institucional**, el cual comprende todas aquellas disposiciones y políticas que permite el liderazgo institucional o capacidad de coordinación de los actores. Se asumen los lineamientos, principios y estrategias que orientan y regulan la gestión de los datos geoespaciales al interior de la organización.

Para el contexto de una IDE corporativa se propone: acuerdos y políticas en cuanto a gobierno y custodia de datos, actualidad y calidad de datos, disposición, acceso y uso de datos, restricciones legales, seguridad y privacidad de los datos, control de recursos financieros para funcionamiento y sostenibilidad de la IDE, acuerdos de contribución y soporte entre dependencias, implementación de estándares corporativos y gestión del conocimiento.

- **Marco tecnológico**, mediante el cual la IDE pueda operar. Esto es, componentes de acceso y distribución: redes, aplicaciones y servicios de interacción humana, de gestión de información, de flujos de datos, de procesamiento, de comunicación y de administración del sistema (Delgado & Cruz, 2009).

Dado que, una IDE consiste en una estructura virtual en red accesible por internet, la infraestructura informática requerida sigue un modelo de arquitectura Cliente-Servidor que trabaja en modo multicapa orientada a servicios.

- **Marco** geográfico. Sin embargo, por conveniencia para este TFM se sugiere sustituir el término geográfico por **informacional**. Está formado por los conjuntos de datos espaciales (datos fundamentales e información temática o de negocio), servicios web geográficos y productos cartográficos que se ofertan a través de una IDE, así como los registros de metadatos que los describen para facilitar su localización y acceso.

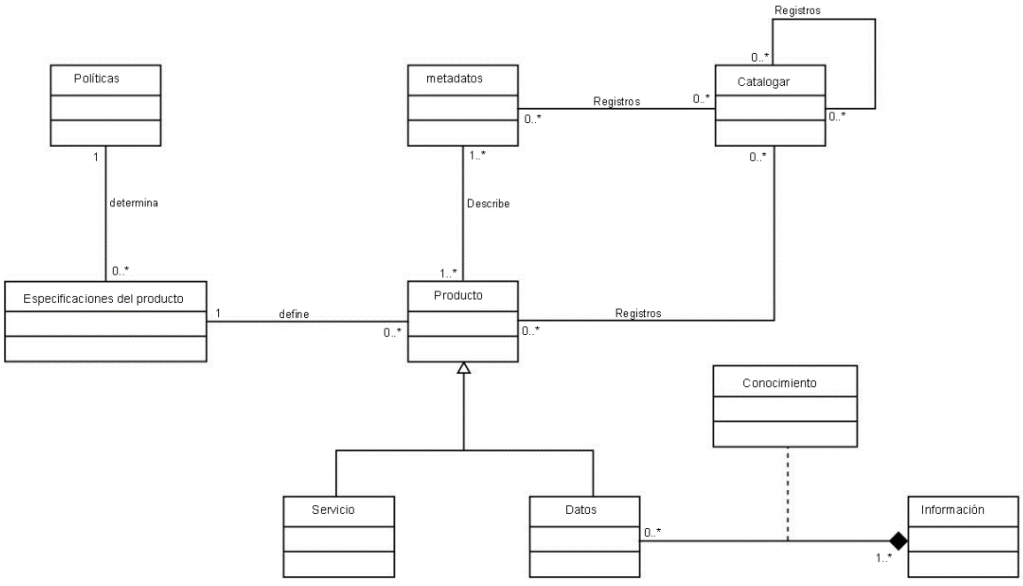


Figura 9. *Propuesta de marco informacional de una IDE corporativa. Fuente: Elaboración propia.*

La figura 9 muestra un diagrama UML de clases que describe la relación de los elementos de esta categoría, desde su semántica hasta su comportamiento en el contexto de una IDE corporativa. Adviértase que, la clase Producto, por ser el elemento más relevante desde este marco, se encuentra en el centro del diagrama.

La clase Políticas representa los lineamientos definidos desde el punto de vista empresarial o corporativo, que restringen y apuntan a las especificaciones técnicas del producto informativo, representadas por la clase Especificación del producto.

Los productos geográficos son descritos por los metadatos (clase Metadatos), y ambos están registrados en catálogos (clase Catálogo). Los productos cartográficos, como es lógico de suponer, se pueden obtener al transformar servicios y datos espaciales, según la necesidad de negocio.

Finalmente, los datos se utilizan con la ayuda de conocimientos previos, como fuente de información que puede generar nuevos conocimientos al interior de la organización.

- **Marco** social. No obstante, para ser consistentes con los planteamientos conceptuales, se propone el término **organizacional** para agrupar el conjunto de actores implicados en el desarrollo y uso de la IDE, las comunidades de interés o cualquier parte interesada (stakeholders).

Estos actores son: proveedores de servicios, productores de datos e información, desarrolladores de software, colaboradores en la definición de estándares, políticas y normas, intermediaros y usuarios finales (empleados o clientes).

La ilustración a continuación sintetiza la interacción de los cuatro componentes anteriormente descritos mediante otro modelo UML de clases, destacándose la participación que adquiere la clase metadatos, al permitir catalogar los productos informativos que circulan al interior de una IDE coporativa.

Según Gallego Priego (2017), la incorporación de metadatos se trata de un componente clave en la gestión, dado que la utilización de éstos permite un mejor acceso y recuperación de los recursos geoespaciales almacenados en los sistemas de una organización.

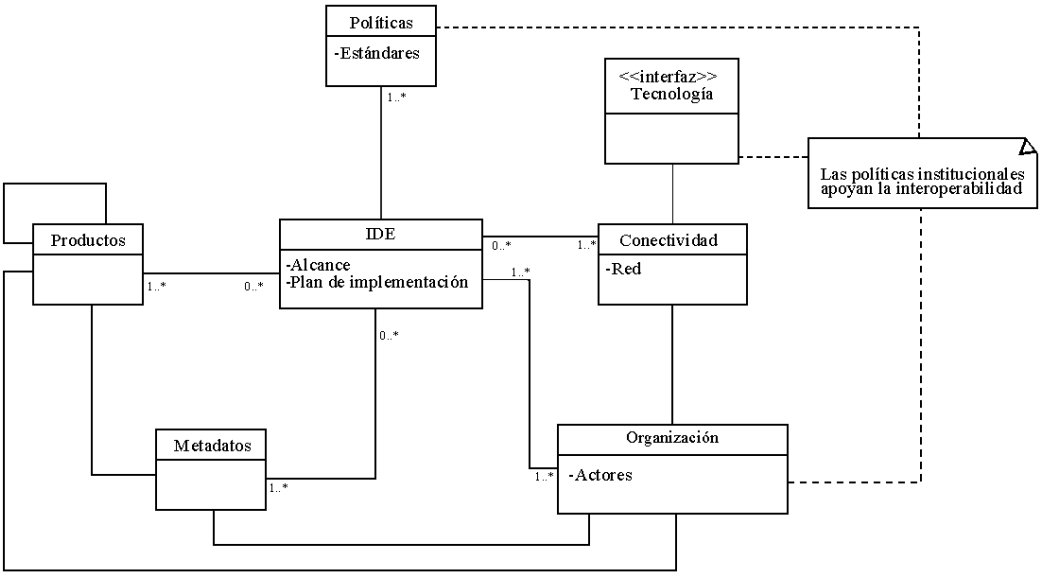


Figura 10. *Componentes de una IDE corporativa. Fuente: Elaboración propia.*

4.2.3. Arquitectura de una IDE corporativa

La implementación funcional y técnica de una IDE se debe fundamentalmente al avance de las tecnologías de la información y las comunicaciones (TIC). En efecto, este soporte tecnológico constituye la pieza tangible mediante la cual es posible llevar a cabo los procesos de gestión de los recursos de información geográfica para armonizarlos, disponerlos de modo distribuido y descentralizado, y reutilizarlos en los procesos misionales.

En concordancia con Borba et al., (2014), el marco tecnológico define la infraestructura de comunicaciones, las aplicaciones de software y el hardware necesarios para su funcionamiento. Sobre el particular, la mayor parte de los sistemas utilizados en la gestión de la información geográfica son sistemas abiertos formados por componentes interoperables, escalables y que utilizan estándares abiertos (Gallego Priego 2017).

Entre las arquitecturas de sistemas posibles, **el modelo Cliente-Servidor** (ver figura 11) es el estándar que se ha preferido por la mayoría de las organizaciones que pretenden implementar este tipo de proyectos.

El modelo se basa fundamentalmente en dos actores: uno con rol de proveedor de recursos y el otro con el rol de consultor de los recursos, comunicándose a través de redes de intranet o internet.

Su operación es bastante sencilla. Un usuario, mediante un programa ejecutable llamado cliente, envía una petición a otro programa llamado servidor, esperando entre tanto una respuesta (servicio).

Existen dos tipos de aplicaciones cliente: **El cliente ligero** o cliente web, el cual permite ejecutar una tarea sin que se tenga que instalar ningún programa (generalmente un navegador web está en capacidad de realizar esta función). **El cliente pesado** o de escritorio que, por el contrario, si requiere la instalación en máquina de la aplicación encargada del dialogo con el servidor para ejecutar la tarea.

El servidor, situado en una máquina remota, provee el servicio que se puede obtener desde una conexión en red, una vez ha aceptado la petición que hace el cliente.

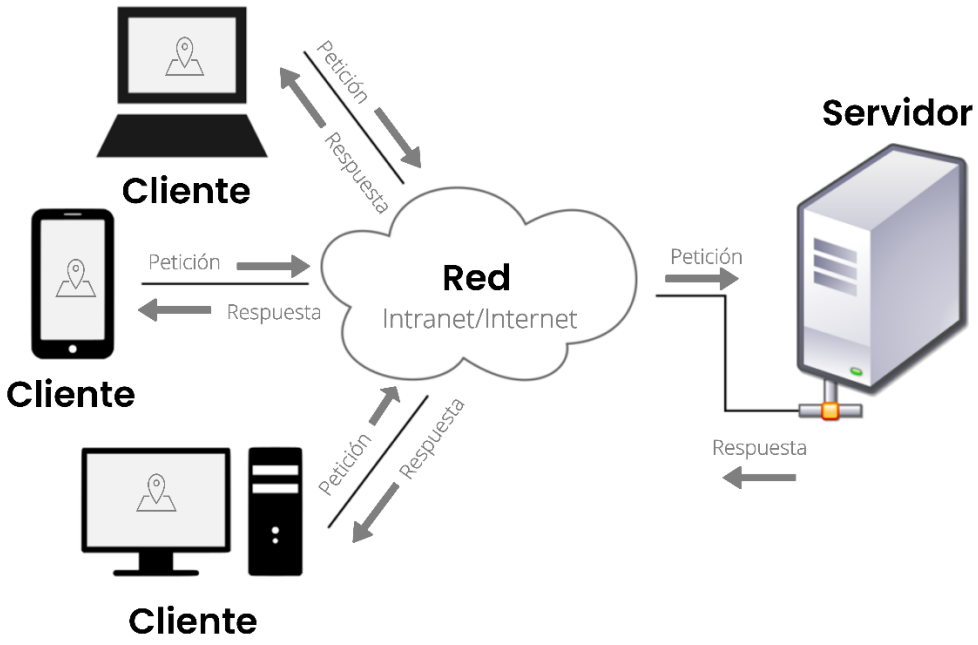


Figura 11. *Arquitectura cliente-servidor. Fuente: Elaboración propia.*

Al estar basada en estándares, el servidor es capaz de comprender y ejecutar la solicitud devolviendo un resultado lógico al usuario solicitante (Lizama et al., 2016). Este funcionamiento se conoce como **Arquitectura Orientada a Servicios** (SOA), y es un tipo de diseño de software que permite reutilizar operaciones gracias a las interfaces de servicios que se comunican a través de una red con un lenguaje común (protocolos).

En ese aspecto, una SOA integra los elementos del software que se implementan y se mantienen por separado, y permite que se comuniquen entre sí y trabajen en conjunto para formar aplicaciones en distintos sistemas.

Estas arquitecturas, según Klopfer (2005), están concebidas para utilizarse bajo el dominio de estándares abiertos, lo que permite que un sistema esté compuesto por bloques de componentes de software. Estos pueden ser elegidos, ejecutados y mantenidos de acuerdo con las necesidades de los usuarios, independientemente del proveedor de soluciones y de los modelos de almacenamiento de información que se hayan implementado.

Constantemente, bajo este modelo de operación, se acude a la ejecución de servicios web (servicios de acceso o servicios en red). Según el consorcio W3C se trata de un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la red y que intercambian datos entre sí con el objetivo de satisfacer las necesidades de negocio.

De esta manera, los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio invocando éstos a través de la red. Debido al marco de interoperabilidad entre las diferentes aplicaciones, es posible acoplar una amplia variedad de plataformas y entornos de trabajo para atender estos fines de gestión (Booth & Liu, 2007; Gallego Priego 2017).

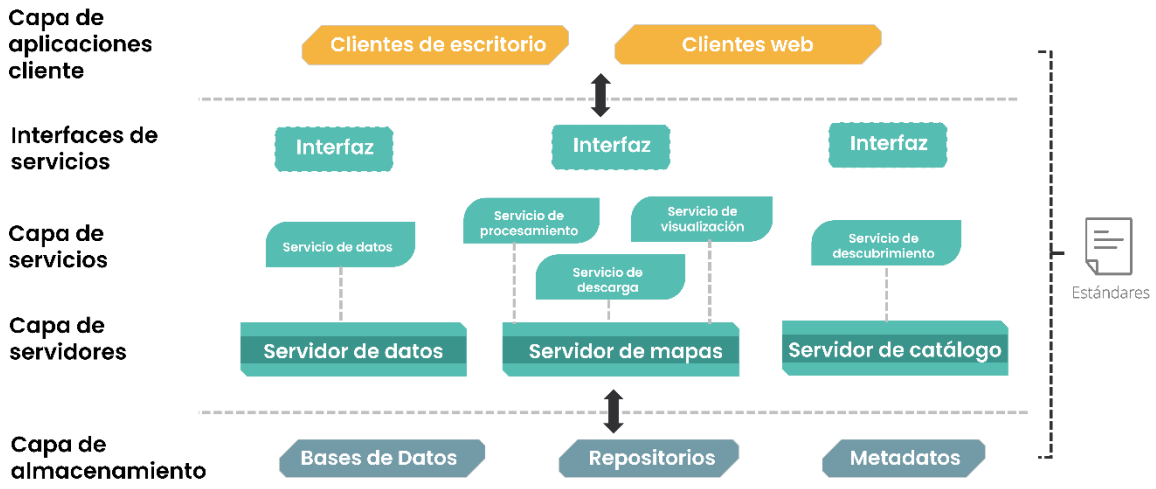


Figura 12. *Arquitectura multicapa tipo SOA de una IDE. Fuente: Elaboración propia.*

Los estándares, en la figura 12, actúan como guías para ayudar al personal técnico en la creación y la conexión de componentes de una IDE de manera interoperable.

La arquitectura funcional para una IDE corporativa se ha diseñado con el propósito de unificar y homogeneizar toda la información, aplicaciones y procesos GIS que se llevan a cabo en una organización, y disponerla al servicio de todas las unidades de negocio. Por ello, típicamente se proporciona una aplicación corporativa tipo geoportal (plataforma con interfaz web actuando como punto de entrada a los contenidos geográficos), garantizando la independencia de sistemas y asegurando la interoperabilidad entre sus elementos constitutivos.

Conviene precisar en este punto que una IDE corporativa no compite ni trata de reemplazar los GIS tradicionales. Por el contrario, permite conectar múltiples soluciones de este ámbito para que sean más productivas e interoperables al interior de la organización.

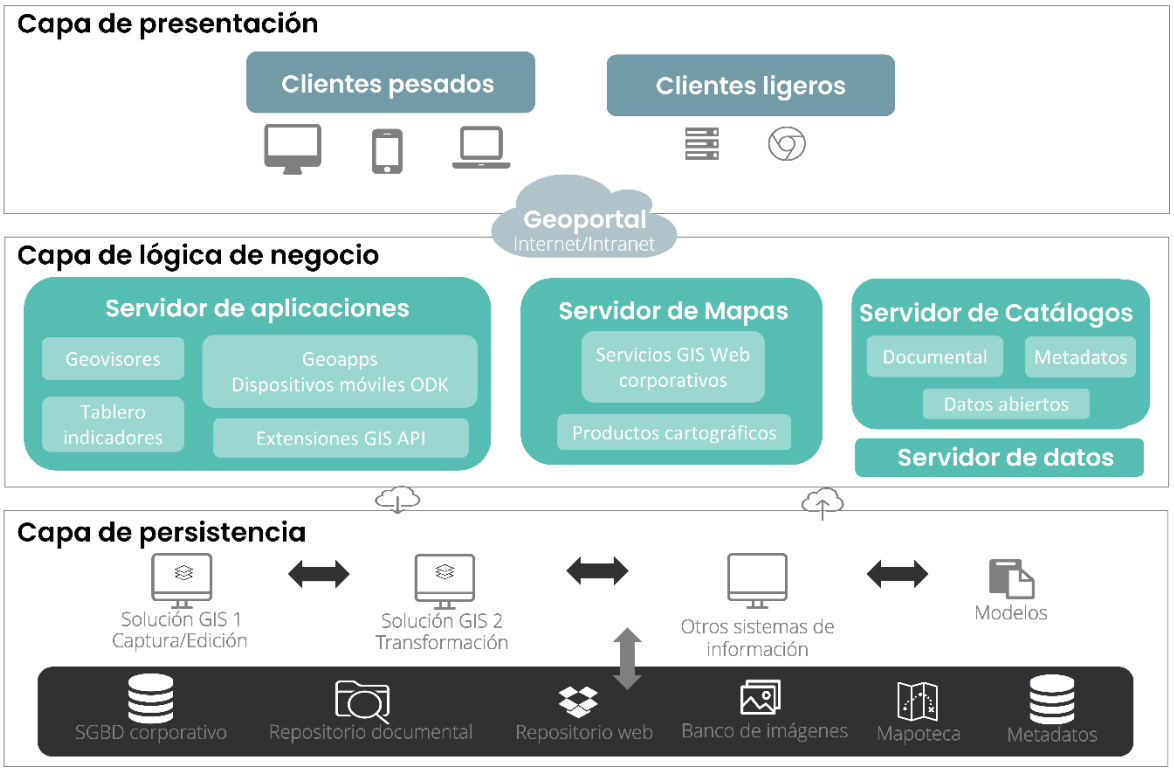


Figura 13. *Propuesta de arquitectura funcional de una IDE corporativa. Fuente: Elaboración propia.*

Una IDE no sólo es un conjunto de geodatos, servicios interoperables, geoportales y metadatos. Es necesario incluir más elementos para que el proyecto alcance su objetivo con éxito.

Para ser interoperables, se debe trabajar activamente en el proceso de asegurar que los sistemas, los procedimientos y la cultura de una organización se gestionen de forma que se maximicen todas las oportunidades de intercambio y reutilización de la información y de los servicios, ya sea interna o externamente (Klopfer, 2005).

A continuación, se lista una propuesta de once criterios que han de ser considerados al momento de calificar el éxito de implementación de una IDE corporativa o institucional, a saber:

- **Datos y Servicios Web Geográficos Disponibles:** Permite relacionar el tipo de datos, conjunto de datos y/o geoservicios web disponibles, su especificación, posibilidad de acceso y descarga, y correcto funcionamiento de éstos al momento de hacer la petición a través de un cliente ligero o pesado.
- **Servicios TIC:** Hace referencia a las soluciones de carácter informático utilizadas en la IDE para brindar los servicios para los que fue concebida, a los proveedores y usuarios. En este sentido, corresponde todas aquellas herramientas disponibles, no sólo para apoyar la producción y la administración de la información espacial, sino además todos aquellos canales que posibilitan el flujo e intercambio de los datos y/o servicios. En principio, basta para este ítem la existencia de un Geoportal y por ende de un visualizador cartográfico o geovisor (con sus operaciones básicas de navegación), un gestor de metadatos y un catálogo de consulta de los datos y geoservicios publicados.
- **Disponibilidad de Metadatos:** Se considera si en la IDE es posible consultar y descargar información asociada a los metadatos de los geodatos, servicios GIS web disponibles o productos informativos de tal manera que se facilite su consulta, evaluación, comparación y utilización responsable de la información que describen.
- **Documentación:** Determina si la iniciativa IDE ofrece a los usuarios documentación asociada a políticas, acuerdos interinstitucionales, manuales de ayuda, memorias técnicas, informes de gestión, documentos de mejores prácticas o datos bibliográficos que faciliten la alfabetización de usuarios para entender el propósito y uso de la plataforma de gestión de información geográfica al interior de la organización.

- **Cumplimiento de estándares:** Permite establecer si la IDE cumple con estándares tecnológicos y estándares de datos. Su implementación hace posible la coherencia, compatibilidad e interoperabilidad entre los datos, los servicios y las aplicaciones customizadas de la IDE corporativa.
- **Institucionalidad:** Identifica qué tan fuerte es el fortalecimiento institucional de la IDE en cuanto a la participación de sus dependencias que participan en el ciclo de vida de la información espacial o el relacionamiento y cooperación con instituciones externas afines al negocio, mediante el cumplimiento de acuerdos y alianzas interinstitucionales.
- **Usabilidad:** Refiere a la capacidad para ser entendida la IDE, para ser operada por el usuario final, capacidad para ser aprendida, armonía e intuitividad en la interfaz gráfica, operatividad y consistencia en la infraestructura tecnológica y servicios que presta.
- **Geoecosistema digital:** Disponibilidad de otras aplicaciones con componente geográfico integradas en la plataforma que enriquecen no solo la utilización de los datos disponibles agregando valor, sino en general atendiendo necesidades de acceso y manipulación de información por parte de los usuarios.
- **Gestión de conocimiento:** Este criterio pretende establecer si en la IDE se observan iniciativas orientadas a la formulación de proyectos de investigación, organización y participación de eventos que contribuyan a visibilizar las acciones desarrolladas por la IDE. Transferencia de conocimiento mediante acciones de formación presencial o virtual diseñadas para cualificar las competencias técnicas en la gestión de la información geográfica, que contribuyan a la promoción de una cultura geográfica en general a todos los usuarios que acceden a la IDE.
- **Interacción con otras iniciativas IDE:** Implica la articulación y colaboración con otras iniciativas del mismo nivel jerárquico o se identifican relaciones de dependencia o complementariedad con proyectos de este ámbito de orden superior o inferior.
- **Innovación:** Relaciona todos aquellos proyectos cuyo propósito es la innovación en los flujos de procesos e incorporación de nuevas tecnologías en el marco de la IDE corporativa. Considérese proyectos que permitan fomentar la investigación, desarrollo e innovación (I+D+i), maximizar el uso y aprovechamiento de los recursos geoespaciales para la generación de valor articulando técnicas de inteligencia

artificial y ciencia de datos, procesamiento bigdata, internet de las cosas (IoT), por citar algunos.

4.2.4. Interoperabilidad y estándares

La interoperabilidad es un requisito previo para permitir la comunicación electrónica y el intercambio de información geográfica entre las diferentes dependencias y al exterior de la organización. De hecho, se trata de uno de los aspectos claves para una implantación efectiva de una IDE, cualquiera sea su nivel jerárquico, y para su óptimo desempeño.

La interoperabilidad puede ser entendida, para efectos de este TFM, como la capacidad de las entidades digitales distribuidas, autónomas y heterogéneas de comunicarse a fin de generar conjuntamente una información integrada a pesar de sus diferencias.

De acuerdo con Gallego Priego (2017), el término interoperabilidad se puede definir como la capacidad de los sistemas de información, y de los procesos corporativos que soportan, de intercambiar datos y posibilitar la puesta en común de información y conocimientos.

Como es lógico de suponer, interoperabilidad no está asociada estrictamente con el componente tecnológico, pese a que, en sus inicios, las organizaciones sólo manifestaban interés de alcanzarla sólo en determinados dominios, mediante la puesta en marcha de una infraestructura común.

En la medida en que ha evolucionado y madurado conceptual y metodológicamente el desarrollo de las IDE, aspectos como la compatibilidad de instrumentos normativos, se han incorporado a este marco de trabajo. Así pues, existen fundamentalmente cuatro dominios de interoperabilidad, como se ilustra a continuación:

***Figura 14**. Tipos de interoperabilidad. Fuente: Elaboración propia.*

Los metadatos proporcionan a los sistemas la capacidad de interactuar a diferentes niveles por lo que resultan ser la pieza clave para lograr la interoperabilidad (Beltrán Fonollosa, 2013).

Buena parte de los requisitos de interoperabilidad pueden recaer en los metadatos. Algunos autores como Oliva Santos et al., (2009) proponen la utilización de elementos semánticos (ontologías) en los GIS e IDE ya que facilita el desarrollo de tareas como la búsqueda de mapas y constituye la base para una verdadera interoperabilidad basada en el conocimiento.

La adopción de normas, estándares y acuerdos tiene como finalidad hacer efectiva la interoperabilidad entre todos los agentes implicados en una IDE institucional. En efecto, estas iniciativas necesitan estándares para asegurar que los datos espaciales, servicios interoperables, productos cartográficos y aplicaciones puedan ser utilizados, compartidos y combinados entre diferentes usuarios.

En palabras de Gallego Priego (2017), los estándares ayudan a crear un entorno efectivo y consistente para implementar las capacidades geoespaciales de los sistemas de gestión de información espacial.

Rojas Guerrero (2014) los define como el conjunto estable de reglas, procedimientos, guías e instrucciones para adelantar esta gestión. Por tanto, son acuerdos específicos que se aplican en la producción, acceso y uso de la información espacial. Se apoya así la estabilidad de la infraestructura y se asegura que la información requerida por los conjuntos de actores cumpla con criterios que permitan compatibilidad e interoperabilidad para aplicarse a cada proceso del ciclo de vida de los datos.

De acuerdo con este mismo autor, se puede asociar a la estandarización tres objetivos para garantizar que los productos, procesos y servicios cumplan con su propósito: Simplificar (empleo de procesos menos complejos, más cortos y eficientes). Unificar (intercambio entre diferentes niveles). Especificar (claridad y precisión en los requerimientos y expectativas).

En el contexto de una IDE corporativa se deben estandarizar todos aquellos elementos que intervienen en la compartición de recursos, estos son: los formatos de intercambio de datos, la descripción de los metadatos para datos, servicios web geográficos y productos informativos, y las interfaces de los servicios interoperables.

Fundamentalmente se tienen dos tipos de estándares en este marco de trabajo: las normas o estándares de jure y los estándares de facto.

Un estándar de jure o norma, es todo documento que armoniza aspectos técnicos de un producto, servicio o componente, definido como tal por un organismo oficial de normalización. La ISO (Organización Internacional de Normalización, por sus siglas en inglés) o la CEN (Comité Europeo de Normalización) tienen esta competencia. Con respecto a la información geográfica, el comité técnico 211 de la ISO tiene como objetivo el desarrollo de normas en este campo, agrupadas en la familia ISO 19100.

En cuanto a los **estándares de facto**, o estándar propiamente dicho, es cualquier documento o práctica que, sin ser norma, está acreditado y aceptado voluntariamente para el uso. Cumple una función similar a la de una norma, pero son desarrollados por un consenso de la industria.

Bajo esta definición, se incluyen las recomendaciones que son directrices que promueve un organismo que intenta armonizar prácticas y usos en un colectivo de personas, por ejemplo, los perfiles de metadatos que adoptan los institutos geográficos nacionales para documentar sus recursos.

De modo semejante, se incluyen las especificaciones, las cuales consisten en una descripción técnica, detallada y exhaustiva de un producto o servicio, que contiene toda la información necesaria para su producción.

La organización principal que proporciona la mayoría de los estándares de facto para una IDE es el consorcio OGC (Open Geospatial Consortium).

Por otro lado, también existe una taxonomía de estándares propuesta por el grupo de trabajo del FGDC (Federal Geographic Data Committee), quien categorizó los estándares en: Estándares de Datos, Estándares de Procesos, Estándares Organizaciones y Estándares Tecnológicos. En este TFM se hace hincapié en los estándares de datos y los estándares tecnológicos.

Entiéndase por **estándares de datos** aquellos que describen como se debe estructurar, generar y documentar los objetos geográficos y productos informativos, buscando garantizar la calidad de éstos. La figura 15 pretende hacer una síntesis de los estándares requeridos en la producción de un recurso informacional al interior de una IDE corporativa.

Figura 15. *Estándares básicos para la producción de recursos de información geográfica. Fuente: Elaboración propia.*

En lo que respecta a los **estándares tecnológicos**, permiten facilitar la interoperabilidad entre desarrollos que se generen independientemente, determinando estructuras ya sean de formato, de lenguaje, distribución y acceso a los datos.

Morales et al., (2012) propone un sumario de los estándares tecnológicos (servicios web geográficos) definidos por la OGC mediante un esquema preconceptual con sus principales propiedades y operaciones (ver figura 16).

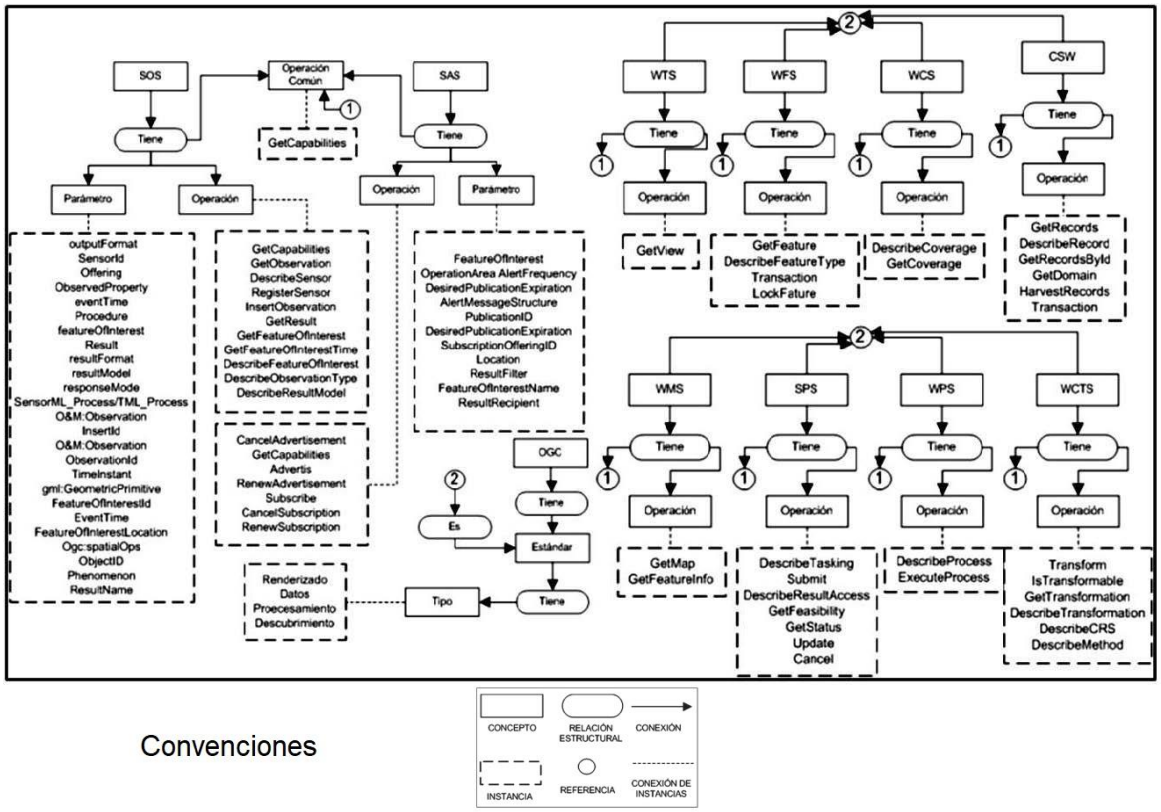


***Figura 16**. Estándares de servicios web geográficos por OGC. Fuente: Morales et al., (2012).*

La OGC propone algunas especificaciones de servicios web geográficos que se utilizan comúnmente en la industria geoespacial. Los servicios web más relevantes para una IDE corporativa son:

a) **Web Map Service (WMS)**, que provee mapas por medio de las interfaces de operaciones GetCapabilities, GetMap y GetFeatureInfo.

b) **Web Feature Service (WFS)** proporciona elementos tipo vector en el formato GML (Geography Markup Language) usando las interfaces de operaciones GetCapabilities, DescribeFeatureType, y GetFeature.

c) **Web Coverage Service (WCS)** posibilita servir datos ráster mediante las operaciones GetCapabilities, DescribeCoverage y GetCoverage.

d) **Web Processing Service (WPS)**, el cual se diseñó para estandarizar la forma de ofrecer operaciones. Este estándar proporciona normas para describir un cálculo o proceso, así como la forma de realizar las peticiones al proceso y de responder dicha petición.

Un WPS puede contener operaciones para tratar tanto datos vectoriales como ráster, provenientes de la red o del propio servidor. De esta manera las operaciones obligatorias definidas para este servicio son GetCapabilities, DescribeProcess y Execute.

e) **Catalogue Service for Web (CSW)** proporciona un puente entre el proveedor y los usuarios de los servicios de información espacial, de esta manera, permite a los clientes descubrir servicios OGC en la red.

Otros servicios web geográficos adicionales, que pueden resultar de interés son:

(a) el **Sensor Observation Service (SOS).** El SOS define una interfaz de implementación y operaciones para el acceso a observaciones desde sensores y sistemas de sensores, incluyendo remoto, in situ, fijos y sensores móviles.

(b) **Sensor Planning Service (SPS).** El SPS define la interfaz para realizar consultas que proveen información sobre las capacidades de un sensor y cómo opera éste.

(c) **Sensor Alert Service (SAS).** El SAS es un servicio especializado en publicación y suscripción de alertas generadas a partir de observaciones.

(d) **Web Terrain Service (WTS).** El propósito del WTS es producir vistas en perspectiva de datos georreferenciados, normalmente coberturas 3D.

(e) **Coordinate Transformation Service (WCTS).** La transformación de las geometrías entre sistemas de coordenadas de referencia (CRS–Coordinate Reference System) es importante cuando se integran datos de distintas fuentes. El uso integrado y la superposición visual de datos que se encuentran almacenados en distintos CRS requiere la transformación de estos a uno común.

En definitiva, la implementación de estándares en una IDE corporativa se constituye como una herramienta estratégica que posiciona la calidad (ver figura 17) como requisito

para asegurar los mayores niveles de interoperabilidad y alcanzar las mejores cualidades posibles de los recursos informacionales.

Así mismo, permite la normalización técnica de los procesos de producción y gestión de información espacial. También facilita el dialogo entre los requisitos de usuarios y las especificaciones técnicas desde la perspectiva de la calidad de los datos, aportando transparencia desde un diseño estandarizado de los productos, fortaleciendo el modelo de gestión de calidad al interior de la IDE corporativa.

Como es de esperarse, los metadatos desempeñan un rol importante al consignar las propiedades técnicas del producto informativo y las evidencias de la evaluación de calidad para ser aprobado y publicado de conformidad a un estándar en particular.

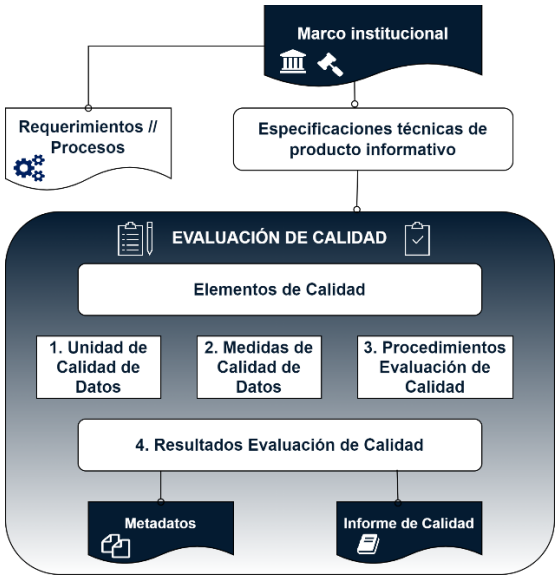


Figura 17. *Modelo de calidad de una IDE corporativa. Fuente: Elaboración propia.*

4.3. Gestión de metadatos geográficos

4.3.1. Generalidades de los metadatos geográficos

El uso de metadatos resulta ser más habitual de lo que podría sugerirse. Absolutamente toda la comunidad circunscrita en la disciplina geomática hace uso de ellos.

El modelo físico de una base de datos espacial, por ejemplo, se materializa a partir de diccionarios de datos al ofrecer las descripciones de las entidades, atributos y tipo de

datos. Los desarrollos de software geoinformático contienen comentarios entre las líneas de código para mejorar su comprensión. Las imágenes de satélite contienen los llamados archivos de cabecera para explicar cómo procesar o visualizar dichos ficheros. Todos estos ejemplos hacen uso de metadatos (no formalmente) que es información adicional que ayuda a comprender mejor el recurso asociado.

Aunque el término metadato se relacionó inicialmente con el campo de la bibliotecología, actualmente se ha extendido a los recursos digitales. Fue acuñado por Jack Myers en la década de los 60 para describir conjuntos de datos (Caplan, 2003). La primera acepción que se le dio (y actualmente la más extendida) fue la de dato sobre el dato, ya que su intención era proporcionar la información mínima necesaria para identificar un recurso.

En la actualidad, los metadatos permiten describir recursos de forma estructurada y, en base a estas descripciones, es posible organizarlos, publicitarlos y facilitar el acceso a ellos. Los metadatos son conjuntos estructurados de datos que describen otros datos y cuyo propósito es mejorar el conocimiento sobre los recursos descritos. De esta forma, las descripciones de los recursos resultan ser la pieza clave de cualquier sistema de información (Beltrán Fonollosa, 2013).

Anaya et al., (2002) por su parte presentan una definición más operacional. Para ellos, los metadatos constituyen el mecanismo para caracterizar los datos y aplicaciones, lo que posibilita que otras aplicaciones puedan hacer uso de dichos datos o invocar sus servicios. Los registros de metadatos, cada uno de ellos describiendo un recurso específico, se agrupan en catálogos, que facilitan al usuario la posibilidad de identificar los recursos que le puedan ser de interés.

Así pues, un metadato es un conjunto de descriptores estructurado y normalmente estandarizado el cual permite identificar datos, conjuntos de datos, servicios interoperables, productos cartográficos o cualquier otro recurso informacional, brindando información sobre su contenido, calidad, condición, limitaciones y otras características, con la finalidad de facilitar su descubrimiento, entendimiento, evaluación, comparación, recuperación y utilización.

Santos & Orozco (2006) argumentan que los datos que conforman un metadato generalmente dan respuesta a las preguntas sobre el recurso: qué (título y descripción),

quién (autor), cuándo (fecha de ceración y periodos de actualización), cómo (proceso de creación del recurso), dónde (localización) y porqué (propósito).

En el metadato usualmente se recoge información sobre cada una de las etapas de la existencia de los recursos que se documentan, así como de su semántica, aspecto vital para determinar la pertinencia y posibilidad de explotarlo en un campo de aplicación específico. Dicho conocimiento contribuye en última instancia a determinar la interoperabilidad de éstos y su capacidad de compartirse e integrarse en diversos sistemas (Rajabifard et al., 2009).

Bajo este enfoque, Seiner (2000) concluye que los metadatos son información documentada que, mediante el uso de herramientas y tecnologías de la información, mejoran la comprensión y el entendimiento de los datos, procesos y recursos asociados que describen.

En definitiva, cuando los recursos geográficos poseen el contexto que los hace comprensibles mediante el uso de los metadatos, entonces se convierten en información valiosa y útil para el usuario.

Los metadatos geográficos contienen diferentes niveles de información según el propósito concreto que persiguen (ver figura 18). Nebert (2004) sugiere los siguientes tres niveles de alcance o categorías:

- **Metadatos de descubrimiento.** Se describe la información necesaria para transmitir la naturaleza y el contenido de los recursos. Son los que facilitan el intercambio de datos porque son los encargados de dar a conocer y publicitar cuales son los datos existentes.

Está orientado a realizar búsquedas para descubrir qué datos existen y qué características principales presentan a partir de unos criterios específicos (búsquedas) (Santos & Orozco, 2006; Callejo, 2003).

- **Metadatos de exploración.** Proporcionan la información necesaria para que los usuarios sean capaces de discernir y comparar si el producto informativo es el más adecuado para un fin concreto o satisfacen una necesidad particular (Santos & Orozco, 2006).

Se dispone de información suficiente para asegurar que los datos son apropiados para un propósito dado, para valorar sus propiedades, así como hacer referencia a algún punto de contacto para obtener más información.

- **Metadatos de explotación.** Contienen aquellas propiedades imprescindibles para el acceso, transferencia, carga, interpretación y uso de los recursos por un cliente final. Permiten conocer el procedimiento de adquisición.

Este nivel de metadatos incluye frecuentemente diccionarios o tesauros de datos, la organización, proyección, características geométricas y otras propiedades de los datos que posibilitan su actualización, almacenamiento y un uso correcto y eficiente de los mismos (Santos & Orozco, 2006; Callejo, 2003).

Tal y como proponen la FGD Committee, (2000), Anaya et al., (2002) y Beltrán Fonollosa (2013), la creación de metadatos geográficos persigue tres principales objetivos:

El primero es organizar y mantener la inversión en geodatos realizada por una organización. Conforme el tiempo pasa o cambia el personal de una organización, los nuevos trabajadores probablemente desconocen el contenido y el uso que se debe dar a los recursos que se crearon con anterioridad y normalmente desconfiarán de los resultados que se puedan generar de ellos.

Por esta razón, las descripciones sobre su contenido y corrección proporcionadas por los metadatos propiciarán una reutilización apropiada de los mismos. Además, tales descripciones protegerán los intereses de la organización productora si surge algún conflicto debido a un uso incorrecto.

El segundo objetivo es proporcionar información referencial para catálogos de datos geoespaciales. Las IDE corporativas requieren a menudo la utilización de datos de diferentes temáticas. Sin embargo, pocas organizaciones se pueden permitir el lujo de crear todos los datos por sí mismas.

Mediante la publicación de recursos de información geográfica a través de un catálogo, las organizaciones pueden encontrarlos, compartirlos y mantenerlos. Los metadatos se almacenan en catálogos o bases de datos distribuidas de forma que se posibilita la búsqueda de estos recursos satisfaciendo ciertas condiciones (Callejo, 2003). Por esta razón, los catálogos se convierten en el corazón de una IDE corporativa. Mediante

la publicación de recursos de información geográfica a través de un catálogo, las organizaciones pueden encontrarlos, compartirlos y mantenerlos.

El tercer objetivo responde a facilitar el acceso a los recursos, su adquisición y una mejor utilización de los mismos facilitando su interoperabilidad cuando éstos proceden de diferentes fuentes (transferencia). Los metadatos ayudan al usuario u organización que los recibe en el procesamiento, interpretación y almacenamiento en repositorios internos.

Adicionalmente, autores como Balfanz (2002) y Jokela (2001), por citar sólo algunos, clasifican los metadatos geográficos (ver figura 18) siguiendo diferentes criterios: desde el punto de vista de su existencia, teniendo en cuenta su ciclo de vida o dependiendo del rol que cumplen al interior de una IDE institucional.

Figura 18. *Taxonomía de los metadatos geográficos. Fuente: Elaboración propia.*

4.3.2. Importancia de la gestión de metadatos geográficos

Figura 19. *Principales funciones de los metadatos geográficos. Fuente: Elaboración propia.*

La creación, el uso y la gestión de metadatos son considerados la columna vertebral de una IDE corporativa. Gracias a ellos, tanto productores como usuarios de información geográfica pueden disponer de productos de información documentados, lo cual contribuye no sólo a facilitar su entendimiento y por ende su usabilidad, sino al mejoramiento en la creación, almacenamiento, actualización y diseminación de éstos.

En virtud de lo anterior, de acuerdo con Gallego Priego, (2017) y Santos & Orozco (2006), los beneficios que ofrece la gestión de metadatos al interior de una IDE corporativa, son:

- Ayudan a organizar y mantener la inversión en datos y en la generación de productos informativos. Hace persistente el conocimiento acerca de estos recursos cuando el personal calificado que los creo o que tiene relación directa con ellos y conoce todas sus características, deja de prestar servicios en la institución.
- Proporcionan información explícita sobre datos, servicios y productos informativos (adhieren contexto como un valor agregado), facilitando su interpretación.
- Permiten un inventario controlado de recursos. Sin metadatos, antes de poder usar cualquier recurso, se puede gastar una gran cantidad de tiempo tratando de descubrir su existencia.

- El desarrollo de los metadatos coordinados evita la duplicidad de esfuerzos al asegurar que la organización esté consciente de la existencia de sus recursos de información, sus propiedades y su ubicación en la infraestructura, fomentando su reusabilidad. Así mismo, facilita los procesos de gestión necesarios para asegurar el mantenimiento y actualidad de los recursos.
- Proporcionan mecanismos para custodiar y auditar los datos gestionados por un sistema.
- Los usuarios pueden localizar de manera rápida, eficiente y confiable todos los datos y recursos disponibles; permiten valorar sus propiedades, validando si son apropiados para una necesidad particular o tema de interés.
- La colección de metadatos mejora los procedimientos de manejo y administración de la información en una organización.
- Reducen el riesgo de que se devalúen los datos cuando se pierde el conocimiento acerca de ellos por causas como la reubicación, reasignación o redundancia del personal responsable de los mismos.
- Posibilita un mejor y mayor intercambio o transferencia de datos entre dependencias e incluso otras organizaciones (interoperabilidad). De esta manera es mucho más claro qué se ofrece y qué se requiere, lo que facilita la cooperación y el trabajo coordinado interinstitucional.
- Los metadatos son un elemento fundamental para implementar un modelo de calidad en los productos informativos generados al interior de un GIS o una IDE corporativa, al proporcionar información sobre este aspecto; bien en la definición de su especificación técnica antes de producirlo o bien como un elemento constitutivo del metadato propiamente dicho después de su publicación.

De este último ítem, vale la pena mencionar que, la baja calidad de los geodatos, servicios de información geográfica o mapas, destruye el valor del negocio en una compañía. Dado que, es a partir del aprovechamiento de estos recursos, donde hay generación de ideas y optimizaciones de negocio que conducen a una ventaja competitiva y aseguran el posicionamiento en el mercado. Por tanto, desconocer la calidad de un producto informativo genera desconfianza en los usuarios, minando su credibilidad y posibilidad de aplicación.

4.3.3. Metodología para la creación de metadatos geográficos

La consulta, el almacenamiento y la gestión de metadatos mediante herramientas de software (gestores o catálogos) facilita el descubrimiento, el acceso y la comprensión de recursos.

Normalmente, los metadatos son creados por los proveedores de los recursos, generados de forma manual y almacenados en catálogos o herramientas de gestión, para más tarde ser encontrados con fines informativos.

Por lo anterior, es absolutamente necesario que la elaboración de los metadatos obedezca a un proceso de estandarización (juego común de condiciones y definiciones) que permitan a los usuarios comprender perfectamente a que recursos tienen acceso.

Es necesario que los metadatos sean homogéneos, completos y coherentes (Callejo et al., 2009), para ello es imprescindible definir procedimientos de catalogación que ayuden al productor a resolver las dudas iniciales y le guíen en la elaboración de los mismos.

Por lo anterior, es importante definir diferentes modelos de catalogación en función de la naturaleza de los productos de información que se desean describir (Gallego Priego, 2017).

En este orden de ideas, el proceso de creación de metadatos está compuesto por un conjunto de fases sucesivas, que serán llevadas a cabo por los productores o catalogadores de recursos de información geográfica, como se ilustra en seguida:

Figura 20. *Proceso general de creación de metadatos. Fuente: Elaboración propia.*

El proceso inicia identificando plenamente el recurso de información a metadatear, un mapa, por ejemplo. Así mismo, se debe decidir la forma de asignación del metadato al recurso.

Para Duval et al., (2002) los metadatos pueden ser embebidos en el recurso. Se almacenan embebidos y codificados en la cabecera del documento, permitiendo que tanto recurso como metadato se transporten de manera unívoca.

Por otro lado, pueden ser almacenados en archivos asociados al recurso. Aporta ventajas en la creación independiente y poder editarlos sin necesidad de alterar el contenido del recurso. Sin embargo, se detectan desventajas en la gestión simultanea de los recursos y los archivos que almacenan los metadatos.

Finalmente, la tercera opción consiste en un almacenamiento independiente o externo, generalmente en una base de datos. De esta forma, es mucho más fácil gestionar tanto los metadatos como los recursos (almacenarlos, mantenerlos, actualizarlos, convertirlos a otros formatos, etc.).

Este es el método que suelen emplear muchas organizaciones para que sus productos informativos no sean públicos, ya que de esta forma permanecen inaccesibles a los motores de búsqueda.

En el paso 2, se procede a levantar la información de los datos que una vez estructurados formarán el metadato con base a un formulario diseñado a partir de un perfil de metadato previamente adoptado. De modo semejante se toman decisiones sobre las estrategias más comunes para la compilación de metadatos:

(a) introducción manual por teclado; (b) extracción de metadatos implícitos del propio recurso o a partir del contenido (datos); (c) recolección de información durante el proceso de creación del recurso; (d) aprovechar la información del contexto en que los datos son creados o explotados; (e) búsqueda (look-up) desde una tabla de referencia; (f) medición del valor, por ejemplo, en el proceso de creación de un conjunto de datos, un sensor de medición puede proporcionar mediciones y otros valores de la captura que pueden ser incorporados en la ficha de metadatos de forma automática; (g) inferencia de metadatos a partir de otros metadatos o de los datos fuentes del recurso (Beard, 1996; Beltrán Fonollosa, 2013).

Estas estrategias pueden ser perfectamente combinadas con el fin de establecer sinergias para dar respuesta a la necesidad de levantamiento de información.

Para el paso 3, se deben explorar las posibles vías para la creación de metadatos. Por ejemplo, Day et al., (2004), contempla la creación manual, semiautomática y automática, por el autor del recurso o por parte de un especialista en gestión de información.

Tal flujo de trabajo prevé todas las posibilidades: creación automatizada, creación automatizada mejorada por el autor de los recursos, creación automatizada mejorada por el autor y por un especialista en gestión de metadatos, creación manual por el autor y mejorada por el especialista, o creación por el especialista directamente.

En este paso en particular, de acuerdo con Callejo et al., (2009), se deberían tener en cuenta algunas consideraciones a saber: Por un lado, la creación manual es un proceso denso y susceptible de comprometerse su calidad (errores); por otro, los procedimientos automáticos no pueden proporcionar la información que los productores del recurso o especialistas en gestión pudiesen aportar.

Beltrán Fonollosa (2013) añade que los metadatos creados de forma automática tienden a ser más eficientes, consistentes y menos costosos que aquellos creados manualmente, optimizando tiempos y actividades de levantamiento y posterior digitación, pese a la dificultad que implica su implementación mediante un lenguaje de programación.

Greenberg (2004) identifica dos métodos de creación automática de metadatos: la extracción y la recolección (harvesting). La extracción usa técnicas de minería de datos e indexación para recuperar ítems o etiquetar contenidos. En el harvesting se acude a técnicas de recolección de contenidos etiquetados ya existentes. Una vez elegido el método, si existen metadatos, se procesan para adaptarlos a la plantilla, y si no existen, se crean.

Así mismo, en el proceso de creación, se debe detectar qué elementos del metadato son variables y cuáles permanecen constantes para otros recursos; bastante útil en procesos de generación masiva de metadatos, por ejemplo, en una serie cartográfica.

En este caso, según Criado et al., (2007) los elementos constantes, respecto de la serie cartográfica, heredarán su contenido del metadato de la serie. Para el caso de los elementos variables, la situación no es tan sencilla ya que existirán distintos tipos de elementos variables como:

(a) elementos constantes entre unidades. En este grupo se encuentran los campos para los cuales la información es constante para todas las unidades de la serie, pero variable respecto a la serie, sirva de ejemplo el nivel jerárquico o las unidades de distribución;

(b) elementos variables entre unidades. En este grupo se encuentran los elementos para los cuales la información varía para cada unidad de la serie, y en función de las características de cada elemento existirán:

Por un lado, elementos en los que varía todo el contenido, por ejemplo, la fecha de referencia del conjunto de datos o la extensión geográfica; y por otro, los elementos en los que varía parte del contenido, es decir, el título del conjunto de datos, el resumen descriptivo, etc. La finalidad de esta tarea es evitar el trabajo manual, repetitivo y abrumador que supone la creación de los metadatos de las unidades de una misma serie.

En el paso 4, se revisa que el borrador de metadato cumpla con el estándar de metadatos adoptado y otras recomendaciones en este dominio temático. El responsable del producto o del proceso de gestión, debe revisar el metadato para dar su conformidad sobre la información contenida en éste en términos de su consistencia y completitud mediante el uso de un validador.

El éxito en la localización de recursos informativos depende de la calidad de los metadatos geográficos en cuestión, la cual se determina principalmente por el grado de conformidad con el estándar de metadatos y la cantidad de elementos que satisfacen las necesidades de los usuarios (Rackham, 2015).

Finalmente, en el paso 5, se acude a una aplicación informática que permita crear los registros que contendrán los descriptores ordenados adecuadamente y generar el archivo de metadato digital. A lo largo de los últimos años se han venido desarrollando una gran cantidad de aplicaciones software que, ya sea como herramientas independientes o como plugins dentro de otras aplicaciones, facilitan en gran medida la creación y exportación de dichos metadatos en formatos estandarizados. El proceso de publicación, para exponerlo en una plataforma IDE, se lleva a cabo típicamente en catálogos de metadatos.

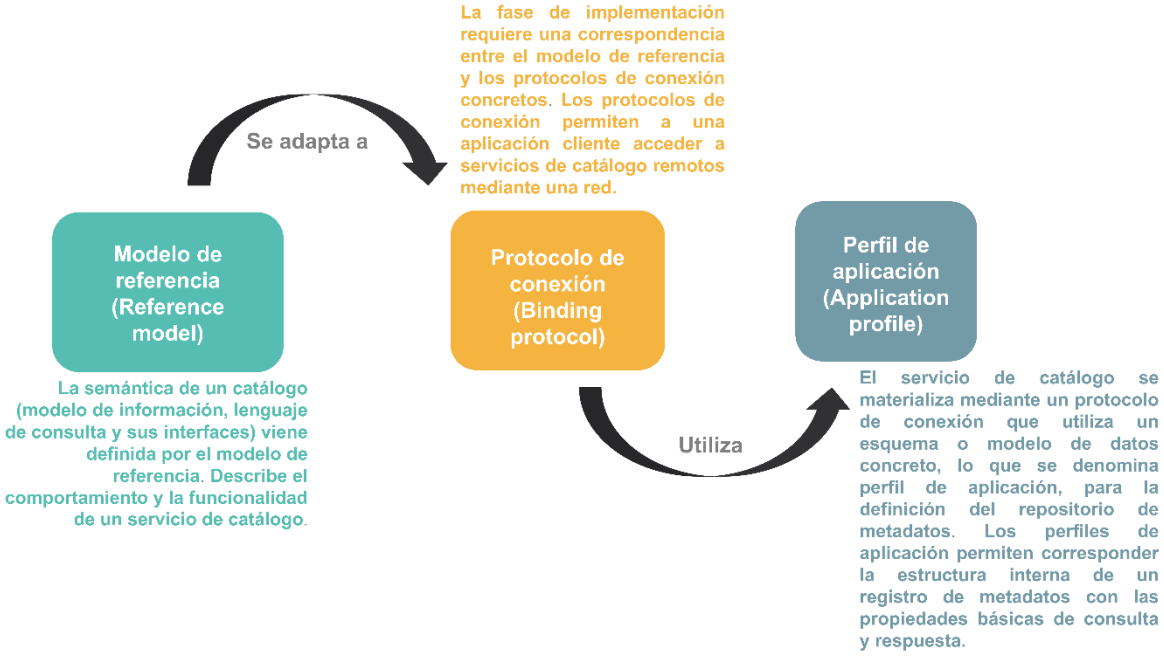


Figura 21. *Elementos básicos de un servicio de catálogo de metadatos. Fuente: Elaboración propia.*

5. Metadatos geográficos. Estándares y herramientas de gestión

Para extender el uso y entendimiento de los metadatos por diferentes actores al interior de una IDE corporativa, es preciso describir consistentemente un recurso geográfico mediante documentos bien definidos a partir de unos criterios comunes, razón por la cual se requiere un estándar de metadatos.

Evidentemente, existe una pluralidad de formas de construir un metadato, aspecto que está determinado por el hecho que éste es un documento de texto libre, por lo que cada organización le asigna el formato e información atendiendo a sus intereses concretos y particulares (Santos & Orozco, 2006).

Tal situación supone problemas que afectan no sólo el intercambio de recursos sino además su lectura e interpretación semántica de usuarios y aplicaciones que los consumen. A esto se puede incluir, el coste que implica la transformación entre formatos para homogenizar estos documentos, la perdida de información relevante y de consistencia lógica e integridad del documento que puede ocurrir en el proceso. Una solución a tales situaciones recae en la adopción de estándares en esta materia.

En este sentido, según afirma López & Pascual (2008), se han definido recomendaciones para la creación de metadatos, cuya finalidad principal es proporcionar una estructura “jerárquica y concreta” que permita describir exhaustivamente cada uno de los recursos a los que hacen referencia. Han sido creadas y aprobadas por organismos de normalización a partir de opiniones de expertos en este campo del saber. Estas recomendaciones, en forma de normas o directrices, proporcionan criterios para caracterizar los recursos con propiedad.

Algunos de los beneficios atribuidos a la implementación de estándares en la creación de metadatos los resumen Santos & Orozco (2006) en que: provee una terminología común y ofrece un conjunto de definiciones para la documentación rigorosa de estos recursos y favorecen la publicación de propiedades fundamentales implícitas y explícitas de los recursos en un formato conocido por los usuarios, promoviendo su interoperabilidad entre sistemas de información e infraestructuras de datos.

Existen diferentes propuestas de estandarización por organismos estatales e internacionales y desde diferentes ámbitos. Los más extendidos son el Dublín Core ISO15836:2003 y en el contexto geomático el ISO19115:2003 y sus perfiles derivados.

El World Wide Consortium ha mostrado un fuerte interés en el contexto de los metadatos al desarrollar la tecnología RDF (Resource Description Framework) y las especificaciones para la plataforma de selección de contenido en internet (PICS). RDF es un modelo de datos y sintaxis para los metadatos de los recursos web; permite la interoperabilidad de los metadatos por medio del diseño de mecanismos que soporten convenciones de semántica, sintaxis y estructura. PICS fue originalmente desarrollado por W3C para permitir la asociación de etiquetas (metadatos) con contenido en internet.

5.1. Dublin core

La alternativa de metadatos para la catalogación más utilizada a nivel mundial es el esquema Dublin Core (DC), creado y mantenido por la Dublin Core Metadata Initiative (DCMI). Específicamente, el esquema usado es el Dublin Core Metadata Element Set v1.1 o su versión más reducida, la norma ISO 15836:2003, conocida como “DC simple” recogiendo la información más relevante de descubrimiento en sólo 15 elementos (ver figura 21), propuestos para describir cualquier recurso disponible en internet.

Se presentan habitualmente divididos en tres grupos que indican la clase o alcance de la información incluida en ellos, y que responden, en cierta medida, a las expectativas que tiene el usuario cuando se enfrenta a la información en la red, estos son: **Contenido** (título, tema, descripción, fuente, idioma, relación y cobertura), **Propiedad intelectual** (autor, editor, colaborador, derechos) e **Instanciación** (fecha, tipo, formato e identificador).

El conjunto de elementos DC se ha convertido en una infraestructura operacional del desarrollo de la Web Semántica. Algunas de las fortalezas propuestas por Beltrán Fonollosa (2013) son: su simplicidad, independencia sintáctica, alto nivel de normalización formal (ANSI/NISOZ39.85-2001, ISO 15836-2003), crecimiento y evolución del estándar a través de una institución formal constituida en consorcio: la DCMI.

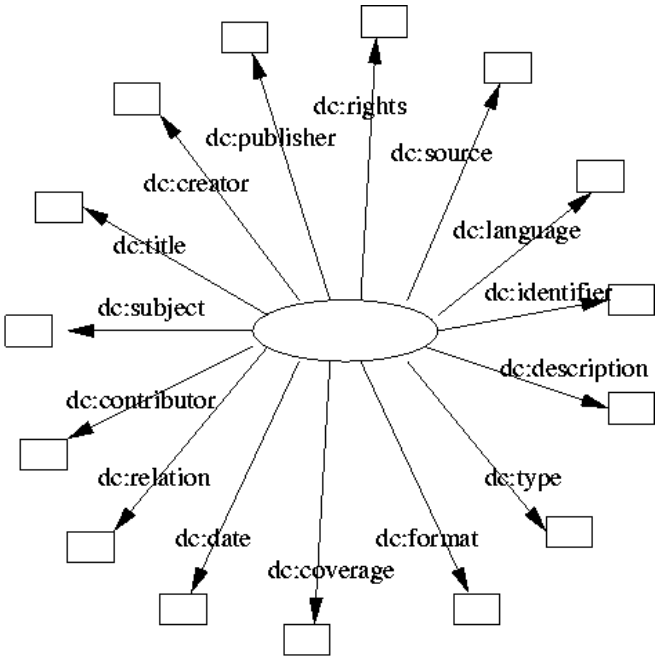


***Figura 22**. Esquema del Modelo Dublin Core.*[3]

5.2. Estándar CSDGM

El Content Standard for Digital Geospatial Metadata (CSDGM) del Comité Federal de Datos Geográficos (FGDC). Esta iniciativa fue la primera que obtuvo el rango de estándar y fue diseñada en los Estados Unidos por el FGDC y aprobada en 1994.

Es un estándar nacional para metadatos espaciales desarrollado para dar soporte a la construcción de la Infraestructura Nacional de Datos Espaciales de los Estados Unidos. Este estándar ha sido adoptado en otros países como por ejemplo Sudáfrica o Canadá (Anaya et al., 2002).

El objetivo de esta norma es proporcionar un conjunto común de terminología y definiciones para la documentación de datos geoespaciales digitales. Define la información requerida por los usuarios para determinar la disponibilidad del conjunto de datos, la propiedad para un uso determinado, el acceso al conjunto de datos y la forma de transferencia de los datos (Pearsall & Hogan, 2005).

[3] Recuperado de https://www.dublincore.org/specifications/dublin-core/dcq-rdf-xml/2001-11-30/

5.3. Estándar ISO para metadatos geográficos

Las normas ISO 19115: 2003 e ISO19119:2005, se corresponden con las primeras normas para definir los metadatos de la información geográfica. Actualmente son las más aceptadas por la comunidad geoespacial y las que mejor soportan la mayoría de las herramientas.

En la norma ISO 19115:2003 (y su corrección posterior ISO19115:2003 Cor1:2006) se definen los metadatos de los datos espaciales, mientras que en la norma ISO19119:2005 se definen los metadatos de los geoservicios. El propósito de esta última es ofrecer un marco de trabajo a los desarrolladores para crear aplicaciones que permitan a los usuarios acceder y procesar datos procedentes de diversas fuentes a través de interfaces de computación genéricas.

El objetivo de la norma ISO19115 es proporcionar un modelo y establecer un conjunto común de terminología, definiciones y procedimientos de aplicación para los metadatos. Mediante la definición de elementos de metadatos se va a poder describir información sobre la identificación, la extensión, la calidad, el modelo espacial y temporal, la referencia espacial y la distribución de datos geográficos digitales (López & Pascual, 2008).

Esta norma define: la secciones de metadatos obligatorios y condicionales, entidades de metadatos y elementos de metadatos; el conjunto mínimo de metadatos requeridos para soportar todo el rango de aplicaciones de metadatos (descubrimiento, determinación de la idoneidad, acceso, transferencia y utilización de datos); los elementos de metadatos opcionales para permitir una descripción normalizada más amplia de los datos geográficos; un método para crear extensiones de metadatos para adaptarse a necesidades especializadas (Danko, 2005).

Pese a que es una norma bastante robusta (409 elementos y 27 listas controladas), dispone de un núcleo que establece el número mínimo de elementos obligatorios de un metadato geográfico.

La norma grosso modo, está constituida por un conjunto de paquetes UML y cada paquete posee un diccionario de datos (ver figura 23). En la norma se preceptúa que todo perfil de metadatos que se cree debe incluir como conjunto mínimo de elementos los presentados en el núcleo.

La norma ISO19115-2:2009, se planteó dado que la ISO19115 carecía de elementos específicos para definir metadatos de datos ráster, de malla o productos fotogramétricos, extendiendo los metadatos de datos vectoriales para definir mejor este tipo de recursos.

Cabe señalar la evolución que han experimentado estas normas y por ende el cambio en su nomenclatura. De esta manera se tiene que, la norma ISO19115-1:2014 sustituye a la ISO19115:2003, unificando los metadatos de datos y de servicios (ISO19119:2005) y es aplicable a todo tipo de recursos (mapas, por ejemplo).

Así mismo, los elementos relativos a los procesos de calidad de la ISO19115 se han eliminado, ubicándolos en la norma ISO19157:2003. Adicionalmente se han creado nuevas listas controladas y se han incluido nuevos elementos en las enumeraciones y listas controladas.

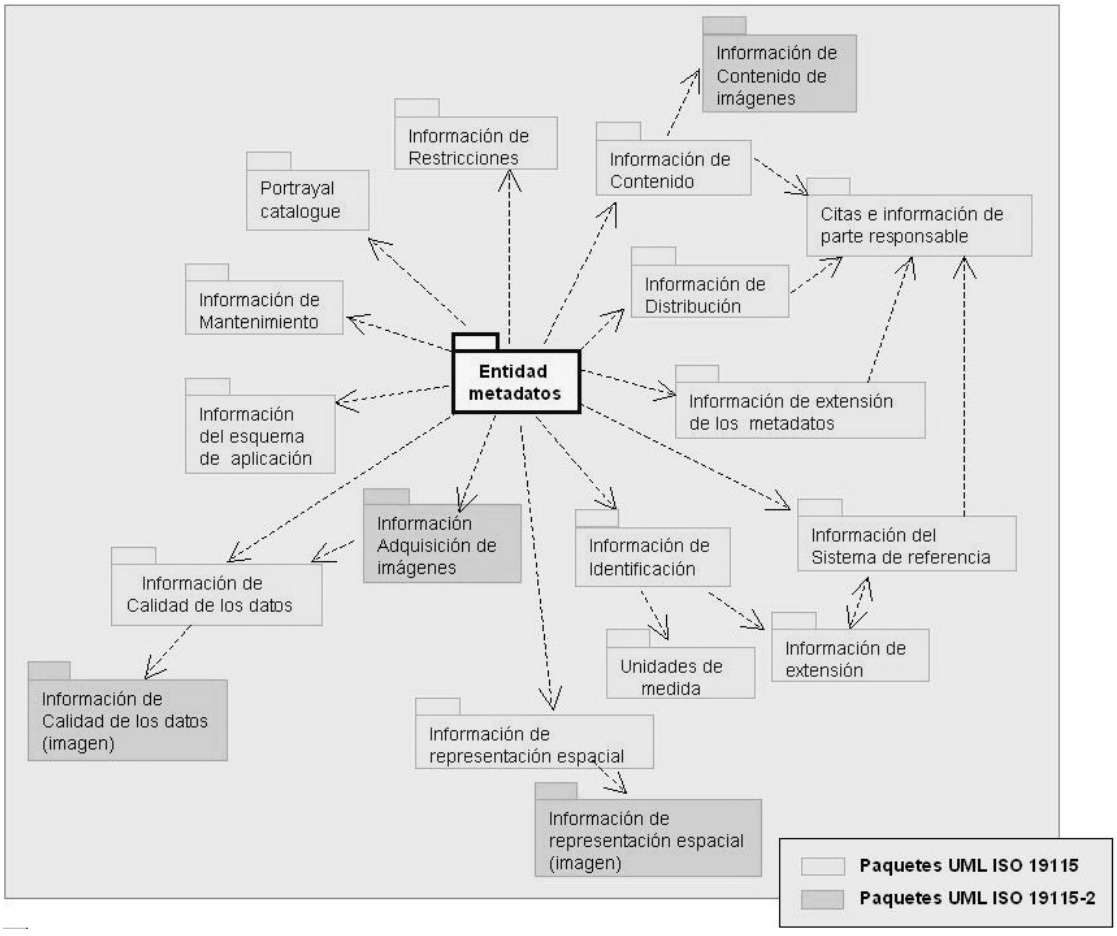


Figura 23. *Paquetes de metadatos de la norma ISO19115-2 y su relación con la norma ISO19115. Fuente: (López & Pascual, 2008).*

La ISO19139:2007 es el esquema de implementación de metadatos geográficos. Define la codificación basada en esquemas XML de implementación para las ISO19115 e ISO19119. Es decir, define el mecanismo de almacenamiento e intercambio a través de este lenguaje de marcado, de una forma estándar (se construyen documentos estructurados mediante el uso de sintaxis XML a partir de los modelos UML definidos en estas normas).

La codificación XML de la ISO19115-1:2014 viene definida en la ISO19115-3:2016, que sustituye a la norma ISO19139:2007. De más reciente aparición se tiene la norma ISO19139-2:2012, que define el esquema XML de implementación para metadatos de recursos ráster de la ISO19115-2:2009.

Cualquier herramienta informática diseñada para gestionar metadatos geográficos, permitirá la recuperación de estos documentos en XML, que sean validos respecto a dicho esquema.

5.4. Directiva INSPIRE

La Directiva 2007/2/CE de la Unión Europea INSPIRE, aprobada el 14 de marzo de 2007, establece que de conformidad con el artículo 5, apartado 1, de dicho organismo, los estados miembros de la UE garantizarán que se creen metadatos para los conjuntos y servicios de datos espaciales correspondientes a los temas enumerados en los anexos I, II y III, y que tales metadatos sean conservados.

El capítulo II de dicho documento específicamente establece los lineamientos y directrices para el establecimiento de metadatos. La directiva ordena a los estados su creación y se establece la información mínima requerida en los mismos (conformidad, condiciones que rigen el acceso y utilización, la calidad y validez, los custodios y las limitaciones de acceso público a los que haya lugar), garantizándose completitud y calidad.

La iniciativa cuenta con 2 instrumentos en este dominio: el reglamento en materia de metadatos (y modificaciones posteriores), para el que actualmente se encuentra vigente el Reglamento (UE) No. 1311/2014 de la Comisión, del 10 de diciembre de 2014, por el que se modifica el Reglamento (CE) No. 976/2009 en lo que respecta a la definición de elementos de metadatos de INSPIRE.

De igual modo, se cuenta con las directrices técnicas que establecen los requisitos a un nivel más abstracto para la creación y el mantenimiento de estos metadatos. El más reciente corresponde a las directrices técnicas de las reglas de implementación de metadatos de INSPIRE, aspectos destacados de cambios clave entre la versión 1.2 de junio de 2010 y la versión 1.3 de octubre de 2013 y las normas de implementación de metadatos de INSPIRE: Directrices técnicas basadas en ISO 19115 y en ISO 19119.

5.5. Estándar OGC para el servicio web de catálogo CSW

Los servicios de catálogo son actualmente el principal método para la publicación de contenido geográfico al interior de una Iniciativa IDE sin importar el nivel de implementación. El servicio de catálogo de metadatos conecta proveedores y consumidores. Los proveedores publican los metadatos en el catálogo, mientras que los consumidores buscan metadatos mediante el catálogo.

El servicio de catálogo de metadatos se convierte así en una pieza básica en una IDE. Por este motivo, la OGC ha definido la especificación de servicios de catálogo (abreviado como CSW de Catalogue Service for Web) para la búsqueda y recuperación de metadatos de datos y servicios geográficos en un sistema distribuido. Las aplicaciones de catálogo normalmente implementan esta especificación estándar.

El CSW es un estándar orientado a facilitar la publicación, búsqueda y utilización de datos espaciales, servicios y otros recursos a partir de información descriptiva almacenada en metadatos que son registrados en un catálogo. Por lo tanto, el estándar especifica un patrón de diseño de interfaces para la publicación y descubrimiento de éstos.

En el acápite anterior se presentó esquemáticamente el funcionamiento genérico de un catálogo de metadatos (ver figura 21), en el que se ilustran la configuración de tres componentes claves, a saber: **el modelo de referencia, los protocolos de conexión y el perfil de aplicación**.

A continuación, se describirá brevemente su funcionamiento de acuerdo a la revisión de la especificación hecha por Nebert et al., (2016) y la publicación de la OGC “Catalogue Services 3.0 - General Model”.

Considerando el **modelo de referencia**, los elementos que lo constituyen son: (a) el modelo de información (propiedades básicas de consulta y de respuesta), (b) el lenguaje de consulta y (c) la interfaz del servicio.

(a) **El modelo de información**, el cual define un conjunto de términos, o vocabulario básico, para las peticiones y las respuestas de búsqueda que cualquier servicio de catálogo debe soportar.

Cualquier servicio de catálogo puede atender peticiones de búsqueda basadas en las propiedades básicas de consulta (Subject/asunto, Title/título, Abstract/resumen, Format/formato digital, Identifier/identificador del registro de metadatos en el repositorio, Modified/fecha de creación o actualización, Type/categoria del contenido, BoundingBox/Ventana espacial, CRS/sistema de referencia espacial, Association/relación con otros recursos).

Cualquier servicio de catálogo puede devolver los resultados de una búsqueda mediante las propiedades básicas de respuesta (Title/título, Creator/proveedor, Subject/asunto o palabras claves, Description/resumen, Publisher/distribuidor, Contributor/contribuciones, Date/fecha de creación del recurso, Type/categoría del contenido, Format/formato digital, Identifier/identificador del registro de metadatos, Source/fuente, Language/idioma del registro de metadatos, Coverage/ventana espacial, Relation/relación y Rights/derechos de autor).

(b) **El lenguaje de consulta**. La especificación CSW también ofrece un conjunto de operaciones de consulta básica (query language). Algunas operaciones son por ejemplo los predicados booleanos para comparaciones lógicas (mayor que, menor que, igual, etc.), operaciones sobre cadenas de textos y, sobre todo, operaciones espaciales o topológicas.

(c) **Mediante la interfaz del servicio** CSW es posible el descubrimiento, el acceso, el mantenimiento y la organización del catálogo por medio de operaciones. La especificación define cada una de estas operaciones en el ámbito de signatura funcional: proporciona el nombre de la operación y su semántica, así como el conjunto de parámetros de entrada y salida junto con el significado de cada uno de ellos.

En total, define tres clases de operaciones (ver figura 24):

OGC_Service, permiten conocer los metadatos del propio servicio, así como de los registros contenidos a través de su identificador. **Discovery**, ofrecen información sobre la ejecución del catálogo, así como la posibilidad de recuperar información sobre los registros). **Manager**, permiten editar los registros del catálogo.

Las operaciones definidas en este estándar son: GetCapabilities, GetRecordByID (pertenecen al grupo OGC_Service), GetRecords, GetDomain (Discovery), Transaction, Harvest y UnHarvest (Manager). De las siete operaciones 4 son obligatorias (GetCapabilities, DescribeRecord, GetRecords, GetRecordByID) y 3 son opcionales (GetDomain, HarvestRecords, Transaction).

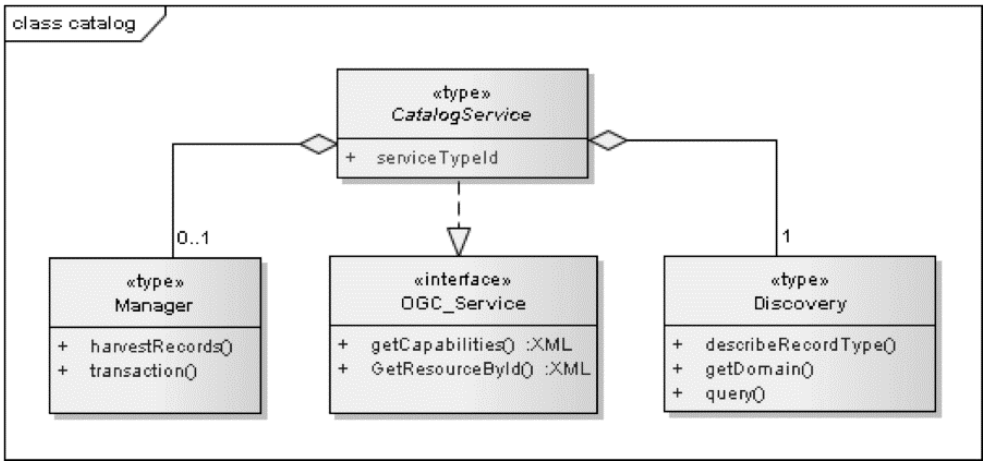


***Figura 24**. Diagrama de clases en UML para representar la arquitectura de un servicio de catálogo CSW[4].*

En cuanto a los **protocolos de conexión** para acceder remotamente a un servicio de catálogo y consultar sus metadatos, el más utilizado corresponde al Hypertext Transfer Protocol (HTTP). La especificación CSW admite tres formas diferentes de empaquetar los mensajes de petición y respuesta para el protocolo HTTP:

GET-KVP (key-value pairs). El empaquetamiento GET-KVP representa una petición HTTP GET contra el servicio de catálogo, que codifica los parámetros de la petición como una lista parámetro-valor en la propia URL, después del signo de interrogación (?). Cada

[4] Recuperado de http://docs.opengeospatial.org/is/12-168r6/12-168r6.html

par parámetro-valor se separa mediante el símbolo de ampersand (&). La respuesta del servicio siempre será un documento XML.

XML-POST. El empaquetamiento XML-POST realiza una petición HTTP POST contra el catálogo, pero esta vez los parámetros de la petición van empaquetados en un documento XML.

Este modo de operar es idéntico al del procesamiento de formularios web. Al presionar el botón de enviar formulario, el navegador realiza una petición HTTP POST contra la URL del servicio encargado de procesar la petición. Los datos del formulario no van codificados en la propia URL del servicio, sino empaquetados en un mensaje aparte. La respuesta del servicio siempre será un documento XML.

SOAP (simple object access protocol). Es una de las especificaciones básicas para los servicios web. Añade explícitamente un envoltorio adicional a los mensajes XML de la petición y respuesta (GET-KVP y XML-POST simplemente hacen uso de los mecanismos propios del protocolo HTTP).

La función de este envoltorio es la de servir de contenedor para otras funcionalidades adicionales, como seguridad y cifrado, que no son posibles con la funcionalidad básica que ofrece HTTP. Los mensajes SOAP de petición y respuesta siempre se realizan mediante peticiones HTTP POST.

Finalmente, sobre los **perfiles de aplicación** de un catálogo, la especificación no establece un esquema o el modelo de información interno del repositorio de metadatos. Sin embargo, existen dos posibilidades sobre esta cuestión: el **modelo ebRIM** propuesto por OASIS (Organization for the Advancement of Structured Information Standards) y el **modelo ISO19115**.

Para ambos casos, la especificación CSW define un perfil de aplicación concreto. Los perfiles de aplicación permiten corresponder la estructura interna de un registro de metadatos con las propiedades básicas de consulta y respuesta.

Como los modelos ebRIM e ISO tienen estructuras distintas, los perfiles de aplicación indican la correspondencia entre cada una de estas propiedades básicas con un determinado elemento XML del registro de metadatos interno.

De esta manera, un usuario puede realizar una consulta para recuperar los registros de metadatos cuyo Title (propiedad básica) contenga la palabra geodesia, sin necesidad de conocer el modelo interno de metadatos utilizado por el catálogo y el descriptor interno utilizado para almacenar el valor del título.

En virtud de los aspectos y consideraciones presentados en este apartado del TFM, la figura 25 presenta la arquitectura de una IDE corporativa requerida para la gestión de metadatos geográficos.

Es de destacar en la figura 25 que los metadatos se almacenan en un repositorio o base de datos. Cada registro de metadatos se estructura de acuerdo con el modelo ebRIM o ISO 19115.

Del lado del servidor existen herramientas de gestión que facilitan la inserción, actualización o eliminado provisional de metadatos en el repositorio antes de su publicación en un catálogo mediante la especificación CSW.

Un servicio de catálogo tiene perfiles de aplicación, en función del esquema de metadatos elegido para el repositorio. Las aplicaciones cliente utilizan normalmente HTTP como protocolo de conexión para interactuar con el catálogo de metadatos de la IDE.

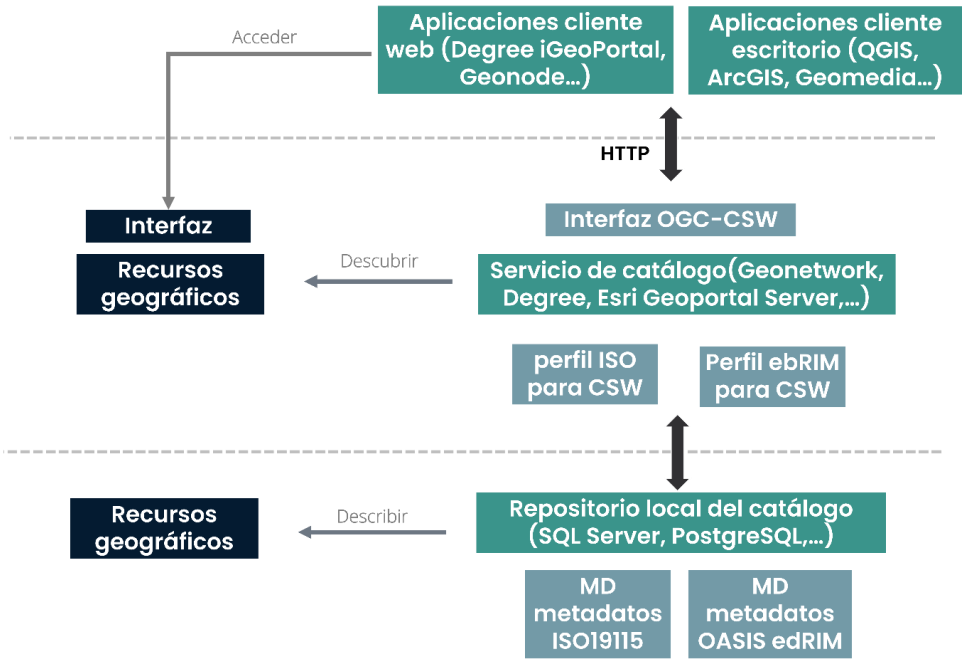


Figura 25. *Arquitectura IDE para la gestión de metadatos geográficos. Fuente: Elaboración propia.*

5.6. Herramientas para la gestión de metadatos geográficos

Existen diferentes entornos informáticos para la creación y visualización de metadatos en el contexto geomático. Dependiendo del grado de automatización y de los requerimientos de participación por parte de los usuarios en el proceso de creación de los metadatos, se distingue entre generadores y editores.

De hecho, la mayoría de estas aplicaciones son presentadas como editores de metadatos y, algunas de ellas, tienen capacidades para generar de forma automática algunos metadatos concretos. Existen diversas opciones en el mercado, de entre las que se puede destacar:

- **CatMDEdit**[5]. Es una herramienta editora de metadatos que facilita la documentación de recursos. Se trata de una iniciativa del Instituto Geográfico Nacional de España (IGN), que es fruto de la colaboración científico-técnica entre el IGN y el Grupo de Sistemas Avanzados de Información (IAAA) de la Universidad de Zaragoza con el apoyo técnico de GeoSpatiumLab (GSL).

La herramienta ha sido implementada en Java y cuenta con las siguientes características: El almacenamiento de los registros de metadatos se gestiona directamente a través del sistema de ficheros.

Posee interfaces de edición adaptadas a diferentes perfiles de metadatos con base al estándar ISO19115/Cor12006, ISO/TS 19139 codificación XML e ISO19119, Dublin Core (ISO 15836).

Generación automática de metadatos para algunos formatos de archivos de datos como el Shapefile y los GeoTiff.

Permite el intercambio de registros de metadatos según diferentes estándares en XML y RDF. Soporta la utilización de tesauros.

- **Metalite**[6]**.** Es un software de distribución gratuita cuyo proveedor es el USGS, EROS data Center. Programada sobre visual basic, está preparada para ejecutarse sobre

[5] https://catmdedit.sourceforge.io/
[6] http://edcnts11.cr.usgs.gov/MetaLite

sistemas Windows y, entre sus características se destaca el intercambio de metadatos en formato HTML, TXT y XML.

Los metadatos son almacenados en bases de datos Access, la cual podrá ser utilizado por cada usuario desde su ordenador, para después unirse a una base de datos central. Pese a que la aplicación está descontinuada, el portal web del FGDC dispone de un editor de metadatos en línea (OME) de USGS Core Sciences (CSAS).

- **IME (ISO METADATA EDITOR)**[7]**.** Creado por INTA (Instituto Nacional de Técnica Aeroespacial) del ministerio de defensa español. Es una herramienta que corre sobre una máquina virtual de Java, JRE 5.0 o versiones superiores. Su última actualización corresponde a la 4.1 del 2007 que incluye ficheros XSD del esquema ISO19139.
- **Geonetwork**[8]. Es una herramienta de código abierto desarrollado por la FAO-UN, WFP-UN y UNEP. Se trata de un potente catálogo web de metadatos geográficos, basado en el estándar CSW de la OGC. Provee un editor de metadatos en línea, un portal de búsqueda, y permite el registro y repositorio de los metadatos. Como es de esperarse, soporta ficheros de metadatos creados conforme a los esquemas ISO, CSDGM y Dublin Core. De igual modo, soporta la utilización de tesauros.
- **Degree**[9]**.** Es una aplicación de catálogo de código abierto construida sobre java ejecutable en todas las plataformas compatibles con el entorno de tiempo de ejecución OpenJDK. Implementa el estándar CSW 2.0.2. Totalmente transaccional. Admite solicitudes KVP, XML y SOAP. Admite los estándares de metadatos: Dublin Core, ISO e INSPIRE.
- **Esri Geoportal Server**[10]**.** Es un producto gratuito de código abierto que permite administrar y publicar metadatos de recursos geoespaciales para permitir que los usuarios descubran y se conecten a éstos. Esri Geoportal Server emplea un servicio de catálogo de geoportal CS-W 2.0.2. Admite y proporciona plantillas para varios perfiles de metadatos comunes, incluidos FGDC, ISO19115, INSPIRE y Dublin Core.

[7] http://crepadweb.cec.inta.es/metadata/index.htm
[8] https://geonetwork-opensource.org/
[9] https://www.deegree.org/about-deegree/
[10] https://www.esri.com/en-us/arcgis/products/geoportal-server/overview

- La mayoría de las aplicaciones de software de procesamiento de imágenes (ERDAS Apolo) y GIS de escritorio proporcionan editores y validadores de metadatos como componentes de sus soluciones. Aplicaciones como ArcGIS (**ArcCatalog**), Geomedia (**Geomedia Catalog**) o QGIS (Complemento **QSphere**[11]) disponen de estas utilidades para gestionar metadatos.
- A pesar de que los editores de metadatos guían al usuario en el cumplimiento de los requisitos derivados de los estándares de metadatos y normativas aplicables, existen herramientas adicionales que facilitan el control de calidad final de los metadatos producidos. Se trata de herramientas para comprobar la corrección de su estructura (formato o estándar de implementación) y/o evaluar la conformidad con respecto a requisitos específicos de los estándares y/o de las normativas vigentes aplicables, como INSPIRE[12].
- En la revisión comercial, no se encontró aplicaciones de gestión que incorporen los códigos QR como mecanismo de almacenamiento y representación de metadatos geográficos.

6. Modelo de gestión de metadatos geográficos codificados con tecnología QR

6.1. Códigos QR

Un código QR (quick response barcode, o código de barras de respuesta rápida) es una tecnología para el almacenamiento de información en una matriz de puntos o un código de barras bidimensional.

Esta manera de codificar la información fue creada por la compañía japonesa Denso Wave subsidiaria de Toyota en 1994. Su intención fue facilitar el acceso a información técnica de manera rápida y mejorar la experiencia del usuario (Sutheebanjard & Premchaiswadi, 2010).

[11] https://qgis.projets.developpement-durable.gouv.fr/projects/qsphere
[12] https://inspire.ec.europa.eu/Tools/Metadata/6541

A diferencia de los códigos de barras convencionales, la información está codificada en un patrón cuadrado de módulos negros (puntos/píxeles cuadrados), dispuestos sobre un fondo blanco (ver figura 26). El código permite almacenar gran cantidad de información alfanumérica (metadatos, por ejemplo) tanto en código ASCII como binaria.

Para leer o interpretar un código QR es necesario un dispositivo inteligente con acceso a internet, cámara integrada y un lector compatible. El éxito en la implementación de este tipo de códigos, en todo tipo de aplicaciones, obedece precisamente a su estándar abierto y a la facilidad de decodificación mediante los teléfonos móviles, los cuales operan como lectores.

Este tipo de códigos se caracterizan por tener una alta velocidad en su decodificación, gran capacidad de almacenamiento de información y codificación de conjuntos de caracteres especiales. De hecho, es capaz de almacenar información vertical y horizontalmente a diferencia de los tradicionales códigos de barras, que sólo lo hacen en una dirección, como se ilustra a continuación:

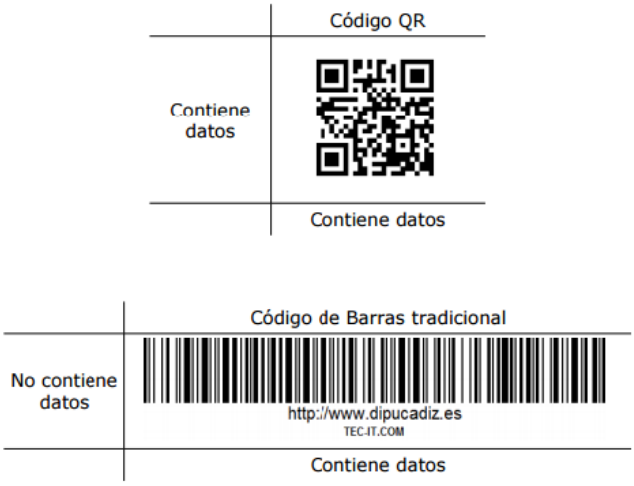


Figura 26. *Comparación código QR y código de barras tradicional*[13]*.*

Los datos contenidos en estos códigos QR pueden ser de tipo numérico, alfanumérico, binario (ISO-8859-1) o Kanji/Kana (Shift JIS X 0208). Para el sistema gestor de metadatos geográficos, se usarán códigos de tipo alfanumérico, mediante los cuales se puede llegar a almacenar hasta 4.296 caracteres o 2.953 bytes del sistema binario.

[13] Recuperado de http://www.dipucadiz.es

Así mismo, para este TFM se utilizará un generador propio de códigos QR, implementado a partir de la librería qrcode de python.

***Figura 27**. Estructura de un código QR*[14]

En relación a la ilustración anterior, el elemento más distintivo del código QR son los bloques cuadrados utilizados para orientar el código cuando es escaneado por el lector. Esto permite escanear el código en cualquier orientación, incluso al revés, y transmitir el mensaje correcto.

Estos cuatro bloques (tres en las esquinas y uno incrustado en la parte inferior derecha) aparecen en cada código y permiten decodificarlo correctamente en cualquier orientación.

Un código QR está compuesto por 5 partes de información diferenciadas:

- Los símbolos o marcadores de posición: se corresponden con los 3 cuadros ubicados en las esquinas que, aunque no contienen información en sí mismos, sirven para que el lector del móvil detecte el código QR en la dirección correcta.
- El símbolo de alineamiento o patrón de alineación (timing) se utiliza para detectar la posición cuando se ha producido un desplazamiento de módulos debido a una distorsión. Permite que se decodifique en 360°.

[14] Recuperado de https://serviciostecnicosmovil.com/codigos-qr-que-son-y-quien-los-invento/

- El margen, importante para la correcta lectura del código QR (identifica el código del entorno). Debe tener un grosor mínimo de 4 módulos, mientras que en el microcódigo QR puede ser de 2.
- Las líneas de dimensión: marcan el tamaño de los módulos que forman el código.
- La información variable: permite verificar el formato o versión del QR, los datos relacionados con la indexación de la matriz de cuadrados y la corrección de errores aplicada en la generación.
- La máscara de datos o región de codificación: una vez que se conocen los datos anteriores es necesario conocer el ID de la máscara de protección del código QR que se obtiene de la información de formato y los bits contenidos. Con el valor de la máscara se determinan los módulos del código QR y se obtiene la matriz en binario. Después de este proceso, el lector de QR traduce de binario a caracteres y da acceso a la información que se había codificado.

Los bits se enmascaran para asegurar que los datos se expresan de forma eficiente. Esta máscara se representa como una cadena binaria, que se combina matemáticamente con el mensaje para producir el código QR. Este proceso no encripta el mensaje, y la clave necesaria para decodificar el enmascaramiento se incluye con el código QR.

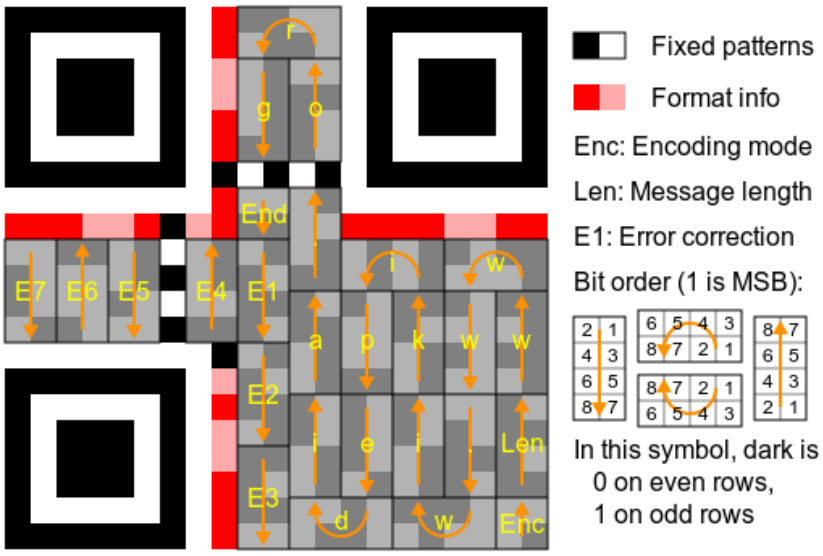


Figura 28. *Lectura de códigos QR*[15]

[15] Recuperado de https://vidatecno.net/como-funcionan-los-codigos-qr/

El código QR se lee desde la esquina inferior derecha. Los píxeles se leen como grupos de 8, con un byte por cada 8 píxeles. Dependiendo de cómo se codifique el código QR, estos patrones de bits se enlazarán a diferentes caracteres.

Mientras que ASCII es probablemente la codificación más común, la especificación también incluye opciones para números, kanji y otras codificaciones. El patrón de cuatro bits en la parte inferior derecha del código QR (marcado Enc en la figura 28) determina cómo se decodificarán los bits.

A diferencia de los códigos de barras, los códigos QR también incluyen un componente de comprobación de errores. Esto significa que el código QR puede ser decodificado con éxito incluso si partes del código son ilegibles.

El último estándar que rige la codificación de los códigos QR es el ISO/IEC 18004:2006. Mediante este sistema de codificación, se permite la corrección de errores gracias al algoritmo Reed-Solomon, que puede llegar hasta un nivel de corrección del 30%. Existen 4 niveles, L-bajo puede restaurar 7% de caracteres, M-medio con el 15%, Q-cuartil con el 25% y H-alto con el 30%.

Este algoritmo divide la información en grupos de bits, llamados símbolos y efectúa la detección mediante polinomios (Hernando & Macías, 2013). En otras palabras, el algoritmo crea una copia de seguridad del código QR para restaurarlo y volver a él en caso de errores de lectura.

El tamaño más utilizado de QR está entre 29 x 29 y 33 x 33, que puede contener unos 50 caracteres ASCII. Todos los tamaños son interoperables con los dispositivos móviles actuales.

6.2. Análisis de requerimientos del sistema

En esta sección se expondrán los requerimientos funcionales y no funcionales para el desarrollo del sistema de gestión de metadatos geográficos mediante tecnología QR denominado *Metadata*.

En este sentido, el proyecto se plantea en dos partes. La primera está asociada a la gestión de metadatos mediante una aplicación de escritorio, que alimenta un motor de base

de datos, y sobre el cual se puedan ejecutar las operaciones básicas de gestión (crear, editar, consultar y eliminar metadatos). Desde esta interfaz debe brindarse la posibilidad de descargar el código para luego ser embebido en el layout del mapa digital a compartir.

El segundo componente del sistema corresponde al despliegue y lectura del metadato desde una aplicación móvil. Para este escenario, se plantea el desarrollo de una aplicación sobre el sistema operativo Android que permita: decodificar el código QR incrustado en el mapa, muestre en pantalla del dispositivo el metadato en cuestión, y se ofrezca la posibilidad de descargar el metadato en formato XML y en PDF.

Dicho esto, se describen las necesidades de gestión de metadatos geográficos para mapa digitales que serán traducidas a requerimientos del sistema. Estos son:

Tabla 3. *Requerimientos funcionales del sistema*

ITEM	DESCRIPCIÓN
RF-Metadata-01	El usuario gestor requiere de autenticación para acceder a la aplicación y utilizar las operaciones de gestión de metadatos geográficos.
RF-Metadata-02	Se requiere de autentificación para la administración de la base de datos del sistema.
RF-Metadata-03	La aplicación móvil no requiere de autentificación para acceder a las funcionalidades del sistema.
RF-Metadata-04	La aplicación desktop del sistema permitirá crear metadatos geográficos del mapa a partir de un formulario de inserción y generar automáticamente el código QR asociado.
RF-Metadata-05	La aplicación desktop del sistema permitirá la consulta y visualización del resultado en pantalla de todos registros asociados a un metadato que satisfaga la búsqueda.

RF-Metadata-06	El usuario gestor podrá aplicar filtros en la interfaz de consulta de la aplicación desktop a partir de palabras claves y por atributos: título del mapa, categoría temática, fecha de creación del mapa, entre otros.
RF-Metadata-07	En el formulario de inserción, el usuario gestor puede adjuntar archivos de imágenes como muestra gráfica o versión miniatura del mapa.
RF-Metadata-08	Los campos asociados al identificador del metadato y su fecha de creación deben generarse de modo automático en la base de datos.
RF-Metadata-09	La aplicación desktop permitirá la modificación y/o edición de la información alfanumérica consignada en los registros de un metadato siempre que se satisfaga un parámetro de consulta requerido.
RF-Metadata-11	El usuario gestor, una vez diligenciado el formulario de inserción, podrá descargar el código QR del metadato geográfico.
RF-Metadata-12	El usuario gestor, una vez aplicado un filtro de consulta, podrá además de acceder a la información del metadato, visualizar el código QR y el archivo de imagen adjunto asociados. Así mismo, podrá eliminar el registro de metadatos en la base de datos.
RF-Metadata-13	La eliminación de metadatos podrá efectuarse de modo individual uno a uno o mediante una selección en pantalla. Para selección de más de un registro, la aplicación desktop permitirá limpiar la selección.
RF-Metadata-13	El usuario podrá recuperar la información del metadato del mapa escaneando el código QR generado desde la aplicación desktop del sistema.
RF-Metadata-14	La lectura del código QR se realizará mediante un dispositivo móvil de forma rápida, y el contenido del metadato se mostrará en la pantalla.
RF-Metadata-15	La aplicación móvil permitirá la descarga y/o exportación del metadato geográfico en formato PDF o XML.

RF-Metadata-16	Una vez el metadato geográfico sea descargado y/o exportado, la aplicación móvil debe limpiar la interfaz de visualización.

Fuente: Elaboración propia.

En cuanto a las características que restringen la operación o funcionamiento del sistema gestor de metadatos geográficos, conocidos como requerimientos no funcionales, se atienden desde 3 frentes: los requerimientos del producto (criterios de calidad de software ISO9126-3 y métricas de experiencia de usuario sobre el producto ISO-9241-11), requerimientos organizacionales (operacionales, de desarrollo) y requerimientos externos (regulatorios, éticos y legislativos).

Tabla 4. *Requerimientos no funcionales del sistema*

ITEM	DESCRIPCIÓN
RNF-Metadata-01	El sistema debe ser accedido por los usuarios mediante una dirección HTTP, sin necesidad de utilizar plugins o software emulador de web.
RNF-Metadata-02	Diseñar la capa de presentación considerando aspectos de usabilidad e integración entre ambas soluciones (desktop y móvil). El sistema debe ser de fácil entendimiento de tal forma que se reduzcan los tiempos de entrenamiento, soporte y prueba por parte del usuario.
RNF-Metadata-03	Ser autoajustable a cualquier tamaño y resolución de pantalla del usuario: La aplicación debe desplegarse sin deficiencias ni distorsiones de diseño en cualquier tamaño y resolución de pantalla del usuario.
RNF-Metadata-04	Ser intuitivo para el usuario y mostrar la información de manera dinámica, ágil y estética: El manejo del aplicativo debe ser intuitivo para el usuario, utilizar imágenes optimizadas y componentes de diseño que permitan mostrar la información de manera dinámica, ágil y estética.

RNF-Metadata-05	El estándar de diseño para la interfaz de usuario debe corresponder a los lineamientos del manual de marca de la organización. Así mismo, los parámetros de diseño deben estar en línea a las disposiciones contempladas en las guías del WCAG2.0 del W3C.
RNF-Metadata-06	Garantizar la lógica del flujo de eventos asociado a cada uno de los elementos de la interfaz de usuario.
RNF-Metadata-07	Debe ser implementada en idioma español: Todo el contenido de texto del sistema de gestión de metadatos geográficos debe ser implementado en este idioma.
RNF-Metadata-08	En caso de presentarse alguna excepción, el sistema debe mostrar un mensaje de error que muestra la descripción del evento
RNF-Metadata-09	Tolerancia a fallas: El sistema debe presentar un nivel bajo de incidencias mientras esté en funcionamiento.
RNF-Metadata-10	El sistema gestor de metadatos geográficos debe ser escalable, es decir, permitir el crecimiento de información, nuevas capacidades y funcionalidades.
RNF-Metadata-11	En cuanto a disponibilidad, se debe permitir que el sistema pueda recibir múltiples peticiones de ingreso, consulta y descarga de metadatos con un porcentaje de fallas inferior al 70%
RNF-Metadata-12	El sistema debe soportar una concurrencia de al menos el 50% de los funcionarios de la organización sobre el 100%
RNF-Metadata-13	La navegación y disposición de funcionalidades debe ser sencillo, intuitivo de fácil acceso facilitando el uso del sistema por usuarios no técnicos en manejo de herramientas geomáticas.
RNF-Metadata-14	En cuanto a la portabilidad de la aplicación móvil, su ejecución está diseñada para sistemas operativo Android en cualquier versión.
RNF-Metadata-15	El sistema debe ser construido para optimizar la eficiencia en términos del espacio y performance de las aplicaciones.

RNF-Metadata-16	La gestión de metadatos geográficos al interior del sistema debe atender a estándares internacionales en esta materia: ISO19115, ISO19139, por citar algunos, y a las disposiciones legales y normativas que rijan su implementación: Directiva INSPIRE, entre otros.
RNF-Metadata-17	Se deben considerar ambientes de desarrollo open source. En este sentido se propone el SGDB PostgreSQL, La IDE de desarrollo Spyder de Python, la herramienta QT Designer y la IDE de desarrollo Android Studio.

Fuente: Elaboración propia.

La tabla 5 expone la definición de los actores del sistema, en la que se presenta la descripción de su rol y sus capacidades.

Tabla 5*. Definición de actores del sistema*

ROL	Administrador	AC-Metadata-01
DESCRIPCIÓN	Representa el funcionario activo de la organización responsable del manejo, mantenimiento, desempeño y gestión de la base de datos que soporta el sistema gestor de metadatos geográficos.	
CAPACIDADES	Realizar procedimientos de copia de seguridad y recuperación de datos. Administrar cuentas de usuarios del sistema, la infraestructura sobre la que opera el sistema y supervisar el rendimiento de las aplicaciones.	
ROL	Gestor	AC-Metadata-02
DESCRIPCIÓN	Representa el funcionario activo de la organización encargado de la creación, actualización y depuración de metadatos geográficos asociados a mapas digitales que se generan al interior de su dependencia.	
CAPACIDADES	Acceder, crear, modificar, consultar y eliminar metadatos geográficos en la aplicación de gestión.	

ROL	Usuario	AC-Metadata-03
DESCRIPCIÓN	Representa el funcionario activo de la organización interesado en acceder y consultar información de metadatos geográficos de mapas digitales mediante un dispositivo móvil.	
CAPACIDADES	Escanear código QR, visualizar metadato y descargar en formatos PDF y XML.	

Fuente: Elaboración propia.

6.3. Arquitectura del sistema

Considerando los requerimientos educidos en el apartado anterior, se procede a modelar la arquitectura (comportamental y lógica) del sistema de gestión de metadatos geográficos para mapas digitales a partir del lenguaje de especificación UML.

El primer modelo corresponde al diagrama de casos de uso (figura 29), el cual es utilizado en este TFM para representar el comportamiento del sistema a partir de sus funcionalidades (operaciones que se llevan a cabo en las aplicaciones) y los actores que interactúan con el mismo.

Para especificar dicho comportamiento, se representa los actores, las relaciones y las operaciones basadas en los requerimientos funcionales identificados previamente.

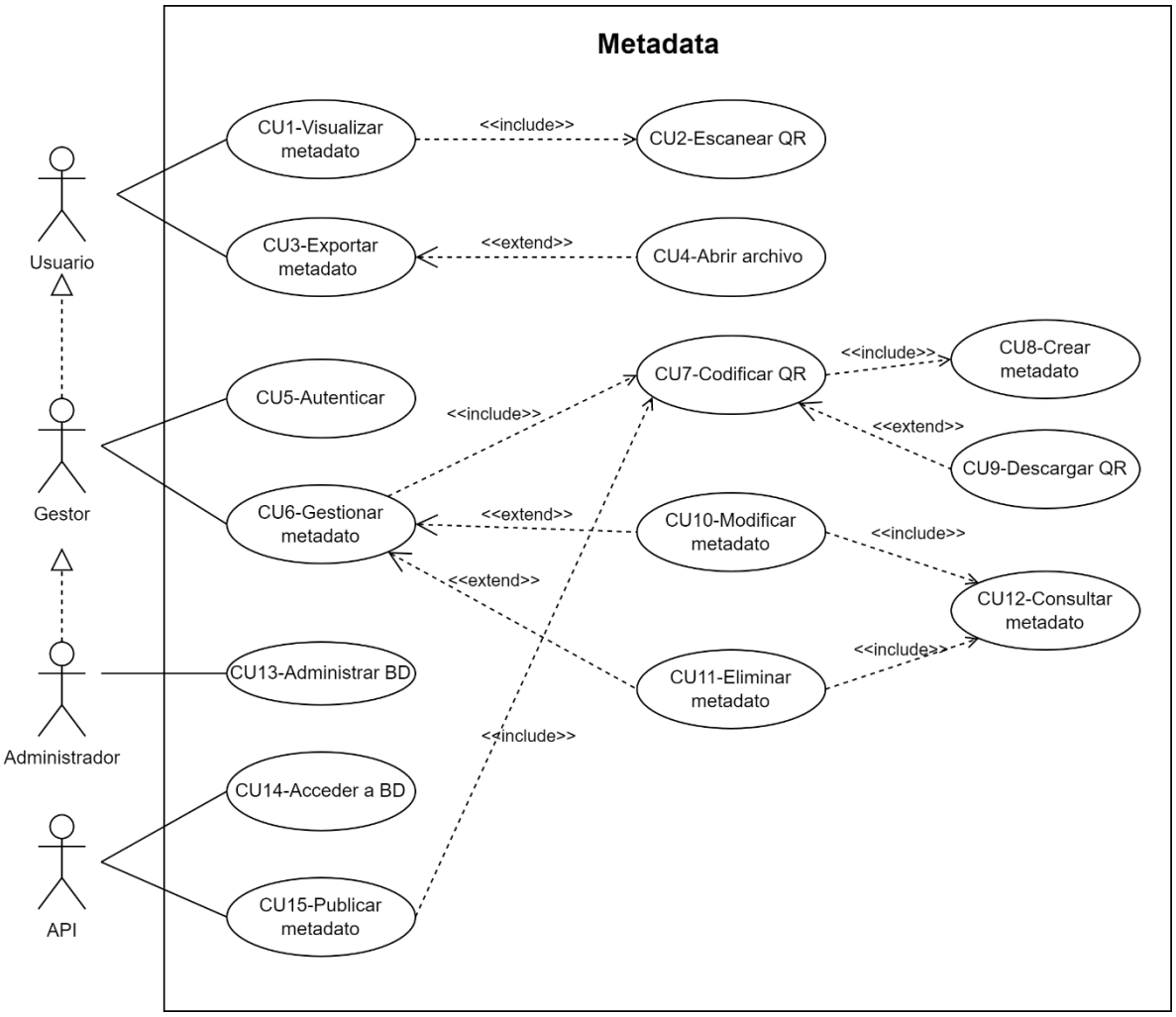


***Figura 29**. Diagrama de casos de uso. Fuente: Elaboración propia.*

En relación al diagrama anterior, la propiedad de mayor interés de los actores, más allá de su identidad, es la relación que estos guardan con los distintos casos de uso del sistema, entendiendo por caso de uso a aquellas descripciones funcionales del sistema.

Así pues, se tiene el caso de uso Visualizar metadato CU1, quien posee una relación de inclusión (dependencia obligatoria de otra función, no se ejecuta la tarea sin previamente ejecutarse la que se ha incluido) indicada por el estereotipo <<include>> con el caso de uso Escanear código QR CU2.

El caso de uso Exportar metadato CU3, posee una relación de extensión (dependencia opcional, el usuario decide si la ejecuta o no) indicada por el estereotipo

<<extend>> con el caso de uso Abrir archivo CU4, pues el usuario decide el lugar de almacenamiento y modo de representación del metadato para su lectura.

En seguida se tienen las operaciones de inicio de sesión Autenticar CU5 y las asociadas a la gestión de metadatos geográficos CU6, la cual posee tres relaciones: una de inclusión con el caso de uso Codificar QR CU7, que a su vez requiere se ejecute previamente el caso de uso Crear metadato CU8 (se requiere crear un registro de metadato en la BD para generar el código QR); y las dos relaciones de extensión Modificar metadato CU10 y Eliminar metadato CU11.

Nótese que para CU10 y CU11 existe una dependencia obligatoria con el caso de uso Consultar metadato CU12 (el sistema requiere satisfacer un criterio de consulta o filtro en la base de datos para poder eliminar o editar un metadato respectivamente).

Adviértase de igual modo que existe una relación de herencia entre los actores. El Administrador, hereda las acciones del Gestor, quien a su vez ha heredado las capacidades del Usuario; incluyéndose las tareas que corresponden específicamente a su rol.

Adicionalmente, vale la pena señalar que el actor, como entidad externa que tiene interés en interactuar con el sistema, a menudo, también puede ser otro sistema o alguna clase de dispositivo hardware que necesita interactuar con el éste como es el caso del actor API que permite la conexión entre la BD y la publicación de los respectivos metadatos geográficos.

El diagrama de paquetes (figura 30) permite, desde una arquitectura lógica, realizar agrupaciones físicas de funcionalidades u operaciones a llevarse a cabo en el sistema gestor de metadatos geográficos, en módulos de acuerdo a la cohesión de las mismas, permitiendo su diseño desde el punto de vista estructural.

Su utilidad estriba en que facilita dividir el sistema en partes para repartir el trabajo o definir subsistemas. El propósito es maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes.

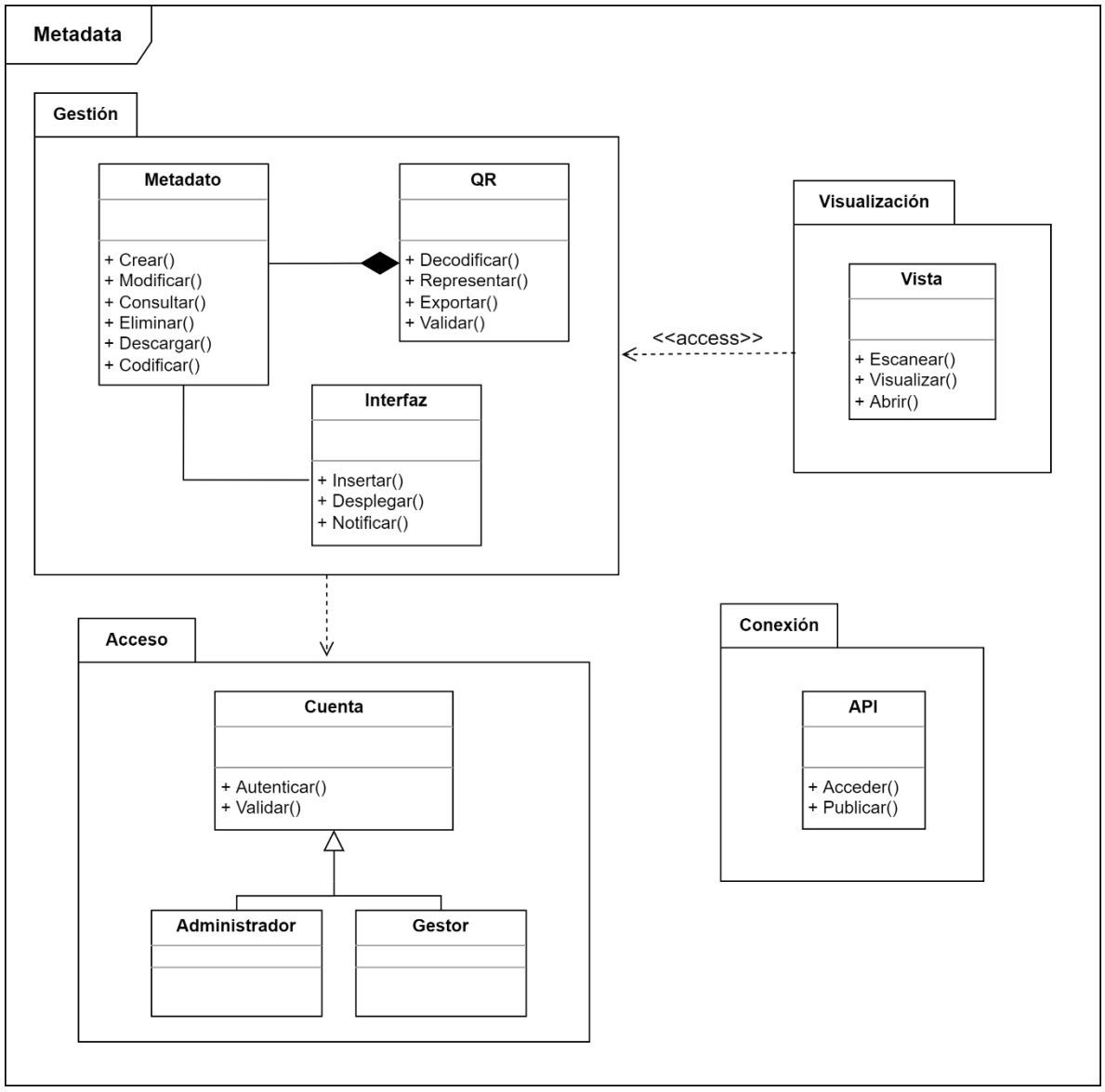


Figura 30. *Diagrama de paquetes del sistema. Fuente: Elaboración propia.*

Obsérvese en la figura 30 que se tienen fundamentalmente 4 módulos (implica que posteriormente se convertirán en ejecutables para cada agrupación). El módulo de gestión agrupa una composición de las clases Metadato y QR.

El módulo de visualización contiene la clase Vista cuya colección de funcionalidades permite la interacción del usuario con el contenido del metadato del mapa digital en pantalla. Este paquete a su vez posee una relación de dependencia con el paquete Gestión mediante el estereotipo <<access>> al requerir acceso al contenido de éste.

El paquete conexión está compuesto por la clase API encargada de la publicación del metadato geográfico para su consumo desde las aplicaciones que componen el sistema.

Finalmente se tiene un módulo acceso, desde el cual se administra los inicios de sesión y las capacidades habilitadas para la cuenta Administrador y Gestor.

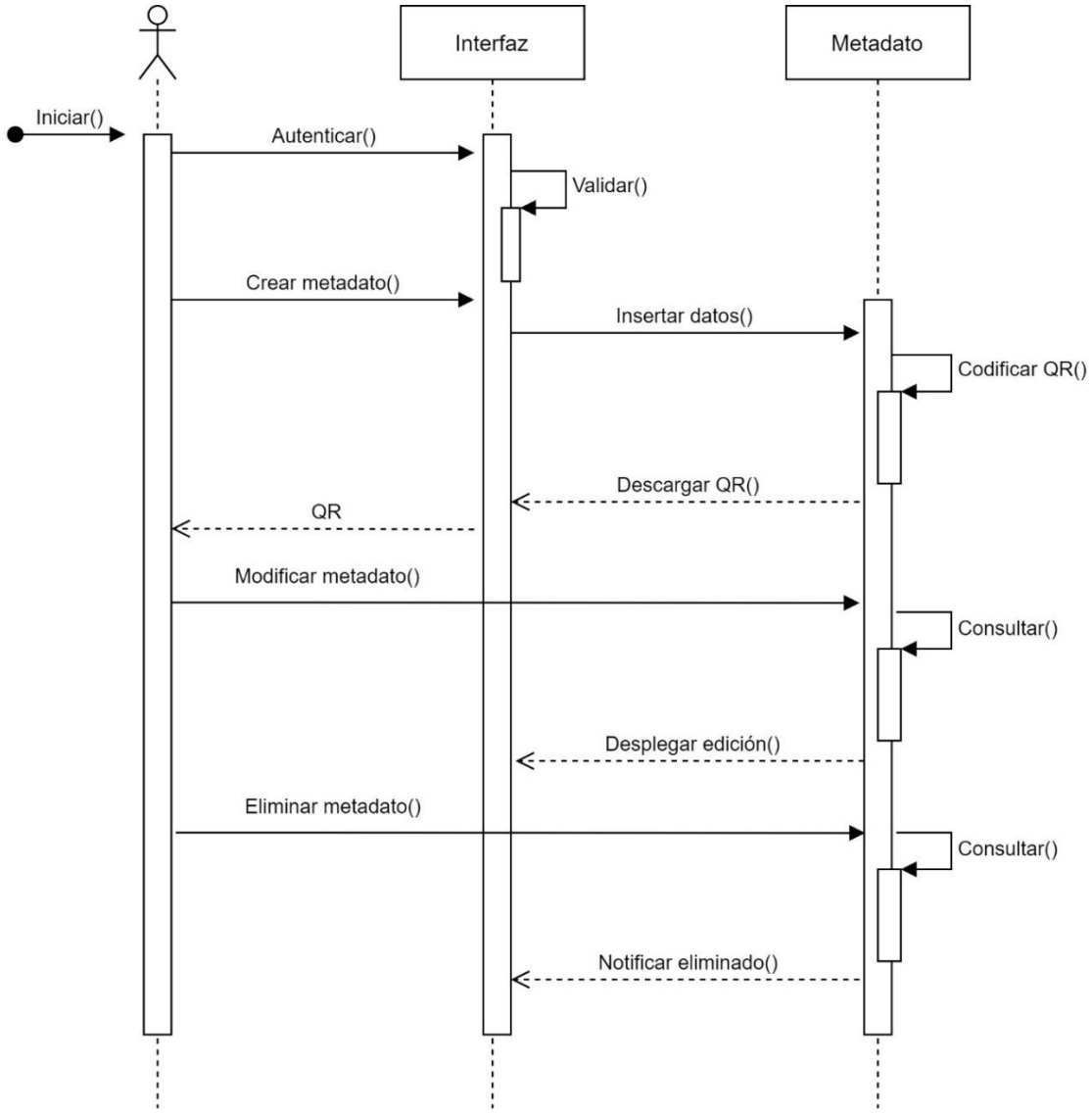


Figura 31. *Diagrama de secuencias del módulo gestión. Fuente: Elaboración propia.*

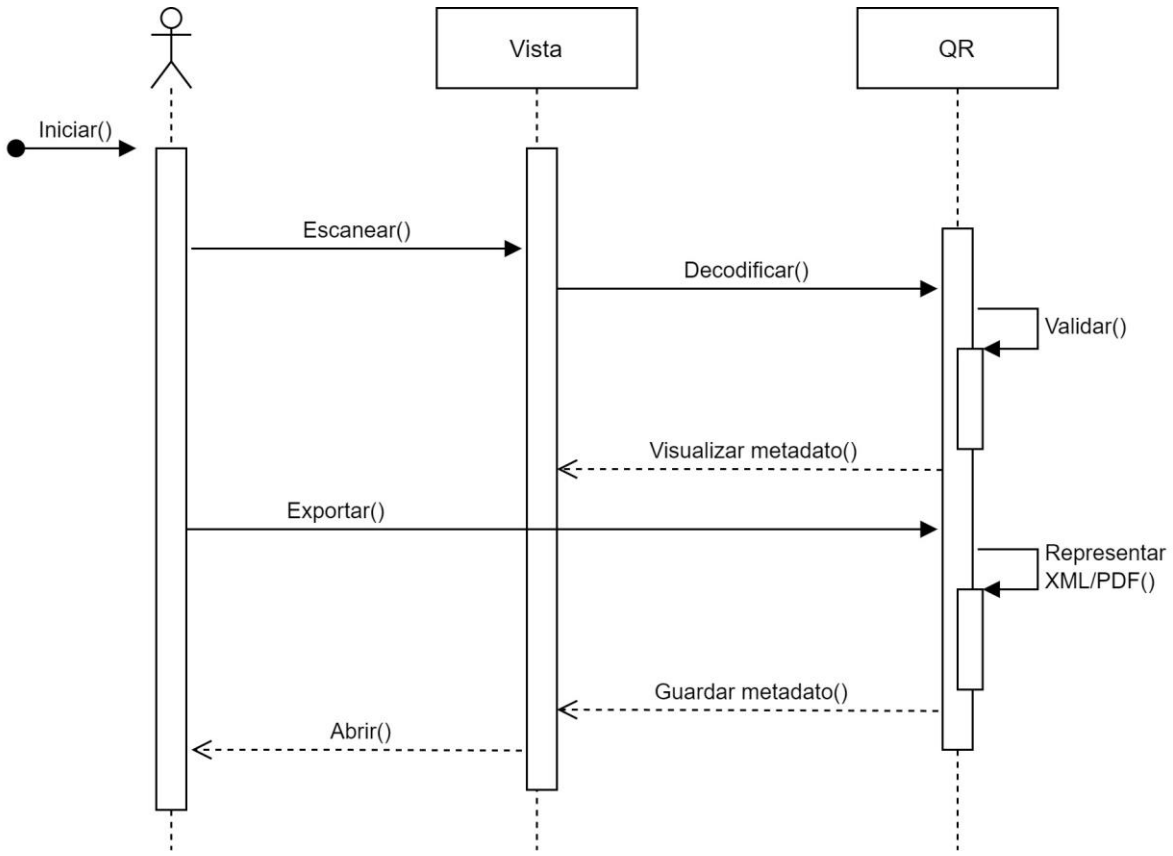


Figura 32. *Diagrama de secuencias del módulo visualización. Fuente: Elaboración propia.*

Las figuras 31 y 32 representan diagramas de secuencia, los cuales facilitan el modelamiento comportamental del sistema. Mediante este diagrama se identifica cómo sus objetos colaboran para lograr el objetivo mediante las interacciones, considerando una secuencia lógica. Por esta razón, cada una de las secuencias representa de manera detallada las tareas del proceso, así como sus ciclos de vida al interior del sistema.

El cuarto diagrama considerado en el diseño del modelo de gestión de metadatos geográficos corresponde al diagrama de componentes (figura 33). Se trata de un tipo de diagrama estructural que proporciona una vista de alto nivel de las unidades modulares dentro del sistema. Permite establecer los componentes, su comportamiento o comunicación en términos de sus interfaces y su nivel de acoplamiento.

Un componente se comunica con otro en términos de interfaces provistas y requeridas.

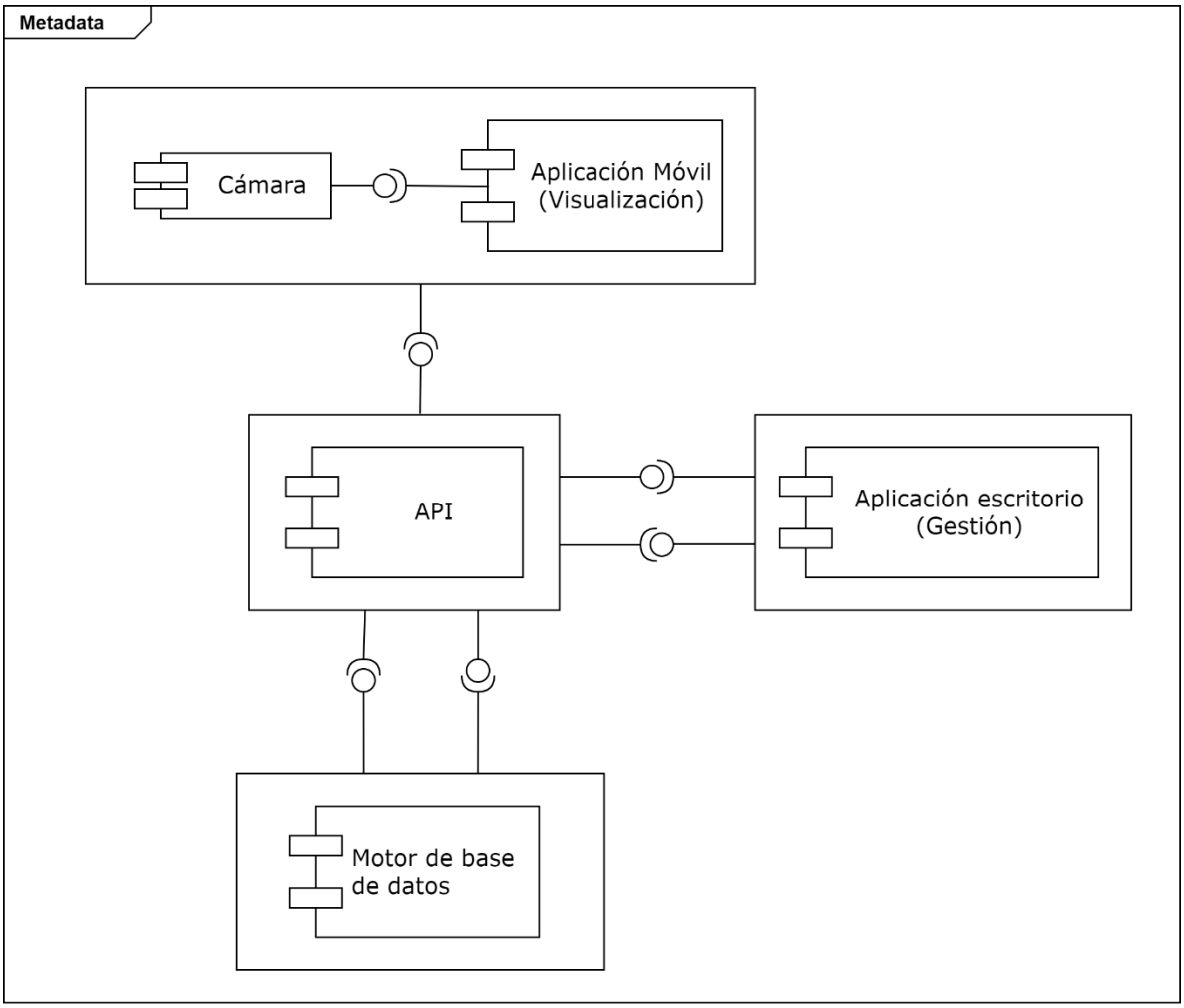


***Figura 33**. Diagrama de componentes del sistema. Fuente: Elaboración propia.*

De la figura 33 es posible identificar 6 componentes con sus respectivas interfaces (proveen y requieren datos al interior del sistema para su funcionamiento). Adviértase además que el diseño se plantea en capas para identificar cuántas se requieren para construir el sistema.

De esta manera, el componente de base de datos o primera capa gestiona la persistencia del sistema. La capa de lógica de negocio la constituyen la aplicación de gestión de metadatos geográficos y la API. La capa de presentación la compone la aplicación móvil mediante la cual se lee y visualiza el metadato geográfico codificado en QR.

El siguiente diagrama de despliegue (figura 34) muestra cómo se sitúan los componentes lógicos previamente identificados, en los diferentes nodos físicos (arquitectura cliente-servidor).

Se incorpora a este documento para dar una visión global de la implementación y, por tanto, la configuración en funcionamiento del sistema. Mediante el diagrama se representan los entornos de ejecución y la relación física de los distintos nodos o servidores requeridos, exponiendo la asociación entre éstos y la forma en cómo se distribuyen.

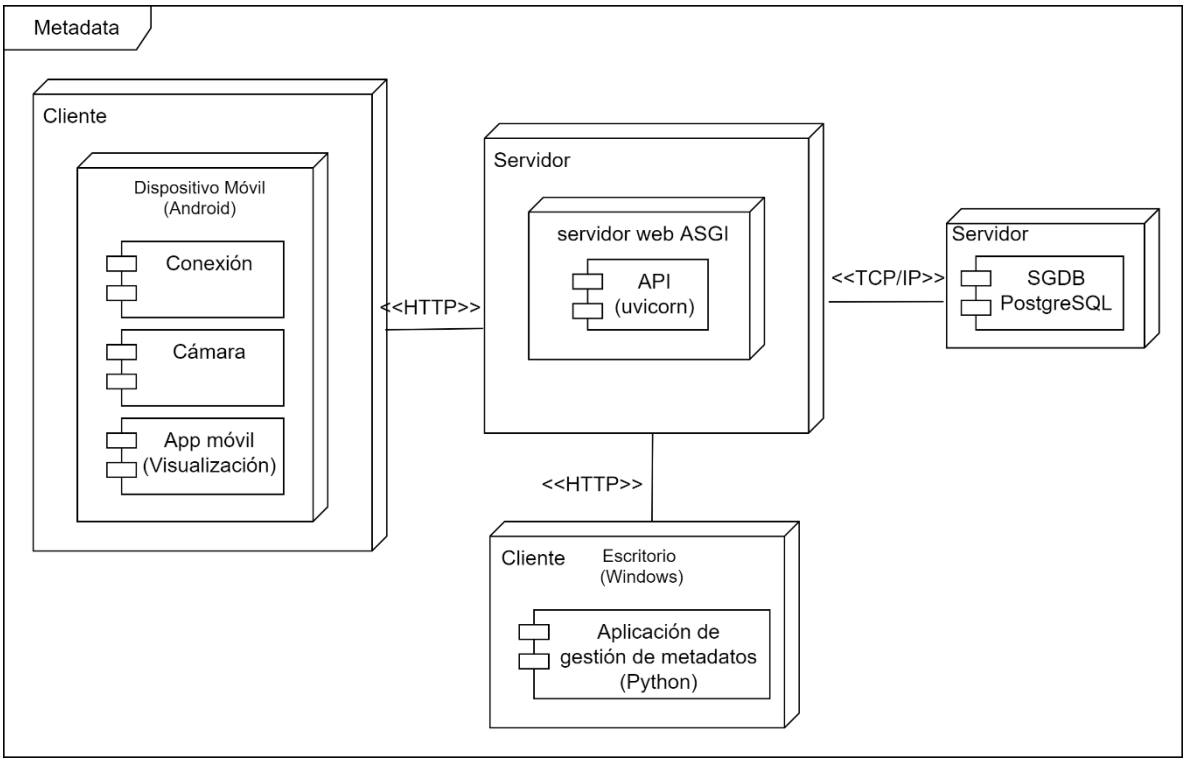


Figura 34. *Diagrama de despliegue del sistema. Fuente: Elaboración propia.*

Obsérvese en el diagrama anterior que se representan 4 nodos con sus respectivos componentes para el funcionamiento en productivo del sistema de gestión de metadatos geográficos.

Se tienen dos nodos cliente que tienen dependencia directa con el servidor de la API. En los nodos cliente se está ilustrando un equipo común de escritorio con sistema operativo Windows con acceso a una red IP que conecta al nodo API.

La segunda aplicación cliente corresponde al dispositivo móvil con sistema operativo Android quien se comunica con el servidor de la API mediante el protocolo HTTP.

El nodo servidor soporta el servidor web ASGI para Python donde está alojada la API. Tiene dependencia directa con el servidor donde está alojado el sistema gestor de base de datos.

Finalmente, se tiene el nodo donde está alojado el motor de base de datos PostgreSQL que alimenta las aplicaciones del sistema.

6.4. Propuesta del esquema de metadatos geográficos y el modelo de persistencia

Como se mencionó en secciones anteriores, potencialmente, cualquier producto informativo en un GIS o IDE de nivel corporativo puede ser documentado mediante metadatos geográficos. Razón por la cual, se constituyen como un mecanismo para caracterizarlos de modo que los usuarios puedan localizarlos y acceder a ellos de manera más fácil y eficiente.

Pero los metadatos únicamente son útiles si realmente mejoran la comprensión y el entendimiento de los datos, procesos y recursos asociados que describen. En definitiva, cuando los mapas digitales poseen el contexto que los hace comprensibles mediante el uso de los metadatos, entonces se convierten en información valiosa y útil para el usuario al momento de atender fines específicos de negocio.

En este sentido, se pretende diseñar un esquema que proporcione información acerca de la identificación, contenido, calidad, y distribución para cualquier mapa digital que se genere al interior de una organización.

Para este fin, se especifican elementos de este metadato, de carácter obligatorio y opcionales, que constituyen el núcleo mínimo requerido a tenor del estándar ISO19115-1:2014, necesarios para cumplir los propósitos de acceder a las descripciones del recurso, su aptitud de uso, restricciones, entre otros aspectos que condicionen su adecuado aprovechamiento.

El metadato geográfico para mapas digitales basado en QR está categorizado en una jerarquía de relaciones y organización de la información (figura 35), que comprende tres secciones principales: referencia del metadato, identificación y contenido. Así mismo, existen dos secciones con dependencia: calidad de los datos y distribución.

Estas secciones están subdivididas en entidades (ítem de datos cuya definición, identificación, representación y valores permisibles son especificados por medio de un conjunto de elementos), y en elementos de metadato que contienen los campos individuales de documentación, usados para identificar del mapa digital y el conjunto de datos que lo alimenta.

Las características de los elementos del metadato geográfico se definen por 7 atributos:

- Numeración (indica la clasificación jerárquica de cada elemento del metadato dentro de cada sección).
- Nombre (etiqueta asignada a un elemento de metadato), descripción (explicación del elemento de metadato).
- Obligatoriedad (descriptor que indica si el elemento del metadato es obligatorio u opcional a discreción del productor del mapa).
- Ocurrencia (número máximo de ocurrencias que el elemento de metadato puede tener 1-N).
- Tipo de dato (propiedad de un valor que determina su dominio) y dominio (valores permitidos para el elemento de metadato).

Estos atributos están contenidos en el diccionario de datos del esquema de metadato. Para este caso, corresponde al contemplado en el estándar ISO19115-1 del metadato mínimo, incluyendo unos elementos adicionales del metadato detallado.

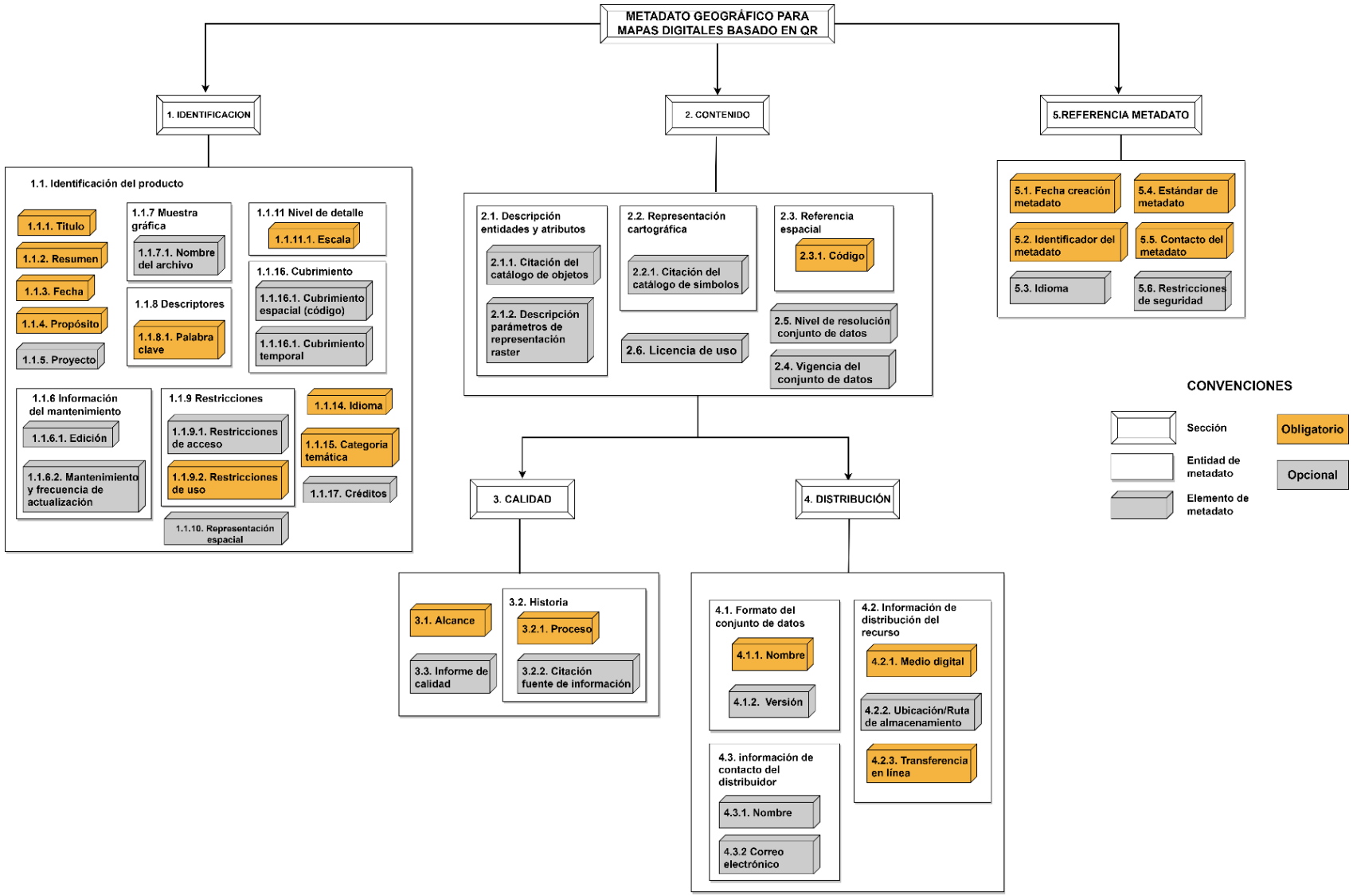


Figura 35. *Esquema del metadato geográfico basado en QR. Fuente: Elaboración propia.*

El modelo presentado en la figura 36 representa el diagrama de persistencia del sistema. A partir de la definición de las clases que serán almacenadas en la base de datos, es posible alimentar las transacciones efectuadas desde la aplicación de gestión y de visualización respectivamente. El diagrama se correlaciona con el inventario y especificación de las clases descritas en el estándar ISO19115-1, adaptado al esquema anterior.

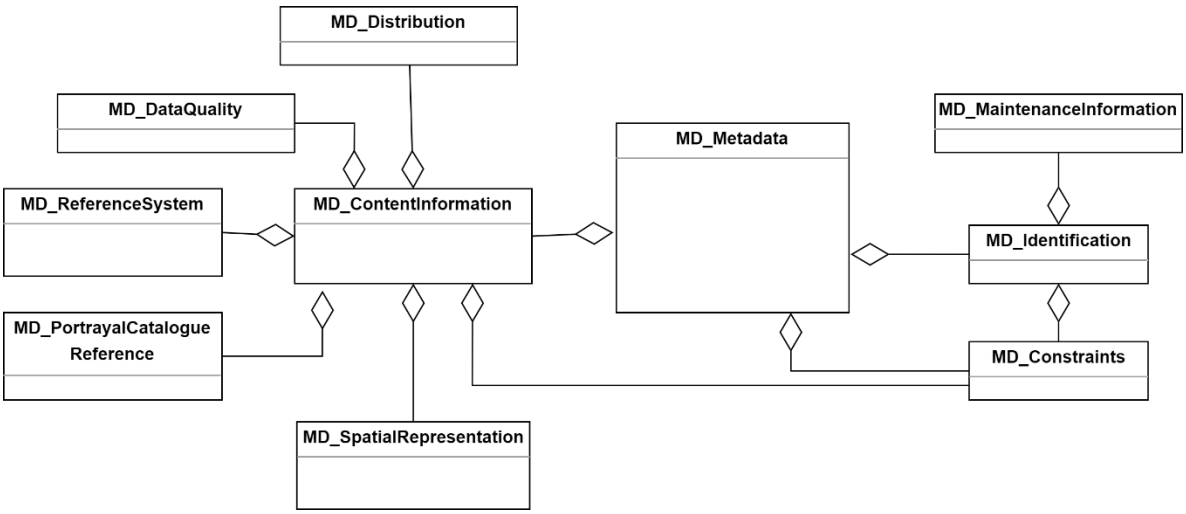


Figura 36. *Diagrama de persistencia del sistema. Fuente: Elaboración propia.*

Conviene mencionar que el diseño de los modelos de las figuras 35 y 36 responden a un modelamiento de datos de tipo conceptual, describiendo en forma estructurada la información alfanumérica que representa el universo de discurso o alcance temático de todo el sistema. Se ha mencionado previamente que la base de datos relacional se implementará en el sistema de gestión de bases de datos PostgreSQL.

6.5. Presentación del modelo de gestión

Finalmente, este apartado concluye con la presentación de un esquema conceptual para la gestión de metadatos geográficos incorporando tecnología QR al interior de una organización como se ilustra en la figura 37.

La mayoría de las instituciones tienen una arquitectura de información que se asemeja a una librería sobrecargada y completamente desorganizada. Una estrategia sólida de gestión de metadatos asegura que los productos de información geográfica de una organización sean de alta calidad, consistentes y precisos para atender los objetivos de negocio, y obtener así una ventaja competitiva.

En general, cuanto más importante es el activo de información, más importante es administrar los metadatos que lo rodean, ya que es un componente preponderante de cualquier iniciativa de gobierno de datos.

Por ello, las principales actividades que se sugieren considerar en este TFM a la hora de implementar un modelo de gestión de metadatos geográficos incorporando la tecnología QR, una vez descrita la arquitectura del sistema requerida, se resume en 5 actividades:

- Formular una política de gestión de metadatos geográficos al interior de la organización (estratégicas y operacionales), determinación de actores y selección de instrumentos de aplicación efectiva (manuales de procedimiento, por ejemplo).
- Adoptar el esquema de metadatos geográficos. El esquema propuesto para este proyecto se hizo de conformidad al estándar ISO. No obstante, la organización a su discreción puede seleccionar estándares de metadatos geográficos que mejor se adapten a sus necesidades de gestión, y definir el respectivo perfil.
- Planear el proceso de gestión desde el punto de vista técnico y administrativo. Esto incluye la definición de objetivos y alcances, definición de fases y sus respectivas actividades, el levantamiento de requerimientos, la definición de los elementos lógicos y conceptuales (diseño de modelos) que viabilizan su implementación y un análisis costo beneficio.
- Administrar el ciclo de vida del metadato geográfico sobre la solución informática, con principal foco en la producción eficiente y la usabilidad.
- Publicar y validar. Esto incluye un seguimiento y retroalimentaciones periódicas para mejorar el funcionamiento y control de calidad. Así como, estrategias de capacitación y entrenamiento de los actores a lo largo del proceso.

***Figura 37**. Modelo de gestión de metadatos geográficos incorporando tecnología QR. Fuente: Elaboración propia.*

7. Construcción del sistema de gestión de metadatos geográficos codificados con tecnología QR

7.1. Configuración del entorno de desarrollo

El proceso inicia con la instalación de las herramientas requeridas para la construcción del sistema de gestión Metadata. Por ende, se inicia con la instalación de la IDE Spyder, como se ilustra a continuación:

```
C:\Users\57322>pip install spyder
```

Sypider es un entorno de desarrollo integrado (IDE) multiplataforma de código abierto para la programación en el lenguaje Python. Spyder usa Qt para su GUI y está diseñado para usar los enlaces PyQt o PySide Python.

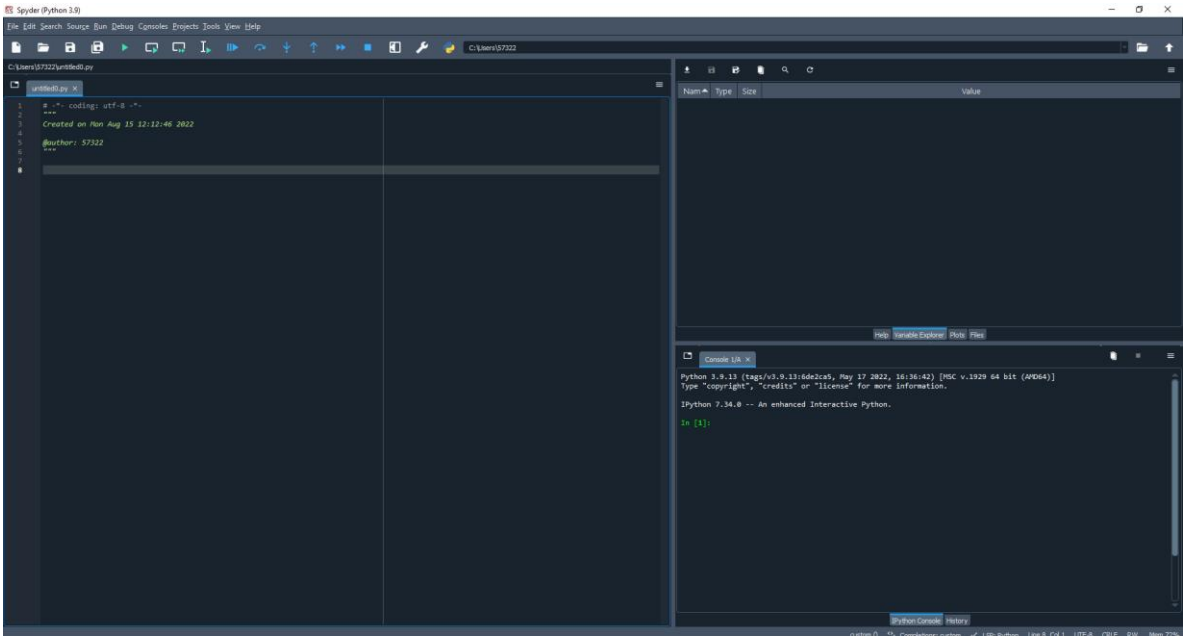


Figura 38. *Entorno de desarrollo Spyder. Fuente: Elaboración propia.*

Posteriormente se procede con la instalación de los módulos requeridos:

- Módulo para crear la API:

```
C:\Users\57322>pip install fastapi
```

- Módulo Uvicorn para poner la API en servicio:

```
C:\Users\57322>pip install uvicorn
```

- Módulo para generar los códigos QR:

```
C:\Users\57322>pip install qrcode
```

- Módulo para interactuar con la base de datos SQL

```
C:\TFM>pip install databases
```

- Módulo uuid para identificadores únicos

```
C:\Users\57322>pip install uuid
```

- Módulo para trabajar con imágenes

```
C:\Users\57322>pip install pillow
```

En seguida se inicia la instalación de la aplicación PgAdmin la cual permite la administración del SGBD PostgresQL. Una vez instalada se procede a crear la base de datos relacional que alimenta el sistema gestor de metadatos geográficos.

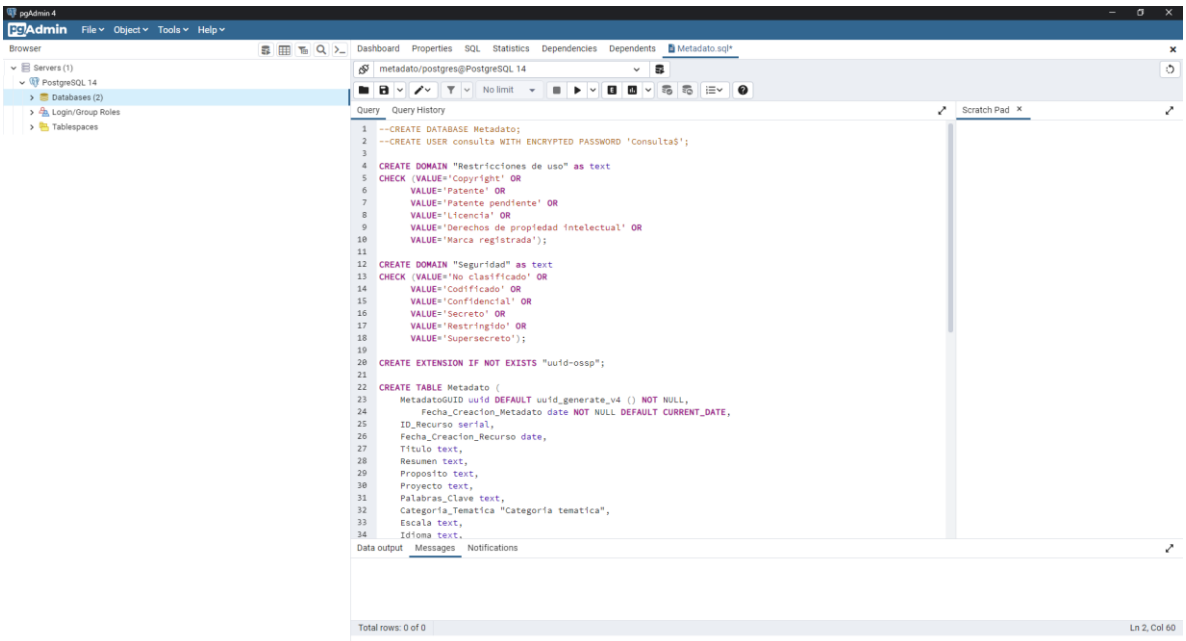


***Figura 39**. Creación de base de datos del sistema en PgAdmin4. Fuente: Elaboración propia.*

Por último, se instala y configura la IDE para codificar la aplicación móvil. Android Studio es el entorno de desarrollo integrado oficial para el desarrollo de apps para Android.

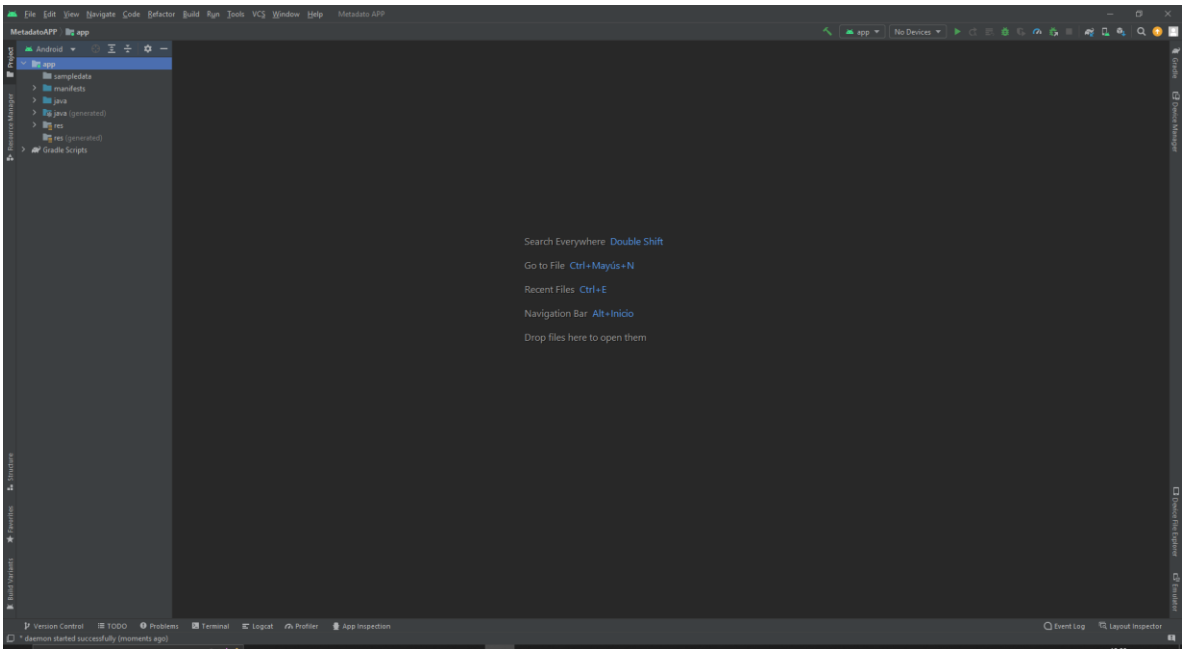


Figura 40. *Entorno de trabajo Android Studio. Fuente: Elaboración propia.*

7.2. Diseño de la interfaz gráfica de usuario del sistema

Antes de proceder con el diseño formal de las interfaces de las aplicaciones que componen el sistema, es importante mencionar que el patrón de diseño adoptado corresponde al Modelo Vista Controlador (MVC).

Este patrón de arquitectura de software permite separar los datos de las aplicaciones, la interfaz gráfica de usuario (GUI) y la lógica de control en tres componentes distintos.

Un diagrama sencillo que ilustra la relación entre el modelo, la vista y el controlador, es el siguiente:

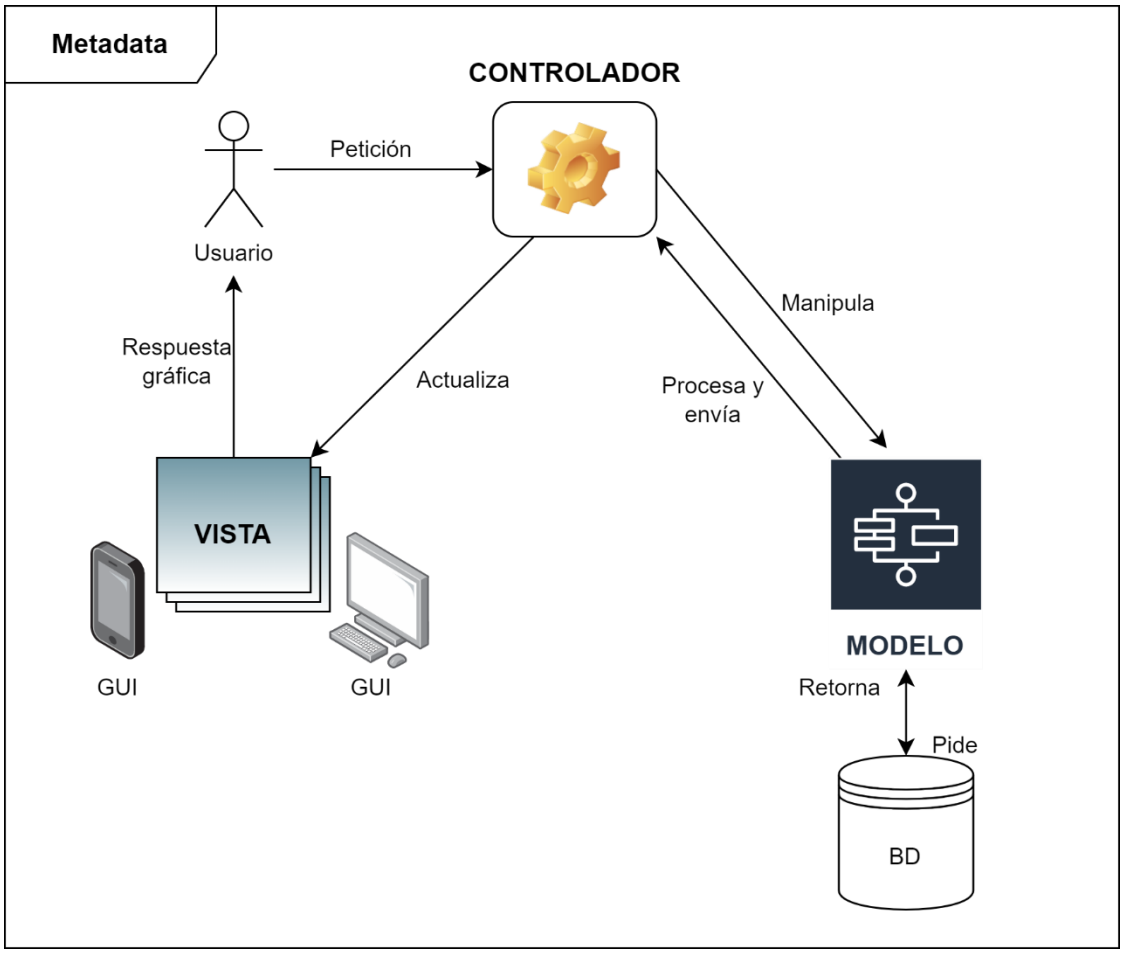


Figura 41. *Patrón de diseño MVC. Fuente: Elaboración propia.*

El modelo. Esta es la representación específica de la información (esquema de metadatos ver figuras 35 y 36) con la cual el sistema opera. La lógica del modelo asegura la integridad de los datos que se implementa en un motor de base de datos.

La vista. Es el encargado de presentar el modelo en un formato adecuado para interactuar con los actores del sistema. Esta corresponde a la interfaz gráfica de usuario GUI.

Para este TFM, la vista del MVC posee dos variantes: la interfaz de la aplicación de escritorio encargada de la gestión de metadatos y la aplicación para dispositivos móviles encargada del acceso y lectura del metadato encriptado en el código QR.

El controlador. Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y por tanto en la vista. El flujo que sigue el control típicamente es el siguiente:

- El usuario interactúa con las interfaces de usuario (por ejemplo, pulsa un botón).
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler o callback).
- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada al usuario que refleje los cambios en el modelo (un metadato recién creado, en el que se desplegará el código QR asociado). El modelo no debe tener conocimiento directo sobre la vista. El controlador no pasa objetos de dominio (el modelo) a la vista, aunque puede dar orden a la vista para que se actualice. La vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Junto con la definición de la arquitectura, en el modelo MVC el cliente se encarga de enviar las peticiones a través del protocolo HTTP a los controladores dentro de la arquitectura MVC, y a su vez se representa las vistas como respuesta a sus peticiones, esto de una forma más detallada es:

- El cliente envía una petición.
- El controlador interactúa con el modelo.
- El controlador invoca la vista

Para el diseño de las interfaces gráficas GUI se tuvo en cuenta 3 aspectos importantes: intuitividad, eficiencia y consistencia de diseño (apariencia) requeridos para el sistema (las interfaces deben estar ajustadas al modelo mental de los actores para evitar problemas de indefensión).

Por esta razón, en esta fase se tomaron decisiones asociadas a los elementos de estilo, aspecto, ergonomía y comportamiento de la interfaz gráfica de usuario (GUI) de la aplicación de escritorio y de la aplicación móvil para maximizar la usabilidad y experiencia de usuario.

La técnica de diseño llevado a cabo en ambas aplicaciones corresponde al maquetado digital, ya que los prototipos se construyeron sobre la plataforma de desarrollo de las aplicaciones (Qt designer y Android Studio), representando las características físicas a implementar.

De modo semejante, se tuvo en cuenta criterios asociados a la adecuada utilización de invitaciones (botones, cajas de activación, barras de navegación y pestañas), metáforas (iconos) y rampas de color basados en el estándar gráfico del manual de marca de la USAL, para customizar ventanas y cuadros de diálogo de manera WYSIWYG (lo que se ve es lo que obtiene). De esta manera, se obtuvo una interfaz sencilla, intuitiva, funcional y de fácil acceso para el usuario.

Las capturas de pantalla a continuación presentan el prototipo de las interfaces o mockups con una dimensión horizontal de baja fidelidad para la aplicación de escritorio desde el software Qt Designer:

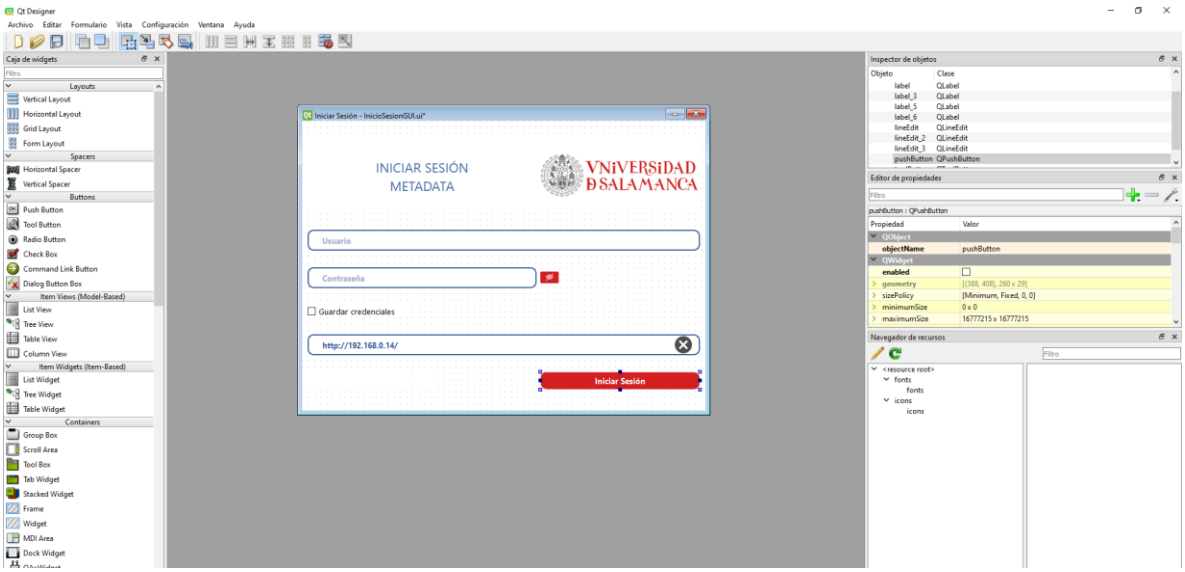


Figura 42. *Diseño GUI inicio de sesión aplicación de escritorio. Fuente: Elaboración propia.*

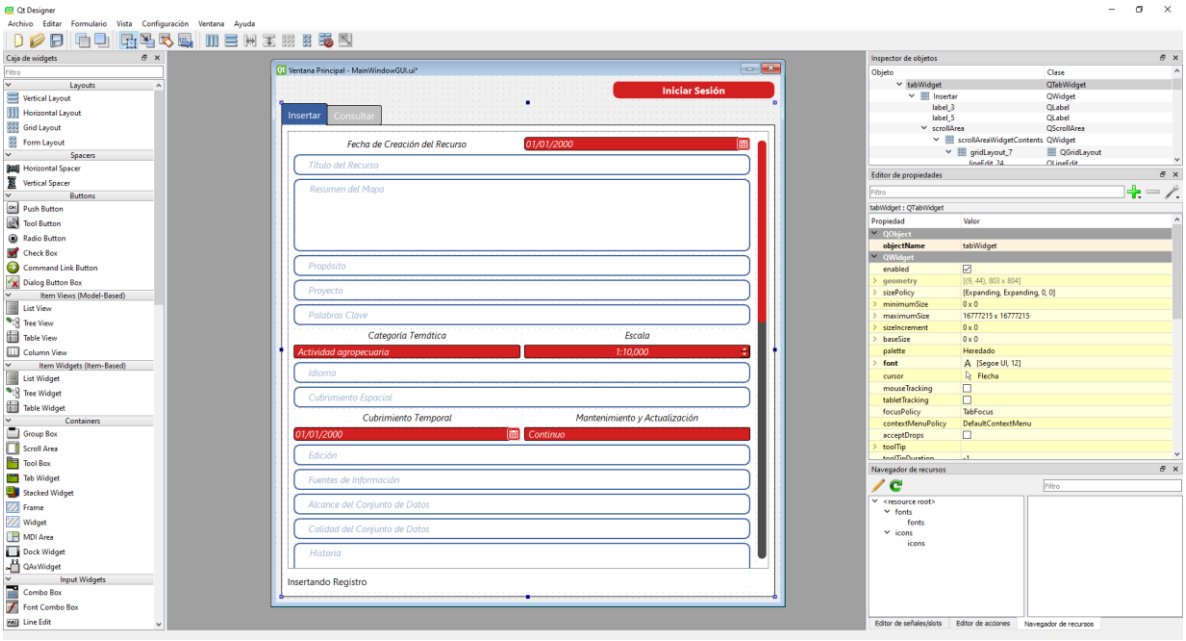


Figura 43. *Diseño GUI ventana principal aplicación de escritorio, pestaña insertar registro de metadato. Fuente: Elaboración propia.*

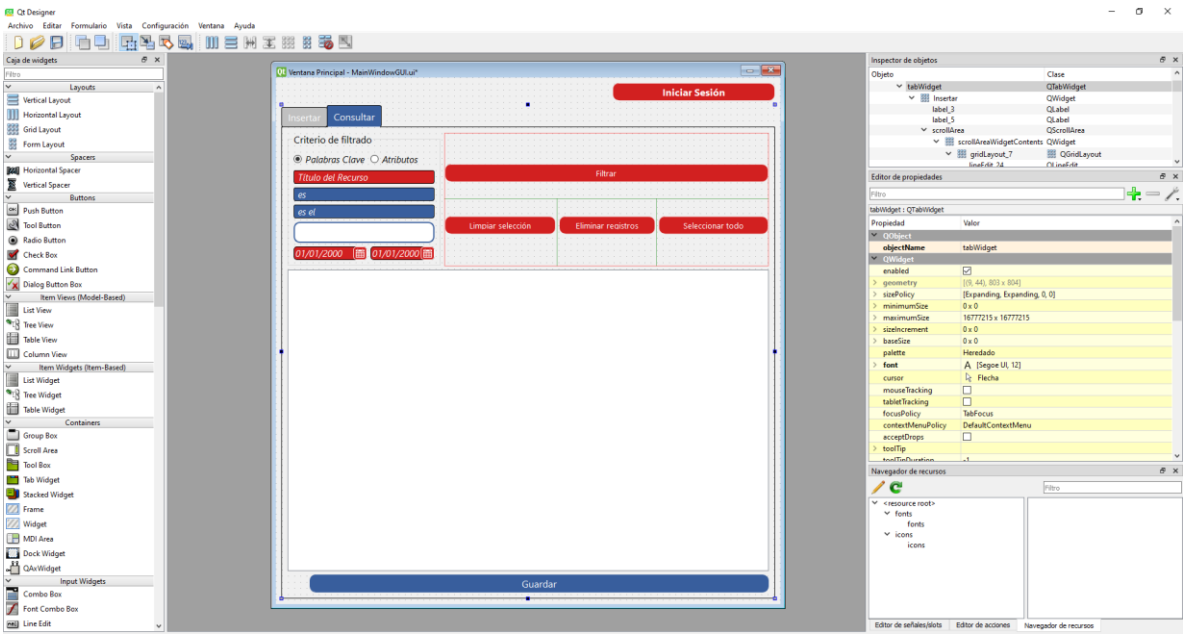


Figura 44. *Diseño GUI ventana principal aplicación de escritorio, pestaña consultar registro de metadato. Fuente: Elaboración propia.*

El siguiente conjunto de capturas corresponden al diseño de la interfaz gráfica de la aplicación móvil configurada sobre la IDE de Android Studio. Android ofrece una variedad de componentes de GUI previamente compilados. De igual modo, la mayor parte de la GUI

se define en archivos en formato XML. El editor de diseño escribe el archivo XML a medida que se arrastra y suelta vistas para compilar el diseño de la app.

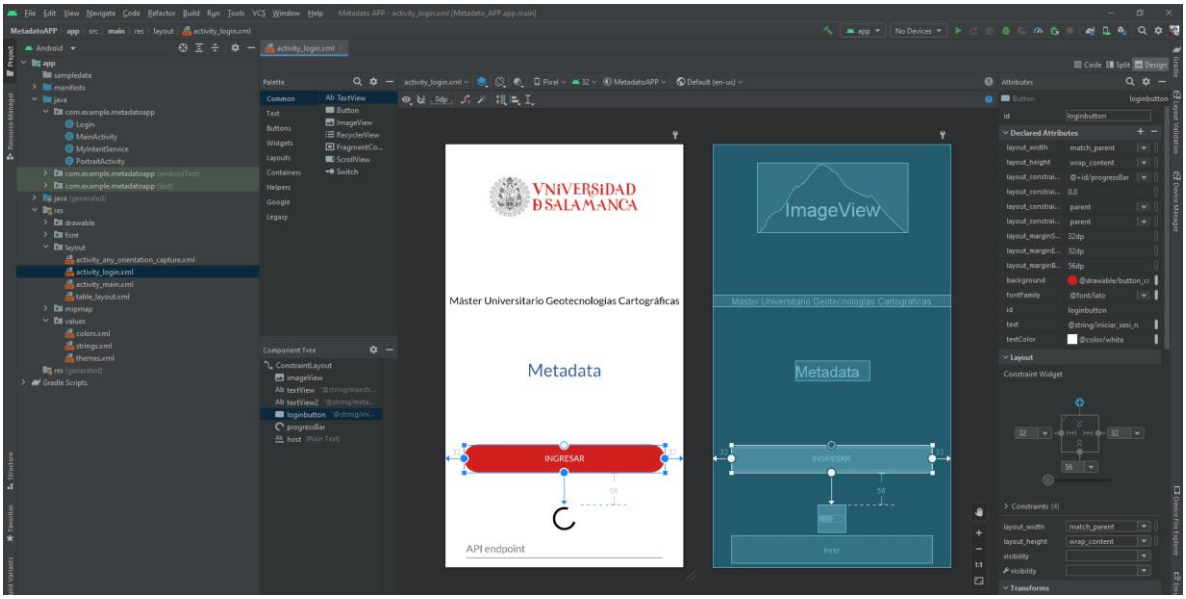


Figura 45. *Diseño GUI ingreso a la aplicación móvil. Fuente: Elaboración propia.*

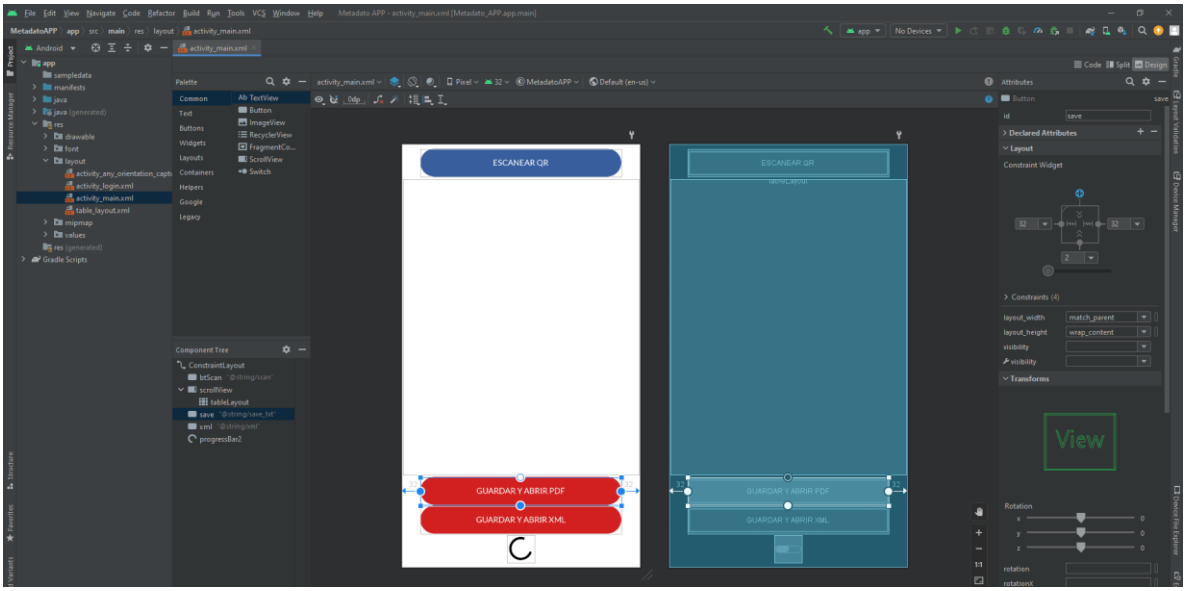


Figura 46. *Diseño GUI ventana principal de la aplicación móvil. Fuente: Elaboración propia.*

7.3. Creación y puesta en servicio de la API

Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. La palabra API significa interfaz de programación de aplicaciones y podría entenderse como una especificación formal que establece cómo un módulo de un software se comunica o interactúa con otro para cumplir una o varias funciones (mediante solicitudes y respuestas).

La arquitectura de las API suele explicarse en términos de cliente y servidor. La aplicación que envía la solicitud se llama cliente, y la que envía la respuesta se llama servidor. Se eligió una API para este TFM ya que es la forma más sencilla de hacer una conexión con un servidor, creada desde la librería de Python FastAPI.

```
from fastapi import FastAPI, Request, status, Depends, Header, HTTPException
from typing import Optional, Literal
from fastapi.responses import JSONResponse
from fastapi.encoders import jsonable_encoder
from pydantic import BaseModel
from databases import Database

import io, qrcode, uuid
```

La importación de librerías presentada anteriormente, dentro del funcionamiento de la API, se tienen unos Request y un Header (es lo que el usuario da como parámetro adicional, estos parámetros pueden ser encabezados o puede ser de tipo Json que ya vendrían a ser el cuerpo de la consulta que se está haciendo a la API).

Los status son cada uno de los códigos que va a retornar la API a partir de las solicitudes de los usuarios, indicando si se obtuvo una ejecución exitosa o se presentó algún error. Los errores se van a obtener mediante el HTTP Exception. EL JSONResponse son todas las respuestas tipo json que se ofrecerá al usuario al ejecutar la API. Del módulo DataBases, se importa la librería Database que es la que tiene FasAPI integrada para hacer las conexiones con la base de datos.

Adicionalmente se importa la librería io que va a permitir obtener los bytes de las imágenes, qrcode para la generación de códigos QR. Luego, con qrcode se crea el código y con io se convierte la imagen QR a bytes para luego ser guardada en la base de datos. Con uuid va a generar el identificador del registro del metadato en la base de datos.

```python
class db():

    databases={}
    tokens=[]

    async def create_db(self,url):
        database=Database(url)
        await database.connect()
        token=uuid.uuid4()
        self.tokens.append(str(token))
        self.databases.update({str(token): database})
```

A través de la clase db se va a hacer la conexión con la base de datos. Justo aquí se hace la inicialización de la Database mediante el módulo del mismo nombre (database = DataBase(url)),y su respectiva conexión con await.database.connect()

```python
class Query(BaseModel):
    Image: Literal['hex']
    MetadatoGUID: Literal['UUID4']
    Fecha_Creacion_Metadato: str
    ID_Recurso: int
    Fecha_Creacion_Recurso: str
    Titulo: str
    Resumen: str
    Proposito: str
    Proyecto: str
    Palabras_Claves: str
    Categoria_Tematica: str
    Escala: str
    Idioma: str
    Cubrimiento_Espacial: str
    Cubrimiento_Temporal: str
    Edicion: str
    Mantenimiento_Actualizacion: str
    Fuentes_Informacion: str
    Alcance_Datos: str
    Calidad_Datos: str
    Historia: str
    Representacion_Espacial: str
    Referencia_Espacial: str
    Nivel_Resolucion: str
    Contenido: str
    Vigencia: str
    Representacion_Cartografica: str
    Estandar_Version: str
    CreditosRecurso: str
    Contacto: str
    Instrucciones_Contacto: str
    Medios_Digitales_Recurso: str
    Ruta_Almacenamiento: str
    FormatoDatos: str
    Distribuidor: str
    Transferencia: str
    Restricciones_Uso: str
    Licencia_Uso: str
    Restricciones_Seguridad: str
    QR: Literal['hex']
```

Mediante la clase Query de tipo BaseModel, corresponde a la documentación que le va a retornar al usuario cuando se haga la consulta, es decir, cada una de las columnas con su respectivo tipo de dato.

```
class Create(Message):
    GUID: Literal['UUID4']
    QR: Literal['hex']
```

Con la clase Create (Message), se va a retornar un mensaje con el identificador del metadato y el respectivo código QR.

```
async def verify_token(access_token: str=Header(default=None)):
    if access_token is None or access_token=='':
        raise HTTPException(status_code=401, detail="No se inicio sesión")
    elif not access_token in db.tokens:
        raise HTTPException(status_code=402, detail="Token inválido")
```

El método anterior very_token permite comprobar cada uno de los tokens que se obtienen al hacer la conexión. A continuación, se procede a inicializar la API y la clase db mediante la siguiente instrucción de código:

```
app = FastAPI()

db = db()
```

La API posee eventos, invocando el de shutdown, es posible apagar todas las conexiones activas para que no haya ejecuciones en segundo plano:

```
@app.on_event("shutdown")
async def shutdown():
    for value in db.databases.values():
        await value.disconnect()
```

Así mismo, las API poseen métodos (GET, POST, Delete, Patch) que a su vez son soportados por el protocolo HTTP.

```
@app.get("/db/connection/{user}", response_model=Connection, responses={400: {"model": Message}, 401: {"model": Message}})
async def conexion_db(user: str, p: str):
    try:
        await db.create_db('postgresql://%s:%s@localhost/metadato' %(user,p))
        token=db.tokens[-1]
        if db.databases[token].is_connected:
            return JSONResponse(status_code=status.HTTP_200_OK, content={'message':'Conexión establecida', 'access_token':token})
        else:
            return JSONResponse(status_code=401, content={'message':"Usuario o contraseña incorrecta"})
    except Exception as error:
        return JSONResponse(status_code=400, content={'message':str(error)})
```

Mediante el método de conexión anterior, sobre el cual solicita el usuario y contraseña de tipo str para hacer la conexión a la base de datos.

```python
@app.get("/db/metadata/query", dependencies=[Depends(verify_token)], response_model=Query, responses= {400: {"model": Message},
async def query_db(request: Request, filter: Optional[str]=""):
    try:
        token=request.headers.get('access-token')
        df = jsonable_encoder(await db.databases[token].fetch_all(query="""SELECT encode(imagen,'hex') AS "IMAGEN", metadatoGUID
            id_recurso AS "ID DEL RECURSO", TO_CHAR(fecha_creacion_recurso :: DATE, 'yyyy-mm-dd') AS "FECHA DE CREACIÓN DEL RECU
            categoria_tematica AS "CATEGORÍA TEMÁTICA", escala AS "ESCALA", idioma AS "IDIOMA", cubrimiento_espacial AS "CUBRIMI
            alcance_datos AS "ALCANCE DEL CONJUNTO DE DATOS", calidad_datos AS "CALIDAD DEL CONJUNTO DE DATOS", historia AS "HIS
            estandar_version AS "ESTANDAR Y VERSIÓN", creditos_recurso AS "CREDITOS DEL RECURSO", contacto AS "CONTACTO", instru
            medios_digitales_recurso AS "MEDIOS DIGITALES DEL RECURSO", ruta_almacenamiento AS "RUTA DE ALMACENAMIENTO", formato
            licencia_uso AS "LICENCIA DE USO",restricciones_seguridad AS "RESTRICCIONES DE SEGURIDAD", encode(qr,'hex') AS "QR"
            FROM METADATO %s;"""%filter))
        if df:
            return JSONResponse(status_code=200, content=df)
        else:
            return JSONResponse(status_code=403, content={"message": "Consulta no arrojo datos"})
    except Exception as error:
        return JSONResponse(status_code=400, content={"message": str(error)})
```

Con este segundo método Get, es posible realizar consultas sobre la base de datos luego se hacer la conexión.

```python
@app.post("/db/metadata/create", dependencies=[Depends(verify_token)], status_code=201, response_model=Create, responses={400: {"model'
async def create_metadata(request:Request):
    try:
        token=request.headers.get('access-token')
        req_info = await request.json()
        met_guid=await db.databases[token].execute(query="INSERT INTO Metadato (%s) VALUES(%s) RETURNING metadatoguid;" %(','.join(req_
        img = qrcode.make(str(met_guid))
        output = io.BytesIO()
        img.save(output, format='PNG')
        await db.databases[token].execute(query="UPDATE metadato SET qr='\\x%s' WHERE metadatoguid='%s';"%(output.getvalue().hex(), met
        return JSONResponse(status_code=201, content={"message": "Creado","GUID": str(met_guid),"QR":output.getvalue().hex()})
    except Exception as error:
        return JSONResponse(status_code=400, content={"message": str(error)})
```

Mediante método Post (crear un recurso nuevo) accediendo a la función create, es posible insertar registros a la base de datos. Nótese que de la sentencia SQL retorna el identificador del metadato (metadatoguid), porque es a partir de éste que se genera el código QR mediante la instrucción qrcode.make(), para luego convertirla en bytes con la sentencia io.BytesIO() y guardar posteriormente al imagen del código en formato png.

```python
@app.patch("/db/metadata/update/{id}", dependencies=[Depends(verify_token)], response_model=Message, responses={400: {"model": Message}
async def update_metadata(id: str, request:Request):
    try:
        token=request.headers.get('access-token')
        req_info = await request.json()
        sql="UPDATE metadato SET"
        for key, value in req_info.items():
            sql="%s %s='%s',"%(sql,key,value)
        await db.databases[token].execute(query="%s WHERE metadatoguid='%s';"%(sql[0:sql.rfind(",")],id))
        return JSONResponse(status_code=200, content={"message": "Actualizado"})
    except Exception as error:
        return JSONResponse(status_code=400, content={"message": str(error)})
```

Utilizando el método de la API Patch (modificar solamente un atributo de un recurso) es posible invocar la función de SQL update para actualizar valores de un campo para un registro de metadato específico.

```python
@app.delete("/db/metadata/delete/{id}", dependencies=[Depends(verify_token)], response_model=Message, responses={400: {"model": Message
async def delete_metadata(id: str,request:Request):
    try:
        token=request.headers.get('access-token')
        await db.databases[token].execute(query="DELETE FROM Metadato WHERE metadatoguid='%s';"%id)
        return JSONResponse(status_code=200, content={"message": "Registro eliminado"})
    except Exception as error:
        return JSONResponse(status_code=400, content={"message": str(error)})
```

Finalmente, mediante la operación Delete de la API (eliminar un recurso determinado) es posible eliminar un registro de metadato almacenado en la base de datos, La API requiere para ello únicamente el identificador (id) del registro.

7.4. Construcción de la aplicación de escritorio para la gestión de metadatos geográficos

7.4.1. Codificación de la Interfaz de inicio de sesión

Para la construcción de la aplicación de escritorio, previamente se crearon las interfaces gráficas en QtDesigner y convertidas posteriormente a archivos .py. Hecho esto, se procede a hacer las respectivas importaciones de librerías requeridas.

```python
import keyring, os, requests
from PyQt5.QtWidgets import QDialog, QMessageBox, QLineEdit
from PyQt5.QtGui import QMovie
from PyQt5.QtCore import QObject, pyqtSignal, QThread
from cryptography.fernet import Fernet

from InicioSesionGUI import Ui_Dialog
```

De la anterior imagen, además de la importación de las librerías propias de la GUI, obsérvese que se ha importado la librería request para hacer las consultas a la API, la librería os para realizar ciertas funcionalidades dentro del sistema operativo y la librería Keyring para hacer el guardado de las credenciales requeridas en la interfaz de inicio de sesión (InicioSesionGUI).

La clase Worker de tipo QObject a continuación permite ejecuciones en segundo plano. Evita que la interfaz se congele mientras ejecuta los procesos. En esta clase, se emiten y reciben señales mediante la librería de pyqtSignal. Posteriormente la clase hace la respectiva conexión con la API.

```python
class Worker(QObject):

    failed=pyqtSignal(int,str)
    finished = pyqtSignal()
    completed=pyqtSignal(str,str)
    username=None
    password=None
    port=None

    def run(self):
        try:
            connection = requests.get('%sdb/connection/%s?p=%s' %(self.port, self.username, self.password))
            if connection.status_code==200:
                self.completed.emit(connection.json()['message'],connection.json()['access_token'])
            else:
                self.failed.emit(connection.status_code,connection.json()['message'])

        except Exception as e:
            self.failed.emit(0,str(e))

        self.finished.emit()
```

La clase Dialog permite consumir la interfaz gráfica en cuestión, allí se inicializa todos los recursos (widgets) dispuestos en el cuadro de dialogo.

```python
class Dialog(QDialog,Ui_Dialog):
    def __init__(self, parent):
        super().__init__(parent)
        self.setupUi(self)
        self.label_5.setVisible(False)
        self.label_6.setVisible(False)
        self.gif=QMovie('resources/icons/loading_dark.gif')
        self.label_5.setMovie(self.gif)
        self.pushButton.clicked.connect(self.user)
        self.lineEdit.editingFinished.connect(self.button)
        self.lineEdit.textChanged.connect(self.text)
        self.lineEdit_2.editingFinished.connect(self.toolBtn)
        self.toolButton.pressed.connect(self.activepassword)
        self.toolButton.released.connect(self.desactivepassword)

        if keyring.get_password('Metadata', 'global_username_key')!=None and os.path.isfile('clave.key')==True:
            try:
                clave=open('clave.key', 'rb').read()
                f=Fernet(clave)
                self.lineEdit.setText(f.decrypt(keyring.get_password('Metadata', 'global_username_key')[2:-1].encode()).decode())
                self.lineEdit_2.setText(f.decrypt(keyring.get_password('Metadata', 'password_key')[2:-1].encode()).decode())
                self.lineEdit_3.setText(f.decrypt(keyring.get_password('Metadata', 'port_key')[2:-1].encode()).decode())
                self.toolButton.setVisible(False)
                self.checkBox.setChecked(True)
                self.pushButton.setEnabled(True)
            except:
                QMessageBox.warning(self, 'Error', 'Las credenciales guardadas son inválidas\nPor favor actualícelas')
```

Con el condicional anterior aplicado al método kearing.get_password, es posible guardar las credenciales encriptándolas, para que otro usuario que ingrese a los archivos del programa no pueda visualizar estos datos de loggin. En seguida, se presentan las operaciones codificadas de esta interfaz:

```
def text(self,text):
    self.pushButton.setEnabled(False)

def button(self):
    self.pushButton.setEnabled(True)

def toolBtn(self):
    self.toolButton.setVisible(True)

def activepassword(self):
    self.lineEdit_2.setEchoMode(QLineEdit.EchoMode.Normal)


def desactivepassword(self):
    self.lineEdit_2.setEchoMode(QLineEdit.EchoMode.Password)
```

Con el método text deshabilita el botón de inicio de sesión. Se ejecuta cuando se esté editando o cambiando el lineEdit.textChanged.connect(self, text), por ejemplo si el usuario está editando el usuario, el botón de inicio de sesión va a estar deshabilitado hasta que termine la edición.

Con el toolBtn permite que el botón de ver la contraseña sea visible para cuando se esté modificando una contraseña. El método activepassword y desactivepassword, posibilita que la visualización se haga de modo normal (visibles los caracteres) o modo contraseña (no visible).

```
def user(self):
    self.lineEdit.setReadOnly(True)
    self.lineEdit_2.setReadOnly(True)
    self.checkBox.setDisabled(True)
    self.toolButton.setDisabled(True)
    self.pushButton.setDisabled(True)
    self.label_5.setVisible(True)
    self.label_6.setVisible(True)
    self.label_6.setText('Iniciando sesión')
    self.gif.start()
    self.thread = QThread()
    self.worker = Worker()
    self.worker.moveToThread(self.thread)
    self.thread.started.connect(self.worker.run)
    self.worker.finished.connect(self.thread.quit)
    self.worker.finished.connect(self.worker.deleteLater)
    self.thread.finished.connect(self.thread.deleteLater)
    self.worker.completed.connect(self.complete)
    self.worker.failed.connect(self.failed)
    self.worker.username=self.lineEdit.text()
    self.worker.password=self.lineEdit_2.text()
    self.worker.port=self.lineEdit_3.text()
    self.thread.start()
```

El método Usuario def user() se ejecuta al momento de dar clic en el botón iniciar sesión, y permite ejecutar los procesos en segundo plano asociados al inicio de sesión. Si hay fallo en esta tarea, se ejecuta el método def failed()

```
def failed(self, cod, message):
    self.gif.stop()
    self.label_5.setVisible(False)
    self.label_6.setVisible(False)
    self.lineEdit.setReadOnly(False)
    self.lineEdit_2.setReadOnly(False)
    self.checkBox.setEnabled(True)
    self.toolButton.setEnabled(True)
    self.pushButton.setEnabled(True)
    QMessageBox.warning(self,'Error '+str(cod),message)
```

Con este método, se para el gif de inicio de sesión (reloj de arena), se ocultan los labels de inicio de sesión, se habilitan de nuevo los lineEdit para que puedan ser modificados usuario y contraseña, así como el botón de inicio de sesión. Si por el contrario el inicio de sesión es exitoso se ejecuta el siguiente método:

```
def complete(self,message,access_token):
    self.gif.stop()
    self.label_6.setText(message)
    self.access_token=access_token
    self.port=self.lineEdit_3.text()
    if self.checkBox.isChecked():
        clave=Fernet.generate_key()
        with open('clave.key', 'wb') as key_file:
            key_file.write(clave)
        clave_f=open('clave.key', 'rb').read()
        f=Fernet(clave_f)
        username_key='global_username_key'
        service_id='Metadata'
        username=self.lineEdit.text().encode()
        username_enc=f.encrypt(username)
        password=self.lineEdit_2.text().encode()
        password_enc=f.encrypt(password)
        port=self.lineEdit_3.text().encode()
        port_enc=f.encrypt(port)
        keyring.set_password(service_id, username_key, username_enc)
        keyring.set_password(service_id, 'password_key', password_enc)
        keyring.set_password(service_id, 'port_key', port_enc)
    self.close()
```

Del método anterior, se obtiene el mensaje y el token para acceder a las funcionalidades de la API. Posteriormente, con el condicional if se verifica que el usuario haya activado en el checkbox el guardar las credenciales. Si esto sucede, se genera una llave de encriptación del usuario y la contraseña, y se guarda posteriormente en los archivos del programa.

7.4.2. Codificación de la interfaz ventana principal

Con respecto a la importación de librerías, se incluye la de QR que es donde se va a exponer el código una vez creado el metadato geográfico, para posteriormente descargarlo.

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QDialog, QMessageBox, QTableWidgetItem, QWidget, QFileDialog
from PyQt5.QtCore import QObject, pyqtSignal, QThread, QPersistentModelIndex, Qt, QDate, QEvent
from PyQt5.QtGui import QMovie, QPixmap, QImageReader
from PIL import Image
import sys, requests, os, io, shutil

from MainWindowGUI import Ui_MainWindow
from InicioSesion import Dialog as IC_Dialog
from QR import Ui_Dialog as QR_GUI
from cellWidget import Ui_Form
```

En el main se procede a crear la aplicación mediante el QApplication() y ejecutar la interfaz win.run().

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = MainWindow()
    win.showMaximized()
    win.run()
    sys.exit(app.exec())
```

Con la clase MainWindow se tienen todos los elementos y widgets del cuadro de dialogo para inicializarlo y definición de variables internas requeridas.

```
class MainWindow(QMainWindow,Ui_MainWindow):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)
        self.access_token=None
        self.port=None
        self.tabWidget.setEnabled(False)
        self.label_3.setVisible(False)
        self.gif=QMovie('resources/icons/loading_dark.gif')
        self.label_3.setMovie(self.gif)
        self.pushButton.setVisible(False)
        self.pushButton_2.setVisible(False)
        self.pushButton_8.setVisible(False)
        self.pushButton.clicked.connect(self.delete_selection)
        self.pushButton_2.clicked.connect(self.clear_selection)
        self.pushButton_3.clicked.connect(self.insert)
        self.pushButton_4.clicked.connect(self.select_all)
        self.pushButton_5.clicked.connect(self.run)
```

```
        self.pushButton_6.clicked.connect(self.dir_image)
        self.pushButton_7.clicked.connect(self.filtrar)
        self.pushButton_8.clicked.connect(self.clear_image)
        self.pushButton_9.clicked.connect(self.update_db)
        self.radioButton_3.toggled.connect(self.switch_filter)
        self.comboBox_6.currentTextChanged.connect(self.switch_column)
        self.comboBox_7.currentTextChanged.connect(self.switch_operator_str)
        self.comboBox_8.currentTextChanged.connect(self.switch_operator_date)
        self.dateEdit_4.dateChanged.connect(self.validate_date)
        self.tableWidget.verticalHeader().sectionClicked.connect(self.allow_dels)
        self.label_4.setVisible(False)
        self.label_5.setVisible(False)
        self.gif_2=QMovie('resources/icons/loading_dark.gif')
        self.label_4.setMovie(self.gif_2)
        self.dateEdit.setDate(QDate.currentDate())
        self.dateEdit_2.setDate(QDate.currentDate())
        self.dateEdit.installEventFilter(self)
        self.dateEdit_2.installEventFilter(self)
        self.comboBox.installEventFilter(self)
        self.comboBox_2.installEventFilter(self)
        self.comboBox_3.installEventFilter(self)
        self.comboBox_4.installEventFilter(self)
        self.comboBox_5.installEventFilter(self)
        self.spinBox.installEventFilter(self)
        self.switch_filter(True)
```

Uno de los métodos importantes corresponde a la inserción de datos en el formulario del metadato geográfico, los cuales se capturan mediante la interfaz gráfica.

```
def insert(self):
    self.scrollArea.setEnabled(False)
    self.gif.start()
    self.label_3.setVisible(True)
    self.label_5.setVisible(True)
    columns=['Fecha_Creacion_Recurso', 'Titulo', 'Resumen', 'Proposito', 'Proyecto', 'Palabras_Clave', 'Categoria_Tematica',
             'Escala', 'Idioma', 'Cubrimiento_Espacial', 'Cubrimiento_Temporal', 'Edicion', 'Mantenimiento_Actualizacion',
             'Fuentes_Informacion', 'Alcance_Datos', 'Calidad_Datos', 'Historia', 'Representacion_Espacial',
             'Referencia_Espacial', 'Nivel_Resolucion', 'Contenido', 'Vigencia', 'Representacion_Cartografica',
             'Estandar_Version', 'Creditos_Recurso', 'Contacto', 'Instrucciones_Contacto', 'Medios_Digitales_Recurso',
             'Ruta_Almacenamiento', 'Formato_Datos', 'Distribuidor', 'Transferencia', 'Restricciones_Uso', 'Licencia_Uso',
             'Restricciones_Seguridad', 'Imagen']
```

Las columnas anteriores, como es lógico de suponer, corresponden a los elementos del metadato definidos en el esquema y en correspondencia al modelo de datos implementado en PostgreSQL. Posteriormente se toman los valores introducidos en una nueva lista Values[ ] para cada elemento de columnas.

```
values=[]
values.append(self.dateEdit.text())
values.append(self.lineEdit_10.text())
values.append(self.plainTextEdit.toPlainText())
values.append(self.lineEdit_11.text())
values.append(self.lineEdit_12.text())
values.append(self.lineEdit_13.text())
values.append(self.comboBox_3.currentText())
values.append(self.spinBox.text())
values.append(self.lineEdit_14.text())
values.append(self.lineEdit_15.text())
values.append(self.dateEdit_2.text())
values.append(self.lineEdit_16.text())
values.append(self.comboBox_4.currentText())
values.append(self.lineEdit_17.text())
values.append(self.lineEdit_18.text())
values.append(self.lineEdit_19.text())
values.append(self.plainTextEdit_2.toPlainText())
values.append(self.comboBox_5.currentText())
values.append(self.lineEdit_20.text())
values.append(self.lineEdit_21.text())
values.append(self.plainTextEdit_3.toPlainText())
values.append(self.lineEdit_22.text())
values.append(self.lineEdit_23.text())
values.append(self.lineEdit.text())
values.append(self.lineEdit_2.text())
values.append(self.lineEdit_3.text())
values.append(self.lineEdit_4.text())
values.append(self.lineEdit_5.text())
values.append(self.lineEdit_6.text())
values.append(self.lineEdit_7.text())
values.append(self.lineEdit_8.text())
```

En el siguiente segmento del mismo método, se incluye el código requerido cuando el usuario desea adjuntar la salida gráfica o miniatura del mapa al que está elaborando el metadato.

```
if self.lineEdit_24.text()!='':
    with Image.open(self.lineEdit_24.text()) as image:
        image.thumbnail((280,280),Image.ANTIALIAS)
        output = io.BytesIO()
        image.save(output, format='PNG')
    values.append('\\x%s'%output.getvalue().hex())
else:
    values.append('')
```

De este en particular, es conveniente mencionar que el proceso requiere un ajuste de tamaño de la imagen que es cargada por el usuario a 280x280, dado que, como se convierte en un valor hexadecimal (cadena de caracteres muy extensa) y al ser tan largo, el programa no es capaz de dejarla en memoria, por lo que el aplicativo pierde eficiencia.

Con ese tamaño, es posible recuperar fácilmente la imagen por la aplicación, sin que pierda rendimiento. El condicional valida la existencia de la imagen en la ruta indicada por el usuario, en caso contrario lo deja vacío.

El método insert() finaliza con la verificación de cada una de las columnas, es decir, si hay una columna que está vacía, ésta no se inserta. Esta tarea se logra mediante el siguiente ciclo for:

```
for i in range(len(values)-1,-1,-1):
    if values[i]=='':
        values.pop(i)
        columns.pop(i)
```

El recorrido es se hace de atrás hacia adelante. El condicional verifica si el valor está vacío lo elimina con la función pop de la lista de columns y values respectivamente.

```
self.thread = QThread()
self.worker = Worker()
self.worker.moveToThread(self.thread)
self.worker.finished.connect(self.thread.quit)
self.worker.finished.connect(self.worker.deleteLater)
self.thread.finished.connect(self.thread.deleteLater)
self.worker.completed[str,str,str].connect(self.insert_complete)
self.worker.failed.connect(self.insert_failed)
self.thread.start()
self.worker.insert(self.port,self.access_token,columns,values)
```

Las anteriores sentencias permiten iniciar la clase Worker mediante la cual la API hace conexión, su ejecución en segundo plano. La clase Worker contiene los métodos requeridos por la API para la manipulación de los registros de metadato a partir de los eventos que suceden de la interacción del usuario con la interfaz gráfica asociados a la gestión (crear, eliminar, consultar y modificar).

```
class Worker(QObject):

    failed=pyqtSignal(int,str)
    finished = pyqtSignal()
    completed=pyqtSignal([str,str],[str,str,str],[requests.models.Response])

    def insert(self, port, access_token, columns, values):
        try:
            connection = requests.post('%sdb/metadata/create' %port,
                                       headers={"access-token":access_token},
                                       json={"columns":columns,"values":values})
            if connection.status_code==201:
                self.completed[str,str,str].emit(connection.json()['message'],
                                                 connection.json()['GUID'],
                                                 connection.json()['QR'])
            else:
                self.failed.emit(connection.status_code, connection.json()['message'])
        except Exception as e:
            self.failed.emit(0,str(e))

        self.finished.emit()

    def delete(self, port, access_token, guids):
        messages=''
        errors=''
        for i in guids:
            try:
                connection = requests.delete('%sdb/metadata/delete/%s' %(port,i),
                                             headers={"access-token":access_token})
                if connection.status_code==200:
                    messages+=connection.json()['message']+'<br>'
                else:
                    errors+='GUID'+i+': '+connection.json()['message']+'<br>'
            except Exception as e:
                errors+='GUID'+i+': '+str(e)+'<br>'
        if messages=='':
            self.failed(0, errors)
        else:
            self.completed[str,str].emit(messages, errors)
        self.finished.emit()
```

```
    def query(self, port, access_token, fil):
        try:
            query=requests.get(port+"db/metadata/query?filter="+fil,
                               headers={"access-token":access_token})
            if query.status_code==200:
                self.completed[requests.models.Response].emit(query)
            else:
                self.failed.emit(query.status_code, query.json()['message'])
        except Exception as e:
            self.failed.emit(0, str(e))
        self.finished.emit()

    def update(self, port, access_token, guids, json):
        messages=''
        errors=''
        for i in guids:
            try:
                connection = requests.patch('%sdb/metadata/update/%s' %(port, i),
                                            headers={"access-token":access_token},
                                            json=json[guids.index(i)])
                if connection.status_code==200:
                    messages+=connection.json()['message']+'<br>'
                else:
                    errors+='GUID'+i+': '+connection.json()['message']+'<br>'
            except Exception as e:
                errors+='GUID'+i+': '+str(e)+'<br>'
        if messages=='':
            self.failed(0, errors)
        else:
            self.completed[str,str].emit(messages, errors)
        self.finished.emit()
```

El código QR se va a obtener a partir del valor hexadecimal que crea la API, por lo que mediante el método insert_complete() se toman los bytes del hexadecimal del código, y luego se convertirán a un fichero png más legible para el usuario final. El despliegue de la imagen lo hace con el cuadro de dialogo QR_dialog tipo ventana emergente.

```
def insert_complete(self,message,guid,qr):
    self.gif.stop()
    self.label_3.setVisible(False)
    self.label_5.setVisible(False)
    self.scrollArea.setEnabled(True)
    data = bytes.fromhex(qr)
    with open('temp/'+guid+'.png', 'wb') as file:
        file.write(data)
    self.QR_dialog=QR_Dialog(self,guid,"INSERTADO CON EXITO<br>GUID: "+guid)
    self.QR_dialog.finished.connect(self.QR_dialog.deleteLater)
    self.QR_dialog.finished.connect(self.QR_close)
    self.QR_dialog.show()
```

Otro de los procesos importantes de la aplicación de gestión corresponde al de consultar el metadato, query que puede hacerse a partir de unas palabras claves o un atributo en específico (se controlan con un RadioButton). El método encargado del filtro en cuestión se ilustra en seguida:

```
def filtrar(self):
    self.tabWidget.currentWidget().setEnabled(False)
    self.gif_2.start()
    self.label_4.setVisible(True)
    self.tableWidget.clear()
    self.tableWidget.setRowCount(0)
    self.tableWidget.setColumnCount(0)
    self.pushButton.setVisible(False)
    self.pushButton_2.setVisible(False)
    switcher = {
        "es": "LIKE '{}'",
        "no es": "NOT LIKE '{}'",
        "comienza por": "LIKE '{}%'",
        "termina por": "LIKE '%{}'",
        "contiene": "LIKE '%{}%'",
        "no contiene": "NOT LIKE '%{}%'",
        "está vacío": "IS NULL",
        "no está vacío": "IS NOT NULL"
    }
    if self.radioButton_3.isChecked():
        if self.lineEdit_25.text()=='' and self.comboBox_7.currentText() not in ('está vacío', 'no está vacío'):
            QMessageBox.warning(self,'Error','No se ingreso ninguna palabra clave')
            self.gif_2.stop()
            self.label_4.setVisible(False)
            self.tabWidget.currentWidget().setEnabled(True)
        else:
            self.thread = QThread()
            self.worker = Worker()
            self.worker.moveToThread(self.thread)
            self.worker.finished.connect(self.thread.quit)
            self.worker.finished.connect(self.worker.deleteLater)
            self.thread.finished.connect(self.thread.deleteLater)
            self.worker.completed[requests.models.Response].connect(self.query_complete)
            self.worker.failed.connect(self.query_failed)
            self.thread.start()
            self.worker.query(self.port, self.access_token, "WHERE palabras_clave "+switcher.get(self.comboBox_7.currentText(), "")
```

Con el diccionario switcher del método anterior, se está haciendo la correspondencia de las opciones definidas wn el combobox de opciones de consulta y los diferentes predicados requeridos para ejecutar la consulta en SQL sobre la base de datos. Así mismo, el primer condicional permite controlar la consulta a partir de palabras claves.

```
elif self.radioButton_4.isChecked():
    switcher_column={
        "Título del Recurso": "Titulo",
        'Categoría Temática': "Categoria_Tematica",
        'Fecha de creación del Recurso': 'Fecha_Creacion_Recurso'
    }
    if self.comboBox_6.currentText() in ('Título del Recurso', 'Categoría Temática') and self.comboBox_7.currentText() not in (
        QMessageBox.warning(self,'Error','No se ingreso dato a buscar')
        self.gif_2.stop()
        self.label_4.setVisible(False)
        self.tabWidget.currentWidget().setEnabled(True)
    elif self.comboBox_6.currentText() in ('Título del Recurso', 'Categoría Temática'):
        self.thread = QThread()
        self.worker = Worker()
        self.worker.moveToThread(self.thread)
        self.worker.finished.connect(self.thread.quit)
        self.worker.finished.connect(self.worker.deleteLater)
        self.thread.finished.connect(self.thread.deleteLater)
        self.worker.completed[requests.models.Response].connect(self.query_complete)
        self.worker.failed.connect(self.query_failed)
        self.thread.start()
        self.worker.query(self.port, self.access_token, "WHERE %s %s "%(switcher_column.get(self.comboBox_6.currentText(), ""),
    elif self.comboBox_6.currentText()=='Fecha de creación del Recurso':
        switcher_date={
        self.thread = QThread()
        self.worker = Worker()
        self.worker.moveToThread(self.thread)
        self.worker.finished.connect(self.thread.quit)
        self.worker.finished.connect(self.worker.deleteLater)
        self.thread.finished.connect(self.thread.deleteLater)
        self.worker.completed[requests.models.Response].connect(self.query_complete)
        self.worker.failed.connect(self.query_failed)
        self.thread.start()
        self.worker.query(self.port, self.access_token, "WHERE %s %s "%(switcher_column.get(self.comboBox_6.currentText(), ""),
```

Con el bloque de código anterior, es posible asignar funcionalidad a la opción de filtrado por atributo considerando el título del recurso, la categoría temática o la fecha de creación del producto cartográfico. Nótese que la consulta SQL que consume la API se construye en método de la Clase Worker (worker.query).

A partir de esta consulta el usuario gestor puede modificar algún valor previamente almacenado o eliminar un registro de metadato completo. La siguiente captura expone el método de borrado de un registro al oprimir el botón de eliminar metadato:

```
def button_del(self, row):
    res=QMessageBox.information(self,'Borrar registro',"<p align='center'>¿Está seguro de querer eliminar este registro?</p>",
                                QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No, QMessageBox.StandardButton.No)
    if res==QMessageBox.StandardButton.Yes:
        self.gif_2.start()
        self.label_4.setVisible(True)
        self.tabWidget.currentWidget().setEnabled(False)
        self.thread = QThread()
        self.worker = Worker()
        self.worker.moveToThread(self.thread)
        self.worker.finished.connect(self.thread.quit)
        self.worker.finished.connect(self.worker.deleteLater)
        self.thread.finished.connect(self.thread.deleteLater)
        self.worker.completed[str,str].connect(self.delete_complete)
        self.worker.failed.connect(self.delete_failed)
        self.thread.start()
        self.worker.delete(self.port, self.access_token,[self.tableWidget.item(row,1).text()])
```

El método a continuación permite la modificación de registros de metadatos desde la interfaz gráfica:

```
def update_db(self):
    if len(self.edit)==0:
        QMessageBox.information(self,'Actualización',"<p align='center'>No hubo modificaciones</p>")
    else:
        self.tabWidget.setEnabled(False)
        self.gif_2.start()
        self.label_4.setVisible(True)
        columns = ['imagen', 'metadatoGUID', 'fecha_creacion_metadato', 'id_recurso', 'fecha_creacion_recurso', 'titulo', 'resumen
                   'proposito', 'proyecto', 'palabras_clave', 'categoria_tematica', 'escala', 'idioma', 'cubrimiento_espacial',
                   'cubrimiento_temporal', 'edicion', 'mantenimiento_actualizacion', 'fuentes_informacion', 'alcance_datos', 'cali
                   'historia', 'representacion_espacial', 'referencia_espacial', 'nivel_resolucion', 'contenido', 'vigencia',
                   'representacion_cartografica', 'estandar_version', 'creditos_recurso', 'contacto', 'instrucciones_contacto',
                   'medios_digitales_recurso', 'ruta_almacenamiento', 'formato_datos', 'distribuidor', 'transferencia', 'restricci
                   'licencia_uso', 'restricciones_seguridad', 'qr']
        json=[]
        index=[]
        guids=[]
        for i in self.edit:
            if i[0] in index:
                json[index.index(i[0])].update({columns[i[1]] : self.tableWidget.item(i[0], i[1]).text()})
            else:
                json.append({columns[i[1]] : self.tableWidget.item(i[0], i[1]).text()})
                index.append(i[0])
                guids.append(self.tableWidget.item(i[0], 1).text())

        self.thread = QThread()
        self.worker = Worker()
        self.worker.moveToThread(self.thread)
        self.worker.finished.connect(self.thread.quit)
        self.worker.finished.connect(self.worker.deleteLater)
        self.thread.finished.connect(self.thread.deleteLater)
        self.worker.completed[str,str].connect(self.update_complete)
        self.worker.failed.connect(self.update_failed)
        self.thread.start()
        self.worker.update(self.port, self.access_token, guids, json)
```

La interfaz le envía a la API una lista de los identificadores de metadatos y un archivo json con los valores de los campos a modificar. La modificación se consigue con el método de la clase Worker (worker.update).

7.5. Creación de la app móvil para la lectura de códigos QR y despliegue del metadato geográfico en pantalla

A continuación, se presenta la codificación para la interfaz de inicio. Esta no requiere autenticación por parte del usuario final para acceder a las funcionalidades de lectura, despliegue y exportación del metadato.

El siguiente segmento de código se procede a definir la clase, inicializar variables y conectar con la actividad (interfaz de login), los botones y demás elementos de visualización.

```
public class Login extends AppCompatActivity {

    Button booting;
    EditText host;
    ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        booting = findViewById(R.id.loginbutton);
        host=findViewById(R.id.host);
        progressBar=findViewById(R.id.progressBar);
        progressBar.setVisibility(View.INVISIBLE);
        SharedPreferences sharedPref = Login.this.getPreferences(Context.MODE_PRIVATE);
        String defaultHost = sharedPref.getString( s: "host", s1: "http://192.168.0.4/");
        host.setText(defaultHost);
        booting.setOnClickListener(this::onClick);
        IntentFilter filter = new IntentFilter();
        filter.addAction(MyIntentService.ACTION_CONNECTION);
        ProgressReceiver rcv = new ProgressReceiver();
        registerReceiver(rcv, filter);
    }
```

El IntentFilter permite la conexión de la clase login con un servicio que se ejecuta en segundo plano denominado IntentService, el cual ejecuta todos los queries, y la conexión a la API. El siguiente bloque de código entra en ejecución una vez se hace clic sobre el botón de inicio:

```
private void onClick(View view){
    ConnectivityManager connMgr = (ConnectivityManager)
            getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
    if (networkInfo != null && networkInfo.isConnected()) {
        progressBar.setVisibility(View.VISIBLE);
        booting.setEnabled(false);
        Intent myIntent = new Intent( packageContext: Login.this, MyIntentService.class);
        myIntent.setAction("ACTION_CONNECTION");
        myIntent.putExtra( name: "url", value: host.getText().toString()+"db/connection/consulta?p=Consulta$");
        startService(myIntent);
    } else {
        Toast.makeText( context: this,  text: "Sin conexión a internet", Toast.LENGTH_SHORT).show();
    }
}
```

De esta manera, al hacer clic lo primero que se verifica es la conexión a internet. Mediante la cláusula condicional se valida si existe tal, para hacer las peticiones a la API e inicio el servicio (intent), de lo contrario mostrará un mensaje de error.

```java
public class ProgressReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals(MyIntentService.ACTION_CONNECTION)){
            booting.setEnabled(true);
            progressBar.setVisibility(View.INVISIBLE);
            int code=intent.getIntExtra( name: "code", defaultValue: 500);
            Toast.makeText( context: Login.this,intent.getStringExtra( name: "message"),Toast.LENGTH_SHORT).show();
            if (code==200) {
                SharedPreferences sharedPref = Login.this.getPreferences(Context.MODE_PRIVATE);
                SharedPreferences.Editor editor = sharedPref.edit();
                editor.putString( s: "host", host.getText().toString());
                editor.apply();
                Intent myIntent = new Intent( packageContext: Login.this, MainActivity.class);
                myIntent.putExtra( name: "access_token",intent.getStringExtra( name: "access_token"));
                myIntent.putExtra( name: "host",host.getText().toString());
                myIntent.setFlags(FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(myIntent);
            }
        }

    }
}
```

Mediante el PogressReceiver va a recibir la información del intent. Con la clasula condicional se valida el tipo de mensaje de retorno, para este caso, el 200 (ejecución exitosa), para guardar las credenciales de inicio y conexión a la API. Por último, se da paso a la siguiente conexión con la segunda actividad de la app (main activity) encargada del escaneo del código QR y despliegue en pantalla.

El IntentService, como ya se mencionó anteriormente, invoca el servicio en segundo plano. Una vez ejecute el proceso y envíe los resultados, se elimina. Por defecto, los servicios que pueden ser filtrados del intent son: ACTION_CONNECTION, ACTION_QUERY y ACTION_FILE.

```java
public class MyIntentService extends IntentService{

    public static final String ACTION_CONNECTION = "com.example.metadatoapp.action.CONNECTION";
    public static final String ACTION_QUERY = "com.example.metadatoapp.action.QUERY";
    public static final String ACTION_FILE = "com.example.metadatoapp.action.FILE";
    private static final Font TITLE_FONT = new Font(Font.FontFamily.HELVETICA,  size: 16f, Font.BOLD);

    String response;
    int code;
    String message;
    String access_token;

    public MyIntentService() { super( name: "MyIntentService"); }
```

Posteriormente se obtendrá la acción, mediante un switch para saber cómo debe proceder el servicio, según se presenta en seguida:

```
protected void onHandleIntent(Intent intent){
    final String action=intent.getAction();
    switch (action) {
        case "ACTION_FILE": {
            String JSON = intent.getStringExtra( name: "JSON");
            Uri uri = intent.getParcelableExtra( name: "uri");
            @SuppressLint("Recycle") Cursor returnCursor =
                    getContentResolver().query(uri, projection: null, selection: null, selectionArgs: null, sortOrder: null);
            int nameIndex = returnCursor.getColumnIndex(OpenableColumns.DISPLAY_NAME);
            returnCursor.moveToFirst();
            if (returnCursor.getString(nameIndex).contains("pdf")) {
                createFilePDF(JSON, uri);
            } else {
                createFileXML(JSON, uri);
            }
            Intent bcIntent = new Intent();
            bcIntent.setAction(ACTION_FILE);
            bcIntent.putExtra( name: "message", message);
            sendBroadcast(bcIntent);
            break;
```

La requerida en el inicio de sesión ACTION_CONNECTION es la siguiente:

```
case "ACTION_CONNECTION": {
    String url = intent.getStringExtra( name: "url");
    db_connection(url);
    Intent bcIntent = new Intent();
    bcIntent.setAction(ACTION_CONNECTION);
    bcIntent.putExtra( name: "code", code);
    bcIntent.putExtra( name: "message", message);
    bcIntent.putExtra( name: "access_token", this.access_token);
    sendBroadcast(bcIntent);
    break;
}
```

Así mismo, la requerida para la acción de consulta ACTION_QUERY es:

```
case "ACTION_QUERY": {
    String url = intent.getStringExtra( name: "url");
    String access_token = intent.getStringExtra( name: "access_token");
    db_query(url, access_token);
    Intent bcIntent = new Intent();
    bcIntent.setAction(ACTION_QUERY);
    bcIntent.putExtra( name: "code", code);
    bcIntent.putExtra( name: "message", message);
    bcIntent.putExtra( name: "response", response);
    sendBroadcast(bcIntent);
    break;
}
```

Mediante este bloque de código, es posible consumir la API mediante la URL. Con el HTTP se realiza la respectiva conexión y se obtiene el código de respuesta. Como las respuestas de la API se entregan mediante archivos json, su lectura se hace mediante BufferedReader línea a línea.

Finalmente se convierte a una variable JSONObject para que pueda ser fácilmente leído por la app. Mediante el flujo de excepción se valida el control de errores, exponiendo en pantalla el mensaje de éste.

```
private void db_connection(String url) {
    StringBuilder json= new StringBuilder();
    try{
        URL http_url = new URL(url);
        HttpURLConnection httpURLConnection = (HttpURLConnection) http_url.openConnection();
        code=httpURLConnection.getResponseCode();
        InputStream inputStream = httpURLConnection.getInputStream();
        BufferedReader responseBodyReader =
                new BufferedReader(new InputStreamReader(inputStream, StandardCharsets.UTF_8));
        String line;
        while ((line=responseBodyReader.readLine())!=null)
            json.append(line);
        JSONObject jsonObject = new JSONObject(json.toString());
        message=jsonObject.getString( name: "message");
        if (code == 200) {
            if (json.length() > 0) {
                access_token=jsonObject.getString( name: "access_token");
            }
        }
        httpURLConnection.disconnect();
    } catch (Exception e) {
        if(code==401){
            message="No se pudo conectar a la Base de Datos";
        }else{
            code = 500;
            message = e.toString();
        }
    }
}
```

Las siguientes líneas de código corresponde a la consulta que hace el servicio a la base de datos a partir del identificador del metadato codificado en el QR:

```
private void db_query(String url, String access_token){
    StringBuilder json= new StringBuilder();
    try{
        URL http_url = new URL(url);
        HttpURLConnection httpURLConnection = (HttpURLConnection) http_url.openConnection();
        httpURLConnection.setRequestProperty("access-token",access_token);
        code=httpURLConnection.getResponseCode();
        InputStream inputStream = httpURLConnection.getInputStream();
        BufferedReader responseBodyReader =
                new BufferedReader(new InputStreamReader(inputStream, StandardCharsets.UTF_8));
        String line;
        while ((line=responseBodyReader.readLine())!=null)
            json.append(line);
        if ( httpURLConnection.getResponseCode() == 200) {
            message="Consulta realizada";
            response=json.toString();
        }else{
            JSONObject jsonObject = new JSONObject(json.toString());
            message=jsonObject.getString( name: "message");
        }
```

```
} catch (Exception e) {
    if(code==400){
        message="Error en Base de Datos";
    }else if(code==401){
        message="Sin conexión a Base de Datos establecida";
    }else if(code==402){
        message="Token inválido";
    }else if(code==403){
        message="QR no se encuentra en Base de Datos";
    }else {
        code = 500;
        message = e.toString();
    }
}
```

En cuanto a la actividad principal de la aplicación (main activity), obsérvese que en la declaración de variables de la clase se tienen las variables que almacenan el resultado de la actividad del login (acces_token y host) , una variable TableLayout metadata en donde se desplegará el metadato geográfico formateado en una tabla una vez se escanee, una variable string para almacenar el identificador del metadato en base de datos (metadataGUID) el cual se empleará para hacer la consulta y su respectivo resultado JSON.

```
public class MainActivity extends AppCompatActivity {

    Button btnScan;
    Button btnSave;
    Button btnXML;
    String access_token;
    String host;
    TableLayout metadata;
    ProgressBar progressBar;
    String metadataGUID;
    Uri uri;
    String JSON;
```

Con el método OnCreate se procede a conectar los botones de la interfaz principal (activity_main) con sus respectivos métodos (tareas), se obtiene el acces_token y host que envían la actividad login. Adicionalmente, se hace el filtrado del servicio (intent) recuperando los resultados de las tareas ACTION_QUERY y ACTION_FILE

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    btnScan = findViewById(R.id.btScan);
    btnSave= findViewById(R.id.save);
    btnXML = findViewById(R.id.xml);
    btnSave.setVisibility(View.INVISIBLE);
    btnXML.setVisibility(View.INVISIBLE);
    metadata=findViewById(R.id.tableLayout);
    progressBar=findViewById(R.id.progressBar2);
    progressBar.setVisibility(View.INVISIBLE);

    access_token = getIntent().getStringExtra( name: "access_token");
    host = getIntent().getStringExtra( name: "host");

    btnScan.setOnClickListener(this::onClick);
    btnSave.setOnClickListener(this::save);
    btnXML.setOnClickListener(this::XML);

    IntentFilter filter = new IntentFilter();
    filter.addAction(MyIntentService.ACTION_QUERY);
    filter.addAction(MyIntentService.ACTION_FILE);
    ProgressReceiver rcv = new ProgressReceiver();
    registerReceiver(rcv, filter);
}
```

Al momento de hacer clic, por ejemplo, en el botón de escanear, se ejecutará el siguiente bloque de código:

```
private void onClick(View view){
    ScanOptions options=new ScanOptions();
    options.setDesiredBarcodeFormats(ScanOptions.ALL_CODE_TYPES);
    options.setPrompt("Lector - QR");
    options.setOrientationLocked(false);
    options.setCaptureActivity(PortraitActivity.class);
    options.setCameraId(0);
    options.setBeepEnabled(true);
    options.setBarcodeImageEnabled(true);
    barcodeLauncher.launch(options);
}
```

Del bloque anterior, se ha establecido las opciones de escaneo, es decir, escanear todos los tipos de códigos, no bloquear la orientación, capturar la información encriptada con la PortraitActivityClass encargada de abrir la cámara del dispositivo, establecer el tipo de cámara a utilizar (0-trasera), el sonido a realizar una vez se hace el escaneo.

Con setBarcodeImageEnable se habilita la lectura de imágenes de códigos QR. Finalmente se lanza la lectura con el launch con el método BarcodeLauncher().

```
private final ActivityResultLauncher<ScanOptions> barcodeLauncher=registerForActivityResult(new ScanContract(),
result -> {
    if(result.getContents() == null){
        Toast.makeText( context: this, text: "Lectura cancelada", Toast.LENGTH_SHORT).show();
    }else {
        ConnectivityManager connMgr = (ConnectivityManager)
                getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected()) {
            progressBar.setVisibility(View.VISIBLE);
            btnScan.setEnabled(false);
            Intent myIntent = new Intent( packageContext: MainActivity.this, MyIntentService.class);
            metadataGUID=result.getContents();
            myIntent.setAction("ACTION_QUERY");
            myIntent.putExtra( name: "url", value: host + "db/metadata/query?filter=WHERE metadatoGUID='" + result.getContents() + "'");
            myIntent.putExtra( name: "access_token", access_token);
            startService(myIntent);
        } else {
            Toast.makeText( context: this, text: "Sin conexión a internet", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Si el resultado de la lectura es nulo, implica que la lectura fue cancelada, en caso contrario, se verifica la conexión a internet con el connectivityManager y se establece la conexión con el servicio (intent).

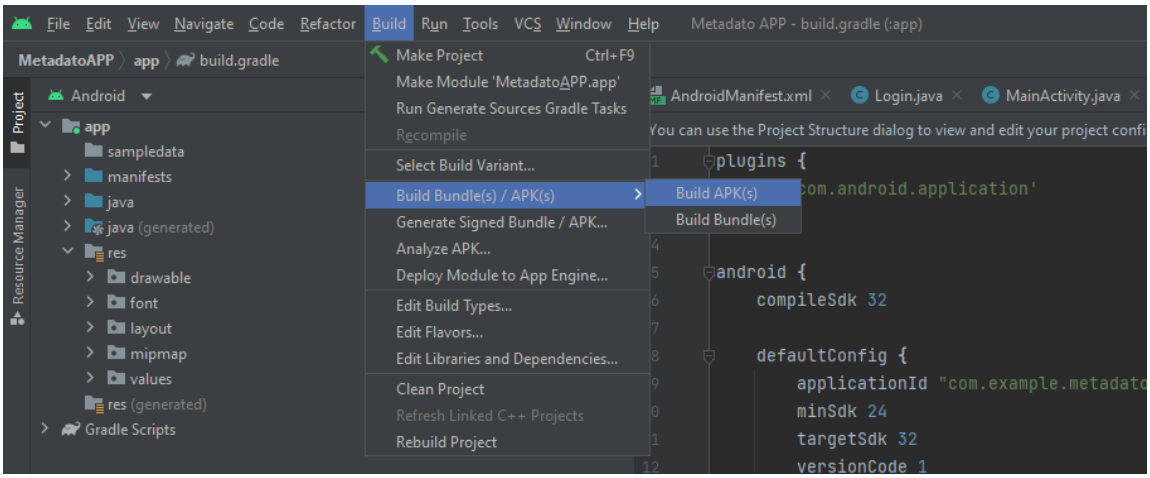
Al momento de leer el código QR, éste contiene la información del identificador del metadato asignado en base de datos, por lo que, al asignarle el resultado de la decodificación (result.getContents()) a la variable metadataGUID para realizar la respectiva consulta a través del ACTION_QUERY.

El resultado de la consulta del metadato y su despliegue es codificado en el método onReceive, como sigue:

```
public void onReceive(Context context, Intent intent) {
    if(intent.getAction().equals(MyIntentService.ACTION_QUERY)){
        progressBar.setVisibility(View.INVISIBLE);
        btnScan.setEnabled(true);
        Toast.makeText( context: MainActivity.this, intent.getStringExtra( name: "message"), Toast.LENGTH_SHORT).show();
        if (intent.getIntExtra( name: "code", defaultValue: 500)==200){
            try {
                metadata.removeAllViews();
                JSON=intent.getStringExtra( name: "response");
                JSONArray jsonArr = new JSONArray(JSON);
                JSONObject jsonObject = jsonArr.getJSONObject( index: 0);
                Iterator<String> keys = jsonObject.keys();
                while(keys.hasNext()) {
                    String key=keys.next();
                    @SuppressLint("InflateParams") View registro = LayoutInflater.from(MainActivity.this).inflate(R.layout.table_layout,
                            root: null, attachToRoot: false);
                    TextView column = registro.findViewById(R.id.column);
                    TextView value = registro.findViewById(R.id.value);
                    if(key.equals("IMAGEN") && !jsonObject.getString(key).contains("null")){
                        byte[] arr = new byte[jsonObject.getString(key).length() / 2];
                        for (int l = 0; l < arr.length; l++) {
                            int index = l * 2;
                            int val = Integer.parseInt(jsonObject.getString(key).substring(index, index + 2), radix: 16);
                            arr[l] = (byte) val;
                        }
                        Bitmap bitmap = BitmapFactory.decodeByteArray(arr, offset: 0, arr.length);
                        ImageView imageView = registro.findViewById(R.id.imageView2);
                        imageView.setImageBitmap(bitmap);
                        imageView.setVisibility(View.VISIBLE);
                        column.setVisibility(View.GONE);
                        value.setVisibility(View.GONE);
                        metadata.addView(registro);
                    }else if (!key.equals("QR")) {
                        column.setText(key);
                        value.setText(jsonObject.getString(key));
                        metadata.addView(registro);
                    }
                }
```

Obteniendo el archivo json de respuesta del intent service, se lee para obtener todas las llaves, las cuales corresponden con los elementos del metadato. Se procede a leer el archivo y a crear la cantidad de filas de tabla requeridas en pantalla para su lectura.

Para generar el archivo APK (ejecutable) y compartir con otros usuarios, se procede de la siguiente manera:



8. Despliegue del prototipo funcional

En esta sección se presenta el funcionamiento de cada uno de los módulos y operaciones codificadas en la fase anterior del prototipo del sistema que permite la gestión de metadatos geográficos utilizando tecnología QR, construido de conformidad a la especificación de requerimientos y a los lineamientos planteados en la fase de diseño.

En primera instancia, es necesario poner en servicio la API para iniciar el plan de pruebas de funcionamiento en el sistema “Metadata”. Por esta razón, se ilustra la instrucción requerida en consola para tal fin:

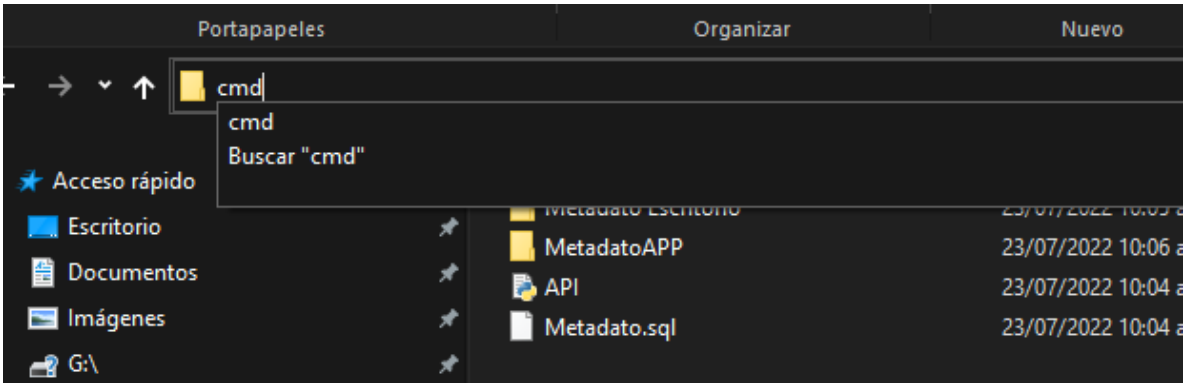


Figura 47. *Accediendo a consola de windows desde el folder del proyecto. Fuente: Elaboración propia.*

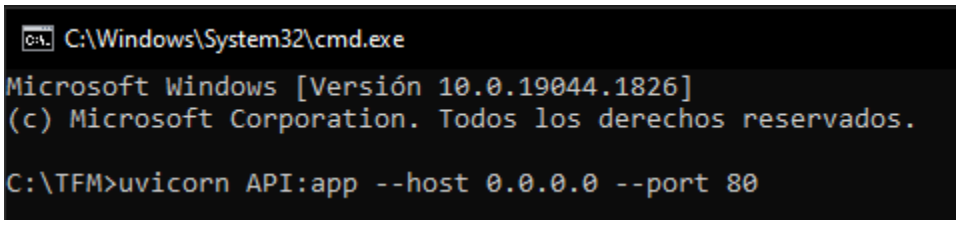


Figura 48. *Puesta en servicio de la API. Fuente: Elaboración propia.*

En seguida se desplegará un mensaje de conexión como sigue:

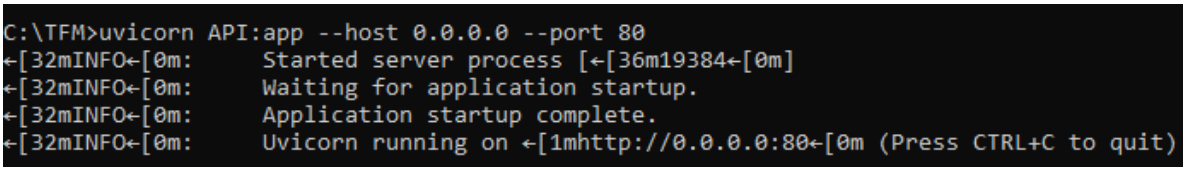


Figura 49. *API en ejecución. Fuente: Elaboración propia.*

Acto seguido, se procede a ingresar a la aplicación de escritorio a través de la cual es posible acceder a las funcionalidades de gestión de metadatos geográficos habilitadas a los usuarios gestores en una organización.

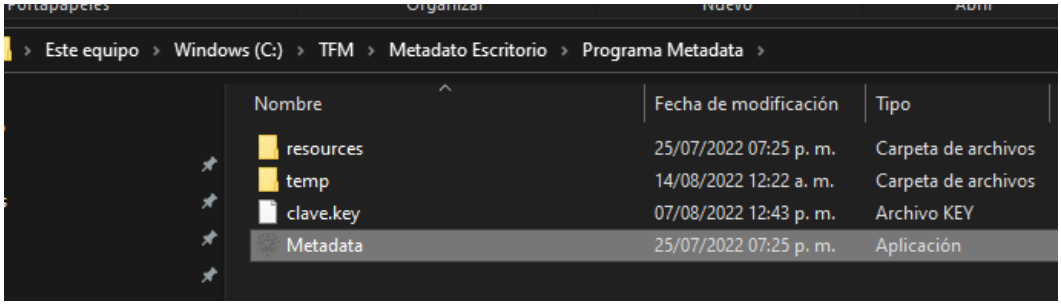


Figura 50. *Ejecutable aplicación de escritorio Metadata. Fuente: Elaboración propia.*

8.1. Presentación del funcionamiento de la aplicación de escritorio Metadata

8.1.1. Módulo de autenticación

El acceso al sistema está controlado a partir de la asignación de unas credenciales a los usuarios Administrador y Gestor, respectivamente. Por esta razón, el primer formulario

requiere la inserción de los datos de usuario, contraseña y dirección IP donde está alojada la API, como se ilustra en seguida:

Figura 51. *Inicio de sesión aplicación de escritorio Metadata. Fuente: Elaboración propia.*

Una vez que el administrador de la base de datos, por ejemplo, ha accedido al sistema, se le presentará la pantalla principal de la aplicación (ver figura 52). Desde este cuadro de dialogo se tendrá acceso a las funcionalidades de gestión disgregadas en dos pestañas. Una dedicada a la captura de información de un recurso cartográfico mediante la creación de un registro de metadato. La otra dedicada a la consulta de información de metadatos almacenada en la base de datos que soporta la aplicación.

Figura 52. *Ventana principal de la aplicación de escritorio Metadata. Fuente: Elaboración propia.*

8.1.2. Módulo gestión de metadatos geográficos

Desde la pestaña Insertar, el usuario gestor podrá crear un nuevo metadato geográfico y generar el respectivo código QR. El formulario diseñado contiene todos los elementos de metadatos seleccionados para documentar un mapa digital de conformidad al esquema de metadatos. La siguiente ilustración expone un ejemplo de creación de un metadato en cuestión:

Figura 53. *Creación de un metadato geográfico para un mapa digital. Fuente: Elaboración propia.*

Habiendo ingresado la información alfanumérica que alimenta los elementos del metadato geográfico, la aplicación procede a generar de manera automática el código QR como se puede comprobar en la figura a continuación:

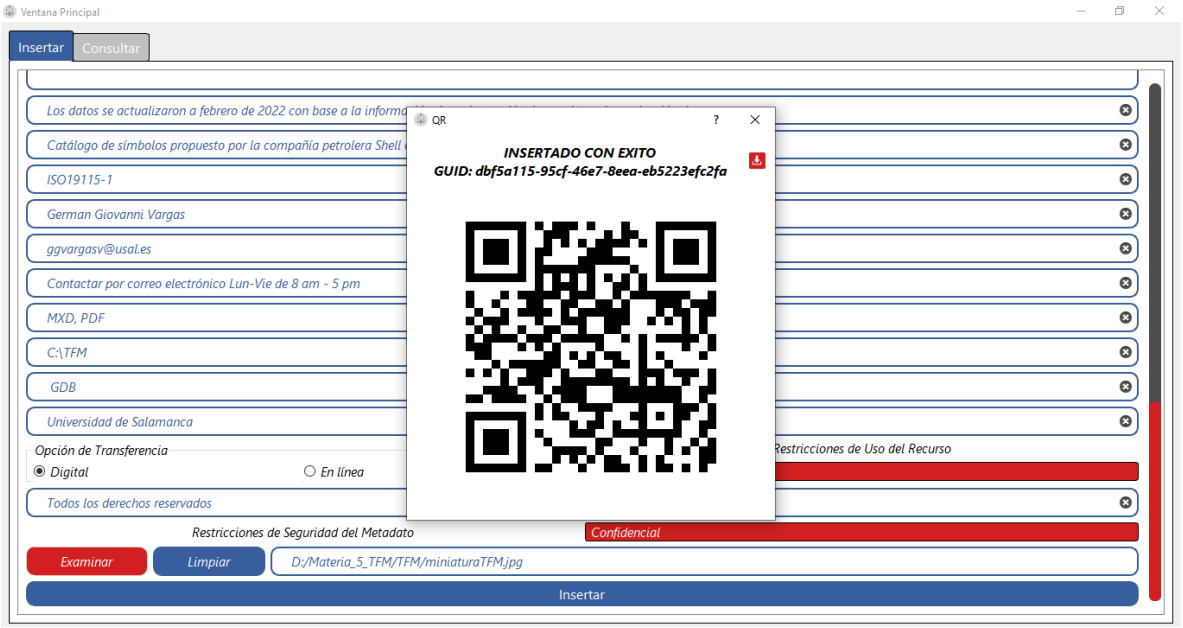


Figura 54. *Generación de código QR del metadato geográfico. Fuente: Elaboración propia.*

Figura 55. *Código QR del metadato geográfico. Fuente: Elaboración propia.*

Desde la pestaña Consulta es posible eliminar un metadato del sistema, ver la información del mismo en detalle o bien editar la información de alguno de sus campos. Dado que para modificar o eliminar un registro de metadato se requiere previamente de una consulta a la base de datos, la imagen a continuación presenta el formulario en donde el gestor puede filtrar registros bien considerando las palabras claves o por un atributo especifico:

Figura 56. *Consulta de un metadato por palabras claves. Fuente: Elaboración propia.*

El usuario gestor dispone de varias opciones de filtrado según el tipo de dato del elemento de metadato que se pretenda aplicar el filtro. Así, por ejemplo, para elementos de metadato cuyos valores son de tipo string (como el campo título del recurso), se dispone de las siguientes clausulas, que internamente la API traduce a expresiones en SQL:

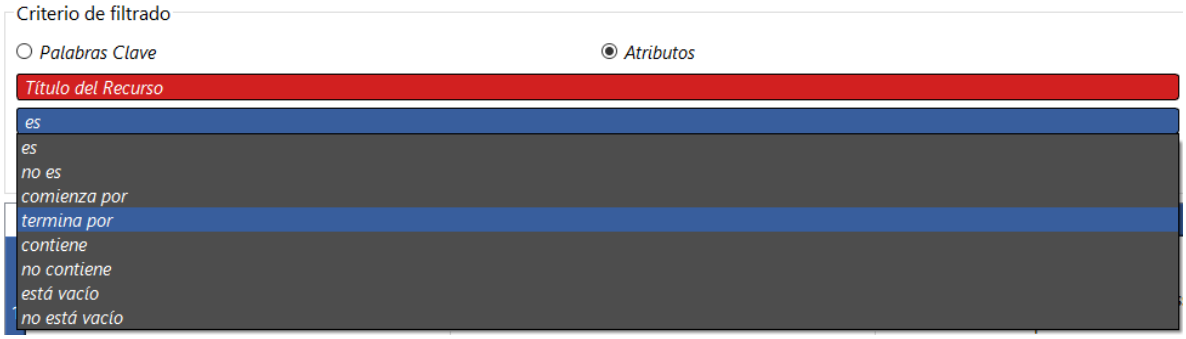


Figura 57. *Cláusulas de consulta para un atributo o campo. Fuente: Elaboración propia.*

***Figura 58**. Consulta de un metadato por atributo. Fuente: Elaboración propia.*

Adviértase en la imagen anterior que, aplicado el filtro, el usuario visualizará un listado con los metadatos existentes que satisfacen el criterio de búsqueda. Cada registro de metadatos de dicha tabla dispone de tres opciones asociadas: la visualización de la miniatura del mapa al que hace referencia en una ventana emergente (icono izquierdo), la visualización del código QR (icono central) y el botón de borrado (icono derecho).

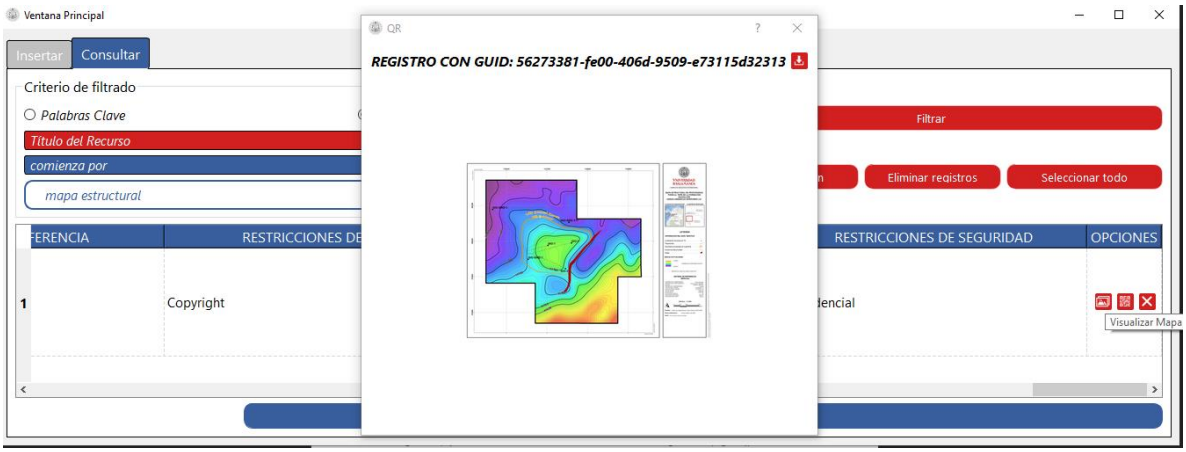


***Figura 59.** Visualización de la miniatura del mapa documentado. Fuente: Elaboración propia.*

Ahora bien, si el usuario gestor desea reemplazar algún valor de un elemento, basta con seleccionar el registro de la tabla en la vista general y hacer doble clic sobre el campo con intención de editar para efectuar dicha modificación. El cambio de valor se conservará una vez se haya oprimido el botón de Guardar ubicado en la parte inferior derecha del formulario.

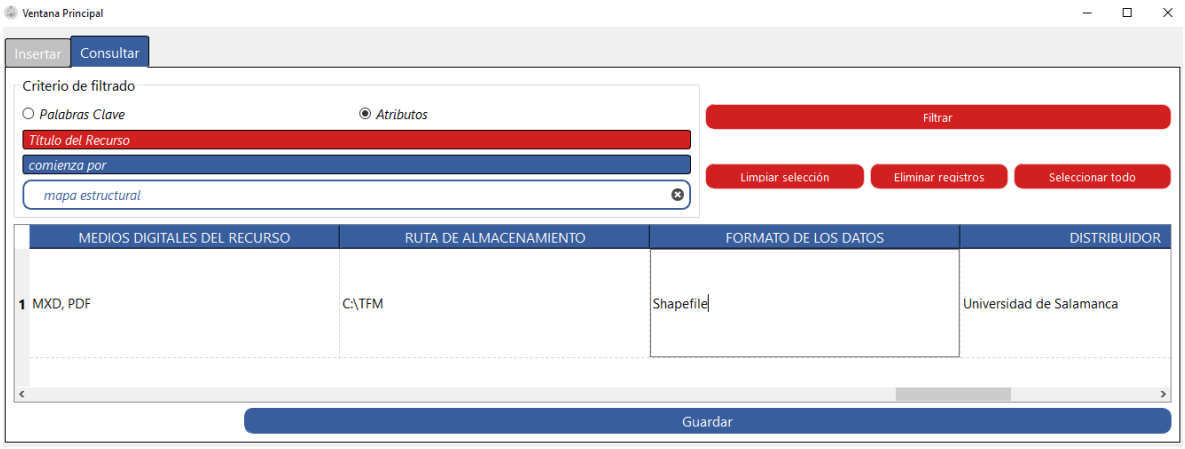


Figura 60. *Modificación de un valor de metadato. Fuente: Elaboración propia.*

Al pulsar el botón guardar, la aplicación presenta un mensaje de modificación satisfactoria, como se observa en la siguiente figura:

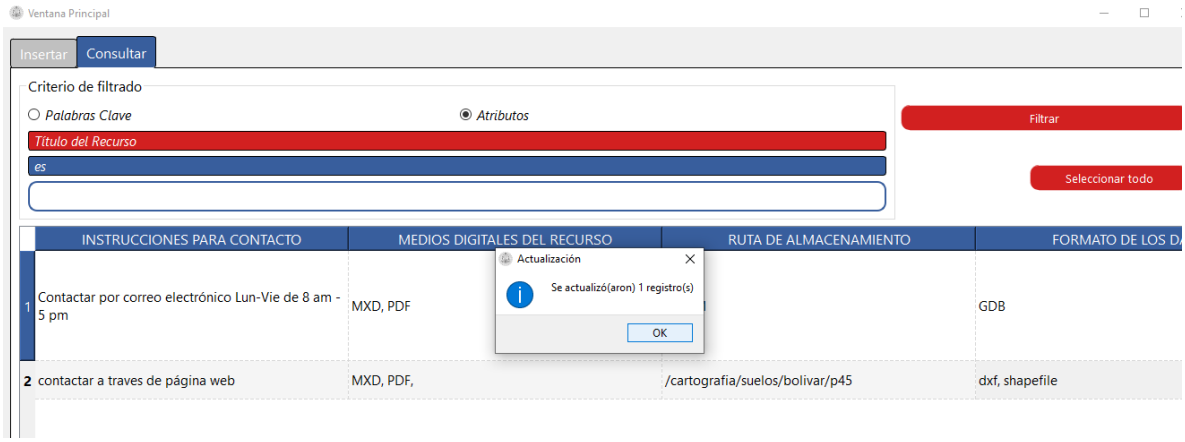


Figura 61. Confirmación de modificación exitosa. Fuente: Elaboración propia.

Para eliminar un registro de metadato, el usuario gestor deberá pinchar sobre el botón con icono x. Al oprimirlo, se desplegará un mensaje de confirmación de borrado en una ventana emergente, como sigue:

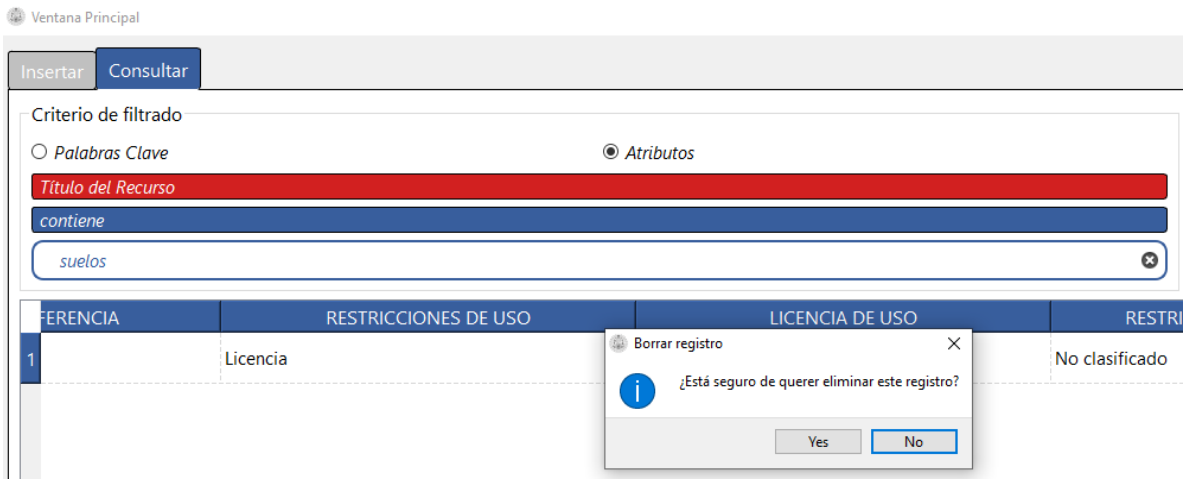


Figura 62. *Eliminación de un registro de metadato geográfico. Fuente: Elaboración propia.*

Para la selección de varios registros de metadatos, se ha facilitado la experiencia del usuario de modo que basta con hacer un clic sobre cada fila manteniendo presionada la tecla shift o haciendo clic directamente sobre el botón seleccionar todo.

Los registros se seleccionarán resaltándose en color azul, como se observa en la figura 62. Sobre dicha selección se puede llevar a cabo la eliminación conjunta (botón eliminar registros) o simplemente limpiar la selección (botón limpiar selección).

Figura 63*. Selección de varios registros de metadatos. Fuente: Elaboración propia.*

Al descargarse el código QR, este puede ser embebido al mapa digital que describe con el fin de que el usuario final pueda acceder fácilmente a la información del metadato a través de la aplicación Metadata del dispositivo móvil.

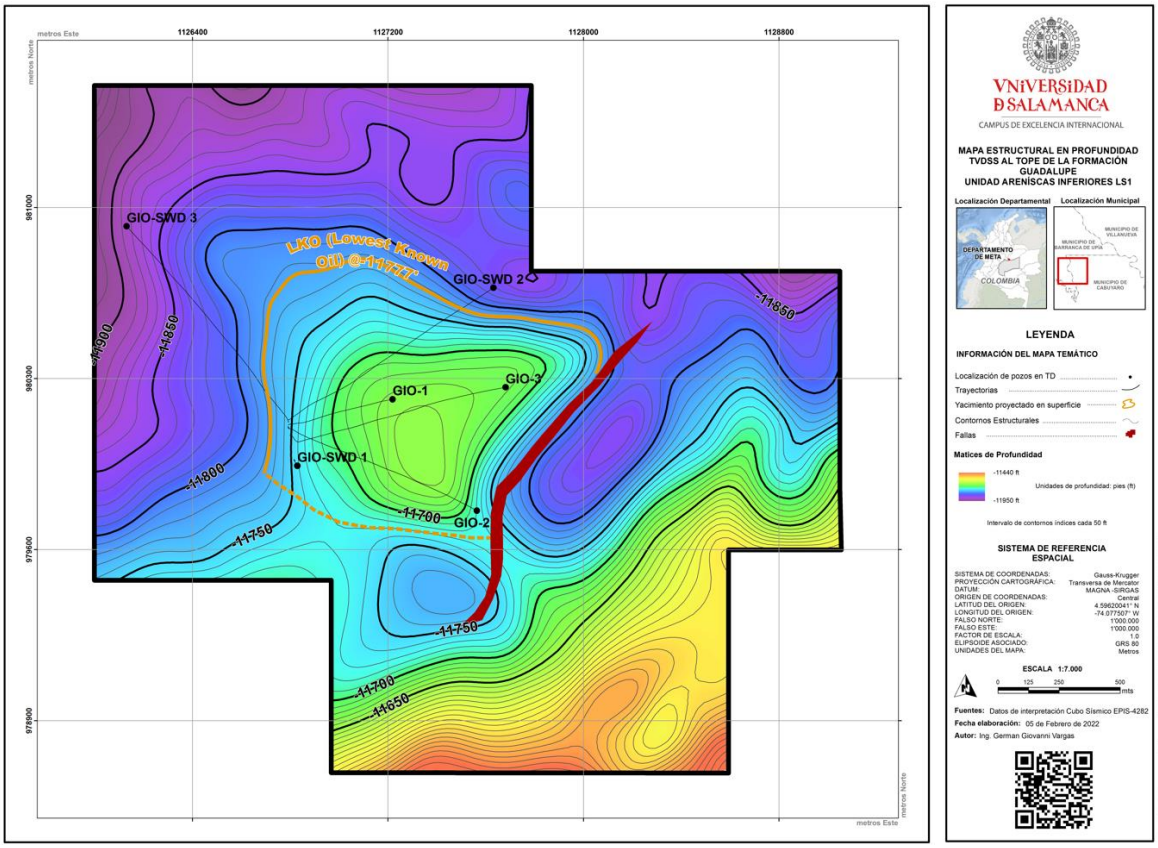


Figura 64. *Ejemplo de mapa digital incorporando el código QR como metadato embebido. Fuente: Elaboración propia.*

8.2. Presentación del funcionamiento de la aplicación móvil Metadata

8.2.1. Módulo de visualización

Una vez descargada e instalada la aplicación (archivo APK) en el dispositivo móvil. Se tiene la siguiente presentación en la galería de aplicaciones:

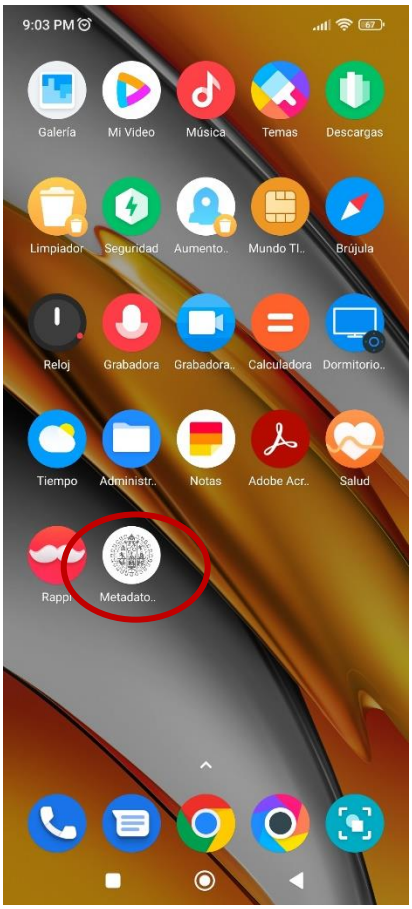


***Figura 65**. Icono de USAL de acceso a la aplicación móvil Metadata. Fuente: Elaboración propia.*

Al ingresar, es posible interactuar con la interfaz que accede a la cámara trasera del dispositivo para la lectura del código. En el momento de ejecutar la aplicación, comenzará a correr un proceso en segundo plano que se encargará de solicitar a la API la comprobación de la conexión a internet y el acceso a la base de datos, para habilitar posteriormente la consulta del metadato una vez se decodifique el identificador encriptado en el QR.

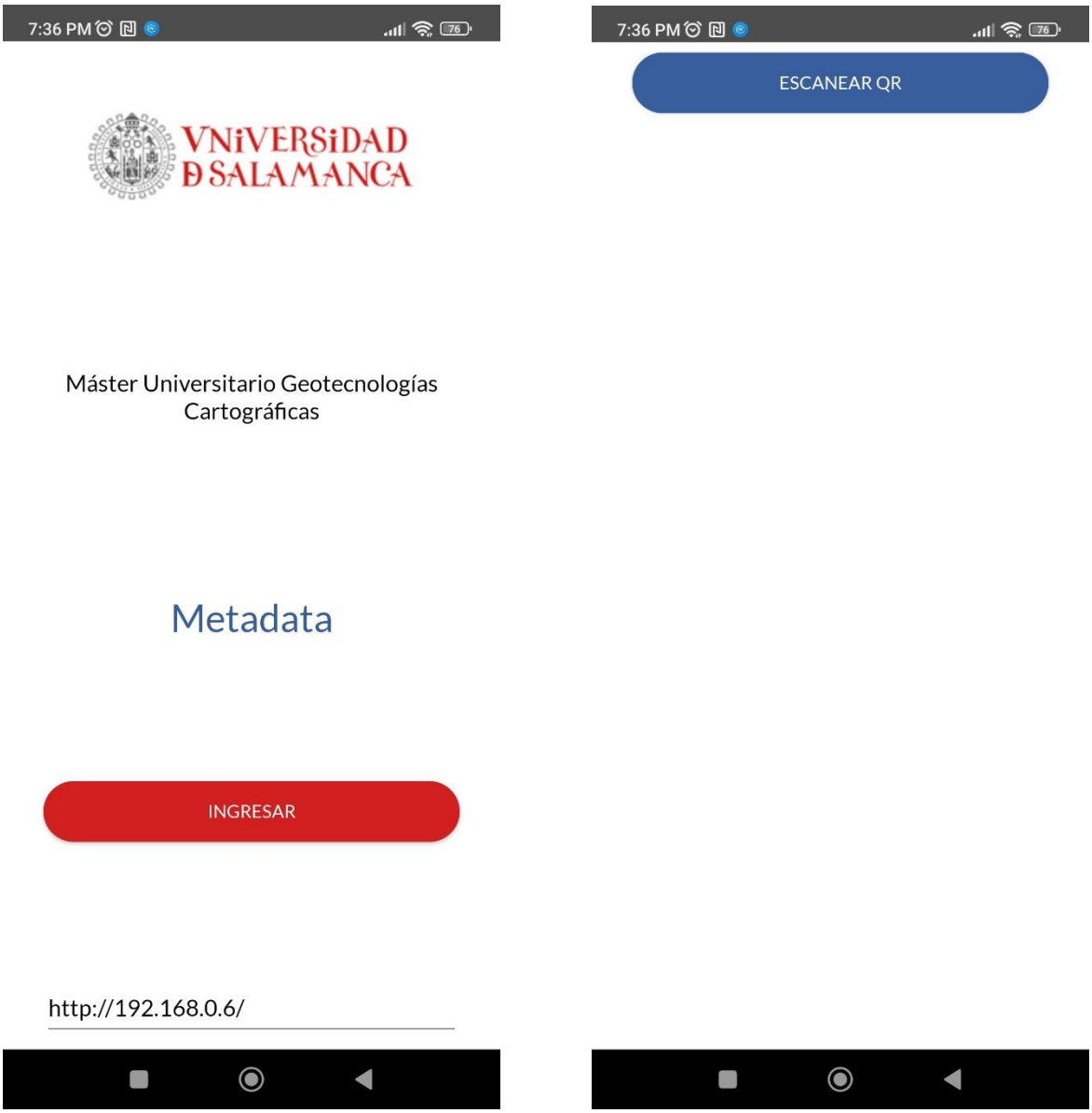


***Figura 66**. Interfaz de acceso a la aplicación móvil(izquierda) y ventana principal de la app Metadata (derecha). Fuente: Elaboración propia.*

Mediante el botón Escanear QR se accede a la función de escáner del dispositivo. Esto se utilizará para recibir la información asociada al código QR generado por el sistema de gestión de metadatos.

La pantalla, mostrada en la figura 67 (izquierda), consiste en la vista de la cámara del dispositivo, con un borde rectangular superpuesto donde se deberá encuadrar el código QR del mapa digital.

Cuando el código es detectado por la aplicación se le comunica al usuario mediante un sonido tipo beat, instante en el cual se envía la información contenida en el código a la API para su consulta. Cuando llega la respuesta por parte de la API, la aplicación expone la información formateada en una tabla, como se presenta en la figura 67 (derecha).

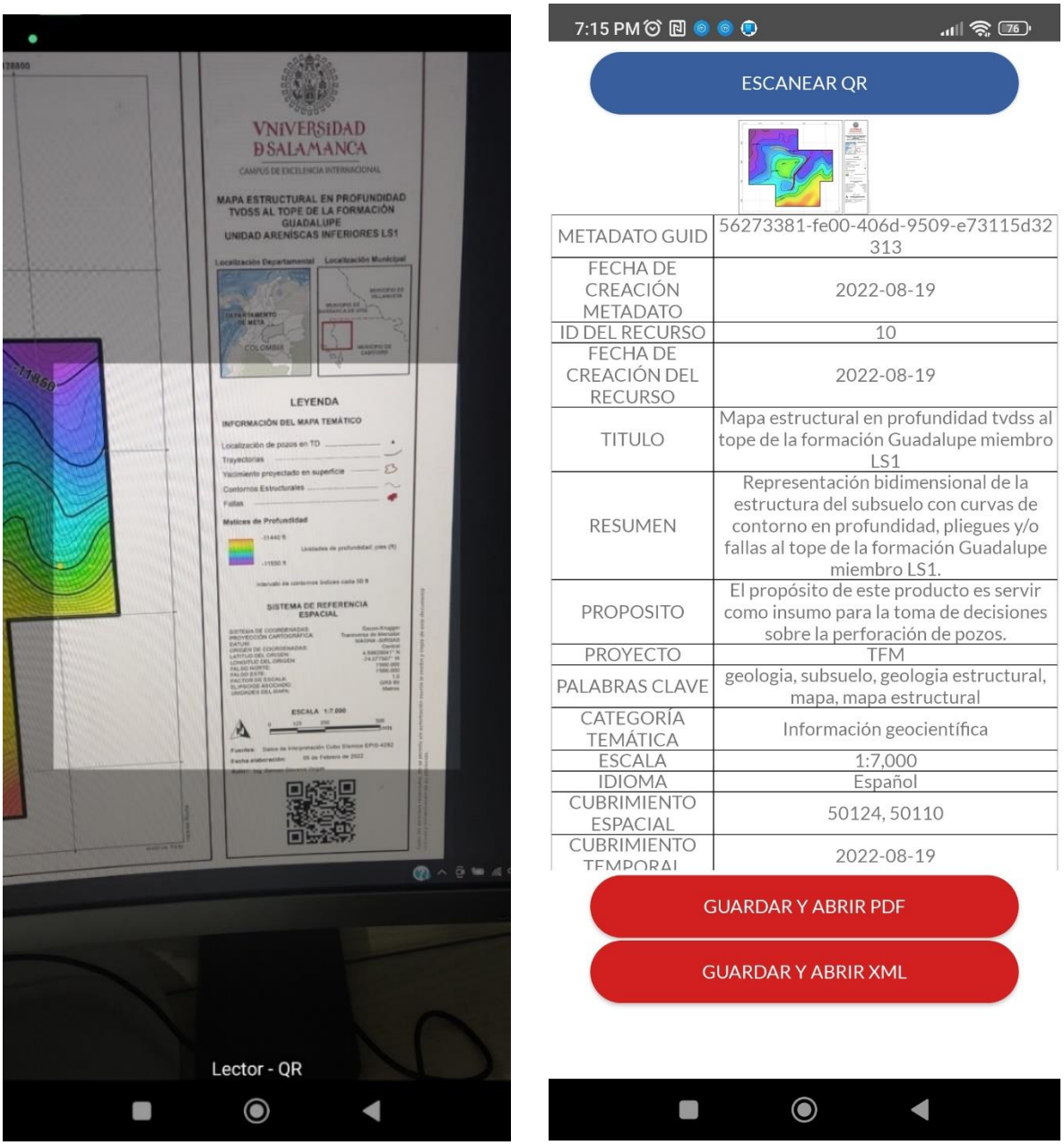


Figura 67. *Lectura y despliegue del metadato codificado en QR. Fuente: Elaboración propia.*

Nótese que, adicionalmente se dispone de dos botones diseñados para exportar la información del metadato y guardar el archivo de representación en formato PDF o XML.

Respecto a la utilización del formato XML como lenguaje de representación de metadatos en el contexto de los metadatos geográficos, cabe destacar que hay un consenso bastante generalizado.

De hecho, los sistemas de catalogación de metadatos geográficos como Geonetwork soporta tres formatos de metadatos: el formato de implementación (dentro de una base de datos o sistema de almacenamiento), el formato de exportación o codificación (diseñado para la transferencia de metadatos entre distintos sistemas), y el formato de presentación (un formato apropiado para ser leído por los usuarios).

Para los dos últimos formatos hay un consenso general respecto al uso de XML dado que es un lenguaje de marcado con reglas estructurales forzadas a través de un fichero de control (Document Type Definition ó DTD) que permite validar la estructura del documento, es decir, comprobar la conformidad respecto al DTD establecido para un estándar de metadatos.

En cuando a la codificación de los metadatos en XML, cabe destacar la especificación técnica ISO/TS 19139:2007. Como ya se mencionó previamente, define la forma de convertir los modelos UML de la norma ISO 19115. De esta manera, un archivo de intercambio de metadatos, acorde con la norma ISO 19115 en formato XML, va a ser un documento XML que siga la sintaxis definida por la especificación técnica ISO 19139:2007.

No obstante, para la representación del metadato geográfico incorporando códigos QR, no se adoptó tal especificación sugiriéndose su adopción como un trabajo futuro de este proyecto mejorando la interoperabilidad con catálogos de metadatos (interpretar la estructura y la semántica de las etiquetas).

9. Conclusiones

Dada la necesidad de facilitar la descripción de mapas producidos al interior de una organización, cuya competencia misional no es la de un instituto cartográfico, se ha propuesto un modelo de gestión de metadatos que puede ser adoptado al interior de una IDE de nivel corporativo e incluso de nivel superior.

A través del desarrollo de este TFM se obtuvo una definición detallada de este modelo, facilitando las tareas de creación, actualización, consulta y eliminación de metadatos asociados a documentos cartográficos, mediante el uso de códigos QR. De esta manera, con la presentación de este proyecto se pretende poner en consideración este mecanismo, para mejorar el proceso de gestión de metadatos geográficos al interior de una IDE corporativa, cuyo principal recurso informacional es el mapa.

Se resalta la aplicación práctica que ofrece la tecnología QR al interior de la gestión de metadatos geográficos en una organización, específicamente en la documentación de mapas digitales, extendiéndose incluso sus potencialidades a la cartografía análoga.

Evidentemente, la posibilidad de acceder y leer los metadatos de un mapa a través de un dispositivo móvil facilita de modo contundente el acceso a los mismos, resolviendo problemas de falta de información y confiabilidad que en ocasiones reviste la manipulación de los documentos cartográficos.

Como es lógico de suponer, acceder de manera fácil, sencilla y oportuna al metadato de un mapa, repercute en mejorar la productividad y competitividad en las organizaciones que recurren a este tipo de recursos para apoyar la toma de decisiones.

Se expuso el diseño detallado del producto de software conforme a la especificación, modelado y dando alcance por completo al levantamiento de requerimientos. Se destaca que, un desarrollo planificado como el presentado en este TFM, permitió que el proceso de construcción del sistema fuera más ágil, eficiente y flexible a las condiciones y tiempos del proyecto.

Por lo anterior, se hace entrega de un prototipo del sistema de gestión construido completamente sobre software libre, obteniéndose un buen rendimiento en su funcionamiento. Se trata de un producto de software de muy bajo costo en su

implementación, el cual brindará la posibilidad de mantenerlo y escalarlo a nuevos requerimientos, dada la amplia documentación que existe de las herramientas y lenguajes de programación utilizados en este TFM.

Esta solución informática no pretende competir con otras herramientas de gestión o catalogadores, pues cada una de ellas atiende necesidades de gestión con propósitos específicos. Pretende hacer sinergia con otras iniciativas en esta cuestión, abordando una estrategia de documentación de mapas, mucho más amigable y sencilla mejorando la experiencia en la elaboración de metadatos (menos complejidad, consumo de tiempo, cambiar la percepción de tarea rutinaria y monótona que predispone negativamente a su elaboración) y en la lectura de ellos (acceso ágil con información útil).

Por todo lo anterior, se destaca que la gestión de metadatos geográficos se constituye como un aspecto relevante en los procesos de gestión de información espacial al interior de una organización. Se concluye que una adecuada gestión de metadatos geográficos facilita 6 operaciones claves:

- Búsqueda (los metadatos deben proporcionar la información suficiente para determinar la existencia de datos de interés dentro del recurso o mapa, o verificar su existencia en un repositorio documental).

- Recuperación (la información que proporcionen los metadatos debe servir a los usuarios para adquirir el recurso que están buscando y que atienda su necesidad de información).

- Transferencia (los metadatos deben brindar la información en un lenguaje común para que puedan ser compartidos por otros usuarios).

- Evaluación (los metadatos deben evaluar la información para determinar si el recurso será útil y puede ser utilizado de conformidad al propósito para el cual fue construido).

- Conservación (los metadatos deben garantizar que los recursos estén documentados, se definan sus responsables y sigan siendo accesibles en el futuro).

- Interoperabilidad (los metadatos deben facilitar esta característica esencial de una IDE corporativa, por lo que deben atender estándares y protocolos en esta materia que permitan su intercambio entre diferentes sistemas).

También se concluye que gestionar adecuadamente estas descripciones, de manera estructurada y estandarizada, son un aspecto fundamental para asegurar un óptimo desempeño de un proyecto GIS o una IDE de nivel corporativo.

En cuanto a mejoras y propuestas de acción futuras, se plantea incorporar algún algoritmo de automatización en la generación del metadato, que recupere directamente de la aplicación donde fue generado el mapa datos como la ruta de almacenamiento o el tipo de formato de distribución. De esta manera, se minimiza la posibilidad de errores y se optimiza tiempos comparativamente con la creación manual.

De igual modo, se sugiere la posibilidad de implementar el estándar ISO19139 como estándar de intercambio del metadato desde la aplicación móvil para que sea perfectamente compatible con catálogos de metadatos como Geonetwork, permitiendo la catalogación y el descubrimiento del mapa al que se hace referencia, pero desde el geoportal de la organización.

Con base en lo anterior, también es conveniente, a partir de la implementación de dicho estándar, se pudiese importar un fichero de metadato XML creado en otra herramienta de edición como catMDEdit. Al respecto, se sugiere que el sistema lea el contenido de cada etiqueta y automáticamente lo asigne a cada atributo del metadato para luego generar el respectivo código QR. Esto implicaría que, antes de ser cargado en el sistema, existiese un validador que garantice la integridad y consistencia del fichero XML a importar.

Otra posible mejora funcional de cara a una implementación rigurosa sería que, en cambio de recurrir a una aplicación de escritorio para la generación de los códigos QR, se planteara la posibilidad de una interfaz web para acometer las mismas tareas. Esto permitirá que el acceso sea más centralizado y que el usuario gestor no deba instalar y almacenar en disco local herramientas para su ejecución. Un framework sugerido es Flask, compatible con el entorno de desarrollo adoptado en este TFM.

Finalmente, un trabajo necesario sería que la aplicación móvil fuera multiplataforma, esto requiere llevar la app creada en este TFM a un entorno de ejecución sobre el sistema IOS, para poder llegar así a prácticamente la totalidad de los consumidores de mapas que pudiesen estar interesados en conocer el metadato de dicho documento.

Referencias

Anaya, D., Cantán, O., Lacasta, J., Nogueras, J., & Zarazaga, F. J. (2002). Interoperabilidad entre estándares de meta-datos geográficos. *Proc. of the II Jornadas de Sistemas de Información Geográfica (JSIG'02)*, 73-86.

Balfanz, D. (2002). Automated geodata analysis and metadata generation. *Visualization and Data Analysis 2002*, *4665*, 285-295.

Beard, K. (1996). A structure for organizing metadata collection. *third international conference. Workshop on Integrating GIS and Environmental Modelling*.

Beltrán Fonollosa, A. (2013). *Descripción, publicación y descubrimiento de recursos georreferenciados* [PhD Thesis]. Universitat Jaume I.

Bernabé-Poveda, M. Á., & López-Vázquez, C. M. (2012). *Fundamentos de las infraestructuras de datos espaciales (IDE)*. BibliotecaOnline SL.

Bodoff, D. (2006). Relevance for browsing, relevance for searching. *Journal of the American Society for Information Science and Technology*, *57*(1), 69-86.

Booth, D., & Liu, C. K. (2007). Web services description language (WSDL) version 2.0 part 0: Primer. *W3C recommendation*, *26*, 39-41.

BORBA, R., STRAUCH, J., Souza, J. M., & COLEMAN, D. (2014). Architectural and technological aspects for the next generation of SDI. *Proceedings AutoCarto 2014 a CaGIS Research Symposium, Pittsburgh, Pennsylvania, USA*.

Callejo, M. Á. M. (2003). Metadatos en los sistemas de información geográfica (ISO-19115). *Escuela Técnica Superior de Ingenieros de Telecomunicación*.

Callejo, M. Á. M., Wachowicz, M., & Poveda, M. Á. B. (2009). *El uso de los metadatos para el desarrollo de un modelo de interoperabilidad para las Infraestructuras de Datos Espaciales* [PhD Thesis]. PhD thesis.

Caplan, P. (1995). *You call it corn, we call it syntax-independent metadata for document-like objects.*

Caplan, P. (2003). *Metadata fundamentals for all librarians*. American Library Association.

Clinton, B. (1994). Coordinating Geographic Data Acquisition and access: The national spatial data infrastructure, Executive Order 12906 (13 April 1994), Edition of the Federal Register. *http://www. fgdc. gov/publications/documents/geninfo/execord. html*, *59*(71), 17671-17674.

Committee, F. G. D. (2000). Content Standard for Digital Geospatial Metadata Workbook (For use with FGDC-STD-001-1998), Version 2.0. *URL: http://www. fgdc. gov/publications/documents/metadata/workbook_0501_bmk. pdf*.

Criado, M., Crespo, M., Rodríguez, C., Bravo, M., & Ballari, D. (2007). Creación de Metadatos: Metodología y experiencia del Grupo de Catalogadores de la Información Geográfica. *Jornadas de las Infraestructuras de Datos Espaciales de España*.

Danko, D. M. (2005). ISO/TC211: Geographic information–metadata ISO 19115. En *World Spatial Metadata Standards* (pp. 535-555). Elsevier.

Dante, G. P. (2011). La gestión de información y sus modelos representativos. Valoraciones. *Ciencias de la Información*, *42*(2), 11-17.

Day, M., Guy, M., & Powell, A. (2004). Improving the quality of metadata in eprint archives. *Ariadne*, *38*.

Delgado, T., & Cruz, R. (2009). *Construyendo Infraestructuras de Datos Espaciales a nivel local. CUJAE, La Habana, 130pp*.

Duval, E., Hodgins, W., Sutton, S., & Weibel, S. L. (2002). Metadata principles and practicalities. *D-lib Magazine*, *8*(4), 1-10.

Gallego Priego, M. (2017). *Aplicación de los fundamentos de las infraestructuras de datos espaciales en la construcción de sistemas de información geográfica corporativos*.

Greenberg, J. (2004). Metadata extraction and harvesting: A comparison of two automatic metadata generation applications. *Journal of Internet Cataloging*, *6*(4), 59-82.

Hernando, R., & Macías, J. A. (2013). Uso de Realidad Aumentada Mediante Códigos QR para la Mejora en el Acceso y Disponibilidad de Recursos Educativos. *Actas del XV Simposio Internacional de Tecnologías de la Información y las Comunicaciones en la Educación (SINTICE 2013)*, 43-50.

Jokela, S. (2001). *Metadata enhanced content management in media companies*. Helsinki University of Technology.

Klopfer, M. (2005). Interoperability & Open Architectures: An analysis of existing standardisation processes & procedures. *OGC White Paper. Consortium, OG, Open Geospatial Consortium: 26p*.

Lizama, O., Kindley, G., Morales, J. J., & Gonzales, A. (2016). Redes de Computadores: Arquitectura Cliente-Servidor. *Universidad Tecnica Federico Santa Maria*, 1-8.

López, F. J. A., & Pascual, A. F. R. (2008). La familia ISO 19100. *Grupo de Investigación en Ingeniería Cartográfica Universidad de Jaén. Jaén Espanha*.

Masser, I. (2005). *GIS worlds: Creating spatial data infrastructures* (Vol. 338). Esri Press Redlands.

Morales Flores, E. (2004). *La gestión y los gestores de la información*. Bibliodocencia.

Morales, M. I. M., Agudelo, F. A. V., & Durán, D. E. S. (2012). Una síntesis conceptual de los servicios web para la gestión de información geográfica [Conceptual review about web services for geographic information management]. *Ventana Informática*, *27*.

Najjar, J., Ternier, S., & Duval, E. (2004). User behavior in learning object repositories: An empirical analysis. *EdMedia+ Innovate Learning*, 4373-4379.

Nebert, D. (2004). Geospatial Data Catalogue–Making Data Discoverable. *Developing spatial data infrastructures: The SDI cookbook (2.0. ed.): The Global Spatial Data Infrastructure Association*.

Nebert, D., Voges, U., & Bigagli, L. (2016). *OGC® Catalogue Services 3.0-General Model, Version 3.0.*

Oliva Santos, R., Garea Llano, E., & Maciá Pérez, F. (2009). *Propuesta preliminar de soporte para la integración de datos, metadatos y conocimiento geográfico mediante geo-ontologías*.

Pearsall, R. A., & Hogan, R. (2005). United States of America Content Standard for Digital Geospatial Metadata FGDC-STD-001-1998. En *World Spatial Metadata Standards* (pp. 469-489). Elsevier.

Ponjuán Dante, G. (2004). *Gestión de información: Dimensiones e implementación para el éxito organizacional*.

Rackham, L. (2015). *Metadata Guidelines for Geospatial Data Resources-Part 3*.

Rajabifard, A., Kalantari, M., & Binns, A. (2009). SDI and metadata entry and updating tools. *SDI convergence*, *121*.

Rajabifard, A., & Williamson, I. P. (2001). Spatial data infrastructures: Concept, SDI hierarchy and future directions. *Proceedings of GEOMATICS'80 Conference*, *10*.

Rodríguez Cruz, Y. (2008). Gestión de información e inteligencia: Integración en los contextos organizacionales. *Acimed*, *17*(5), 0-0.

Rojas Guerrero, M. N. (2014). Diseño metodológico para crear Infraestructuras de Datos Espaciales a escala Ciudad-Región en Colombia. *Escuela de Posgrados*.

Salas, K. R. (2002). Gestión de la Información en las Organizaciones. *Bibliotecas*, *20*(1), 19-34.

Santos, R. O., & Orozco, E. Q. (2006). Los metadatos geográficos: Actualidad y estándares. *Mapping*, *112*, 18-29.

Seiner, R. S. (2000). Questions metadata can answer. *Disponible en URL*.

Sutheebanjard, P., & Premchaiswadi, W. (2010). QR-code generator. *2010 Eighth International Conference on ICT and Knowledge Engineering*, 89-92.

Tomlinson, R. (2008). *Pensando en el SiG: Planification del sistema de informacion geografica dirigida a gerentes*. Esri Press.

Valencia, J. (2008). Pasado, presente y futuro de las Infraestructuras de Datos Espaciales. *España. Recuperado de http://ww2. pcypsitna. navarra. es/Aprende/Documents/PASADO-PRESENTE-YFUTURO-DE-LAS-INFRAESTRUCTURAS-DE-DATOS-ESPACIALES. pdf*.

Warnest, M. (2005). *A collaboration model for national spatial data infrastructure in federated countries*.

Wayne, L. (2005). *Institutionalize metadata before it institutionalizes you*. Federal Geographic Data Committee Reston, VA.

Wytzisk, A., & Sliwinski, A. (2004). Quo vadis SDI. *7th AGILE Conference on Geographic Information Science*, 43-49.