

**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**

*La Universidad Católica de Loja*

**FACULTAD DE INGENIERÍAS Y ARQUITECTURA**

**CARRERA DE COMPUTACIÓN**

**REPRESENTACIÓN AVANZADA DEL CONOCIMIENTO Y RAZÓN (ONTOLOGÍA DATASET VIDEOJUEGOS)**

**Integrantes:** Kevin Bustamante, Freddy León, Miguel Álvarez y Byron Castillo.

**Docente:** Irma Elizabeth Cadme Samaniego.

LOJA - 2025

**Propósito:**

Desarrollar una ontología semántica basada en el dataset RAWG Game Dataset, permitiendo consultas inteligentes sobre videojuegos, integración con otras fuentes de datos y razonamiento automático.

**Lenguaje Ontológico:**

La ontología será desarrollada en OWL 2 DL y serializada en RDF/XML o Turtle. El namespace sugerido es: http://example.org/rawg#.

# **Resumen**

El presente proyecto tiene como objetivo principal el desarrollo de una ontología híbrida para organizar y representar datos de videojuegos extraídos del dataset denominado RAWG Game Dataset.La ontología, diseñada en el lenguaje OWL 2 DL y serializada en formatos como RDF/XML o Turtle, busca facilitar consultas inteligentes, integrar vocabularios establecidos como Schema.org (específicamente la clase VideoGame), Dublin Core para metadatos esenciales, FOAF para describir organizaciones involucradas y propiedades personalizadas adaptadas a las particularidades del dataset ya mencionado, permitiendo proporcionar una representación semántica robusta que facilite la interoperabilidad entre sistemas, la ejecución de consultas avanzadas y el profundo análisis de datos relacionados con videojuegos.

La ontología se desarrolló siguiendo un enfoque metodológico estructurado, se implementó en el lenguaje OWL (Web Ontology Language) y se instanció utilizando datos reales del dataset seleccionado. Posteriormente, se validó la consistencia del mapeo y la capacidad de la ontología para responder a preguntas del dominio. Este esfuerzo no solo organiza la información de manera coherente, sino que también abre la puerta a aplicaciones prácticas como motores de búsqueda semánticos especializados, sistemas de recomendación personalizados basados en preferencias de usuarios y análisis detallados de tendencias en la industria del entretenimiento interactivo. La combinación de estándares existentes con elementos personalizados asegura que la ontología sea flexible y específica, adaptándose a los datos de RAWG mientras mantiene compatibilidad con ecosistemas semánticos más amplios.

Uno de los principales usos de esta ontología es la realización de consultas semánticas avanzadas, que permitirán responder preguntas complejas como: ¿Qué juegos de rol (RPG) lanzados desde 2022 tienen una calificación igual o superior a 4 o 0? ¿Qué desarrolladores han publicado más de cinco juegos del género ‘Action’? o ¿Qué plataformas albergan juegos desarrollados por Ubisoft? Además, la ontología facilitará la recomendación inteligente de videojuegos, el enriquecimiento de datos mediante la integración con fuentes externas, el razonamiento lógico para validar consistencia y la visualización de relaciones entre diferentes entidades, como juegos, géneros y desarrolladores.

Para garantizar su utilidad y eficacia, la ontología debe cumplir con una serie de requisitos funcionales y no funcionales. Entre los funcionales, destacan las denominadas preguntas de competencia, que sirven como casos de uso clave para validar el modelo. Estas incluyen consultas sobre combinaciones de géneros (como Adventure y Puzzle), estudios desarrolladores con mayor producción de juegos conocidos y cruces de información entre plataformas y desarrolladores. Por otro lado, los requisitos no funcionales exigen que la ontología sea lógicamente consistente, modular y escalable, permitiendo futuras ampliaciones sin comprometer su estructura. Así mismo, al estar bien documentada, es compatible con herramientas como Protégé que mantienen la interoperabilidad con vocabularios externos para facilitar la vinculación con otros conjuntos de datos semánticos.

# **Contexto del Dominio Utilizado**

El dominio de esta ontología se centra en los videojuegos, abarcando información detallada de títulos individuales presentes en el dataset de RAWG. Este incluye metadatos como identificadores únicos (id), nombres (name), fechas de lanzamiento (released), calificaciones (rating), plataformas soportadas (platforms), géneros (genres), desarrolladores (developers), publicadores (publishers), así como atributos específicos como tiempo de juego promedio (playtime), conteo de logros (achievements\_count) y estados de interacción de los usuarios (“yet”, "owned", "beaten", "toplay", “dropped, ”playing”).

La industria de los videojuegos ha experimentado un crecimiento exponencial, superando a otras formas de entretenimiento en ingresos y participación de usuarios, este auge genera una enorme cantidad de datos que, sin una estructura adecuada, pueden ser difíciles de gestionar y aprovechar. La ontología propuesta aborda estos desafíos al ofrecer un marco semántico que organiza los datos de manera lógica y accesible, permitiendo aplicaciones como la recomendación de juegos basada en géneros o plataformas preferidas, el análisis de popularidad a través de ratings y reseñas, y la mejora de la experiencia en plataformas digitales mediante una búsqueda más precisa y contextualizada.

# **Diseño y Construcción de la Ontología**

* 1. **Metodología de Desarrollo**

La ontología se desarrolló utilizando una versión adaptada de METHONTOLOGY, una metodología ampliamente aceptada en la ingeniería ontológica por su enfoque sistemático y bien definido. Se prefirió METHONTOLOGY sobre alternativas como NeOn debido a su estructura clara, que incluye etapas como especificación, conceptualización, implementación y evaluación; este enfoque garantizó que el desarrollo fuera ordenado y alineado con los objetivos del proyecto.

Se comenzó con la búsqueda del dataset con el que trabajaríamos, definiendo los objetivos, propósito, alcance, lenguaje ontológico con la que será desarrollada, potenciales usuarios, usos que tendrá nuestra ontología, requisitos funcionales, requisitos no funcionales, diccionario de clases, matriz de atributos y una tabla de relaciones binarias, para poder comprender con qué datos estamos trabajando y así tener in contexto de toda la información que contiene este dataset.

Las entidades principales y las que más se destaco en el desarrollo fueron:

1. **Especificación:** Se estableció el propósito de la ontología: representar datos de videojuegos de RAWG para consultas y análisis semántico, además se identificaron usuarios potenciales:

* Analista de Datos
* Desarrolladores de sistemas de recomendación
* Investigadores en IA
* Integradores de plataformas de videojuegos
* Entusiastas con conocimientos previos

1. **Conceptualización:** Se analizó meticulosamente el dataset para la extracción de conceptos clave como (videojuegos, plataformas) y algunas relaciones como (un juego tiene un desarrollador), haciendo el reúso de vocabularios como Schema.org y Dublin Core, complementándolos como propiedades personalizadas.
2. **Implementación:** La ontología se codificó en OWL haciendo uso de la plataforma de Protegé, definiendo clases, propiedades y restricciones para reflejar el dominio.
3. **Evaluación:** Se verificó la consistencia con un razonador y se probaron consultas de competencias para validar su funcionalidad.
   1. **Dominio de Conocimiento**

El dominio de conocimiento abarca todos los aspectos de los videojuegos según el dataset de RAWG. Incluye identificadores únicos, nombres, fechas de lanzamiento y modificación, ratings (usuarios y agregados como Metacritic), plataformas, géneros, desarrolladores, publicadores, tiempo de juego promedio, conteo de logros, juegos en serie, sugerencias de juegos similares, clasificaciones de contenido y estados de interacción de usuarios. No cubre jugadores ni detalles técnicos de hardware más allá de las plataformas.

* 1. **Requisitos Ontológicos**
* Alcance

Modela los videojuegos del dataset de RAWG, sus propiedades y relaciones, habilitando consultas semánticas e interoperabilidad con sistemas externos. Excluye información sobre jugadores o especificaciones de hardware.

* Requisitos Funcionales (Preguntas de Competencia)

1. ¿Qué juegos del género RPG lanzados desde 2022 tienen un rating mayor o igual a 4,0?
2. ¿Qué desarrolladores han creado más de 5 juegos del género ‘Action’?
3. ¿Qué plataformas comparten juegos desarrollados por Ubisoft?
4. ¿Qué juegos combinan los géneros ‘Adventure’ y ‘Puzzle’?
5. ¿Qué estudio desarrollador tiene más juegos en el dataset?
6. ¿Cuáles son los juegos con una calificación de Metacritic superior a 90?
7. Obtener los juegos lanzados en un año específico (ej., 2023):
8. ¿Qué juegos fueron actualizados por última vez después de una fecha específica (ej., 2024-01-01)?
9. Encontrar los 10 juegos mejor valorados por los usuarios de RAWG:
10. ¿Cuántos logros tiene el juego "The Witcher 3: Wild Hunt"?

* Requisitos no Funcionales
* Consistencia lógica
* Modularidad y escalabilidad
* Documentación adecuada
* Compatibilidad con Protegé
* Interoperabilidad con Schema.org, Dubling Core,etc
  1. **Descripción de la Ontología**
* Clases y Jerarquías

La ontología usa schema:VideoGame como clase principal, relacionada con schema:VideoGamePlatform, schema:Organization, schema:Genre, y schema:GameRating. vgo:VideoGame extiende propiedades específicas de gaming, y rawg:UserGameStatus añade estados de usuario. La jerarquía es plana, con relaciones definidas por propiedades.

|  |  |  |
| --- | --- | --- |
| CLASE | URI | DESCRIPCIÓN |
| schema:VideoGame | <https://schema.org/VideoGame> | Clase principal para videojuegos |
| schema:VideoGamePlatform | <https://schema.org/VideoGamePlatform> | Plataformas de videojuegos |
| schema:Organization | <https://schema.org/Organization> | Desarrolladores y publicadores |
| schema: Genre | <https://schema.org/Genre> | Géneros de videojuegos |
| schema:GameRating | <https://schema.org/GameRating> | Clasificaciones de contenido |
| vgo:VideoGame | <http://purl.org/net/VideoGameOntology#VideoGame> | Videojuegos con propiedades de gaming |
| rawg:UserGameStatus | <http://rawg.videogames.ontology#UserGameStatus> | Estados de interacción de usuarios |

* Propiedades

|  |  |  |  |
| --- | --- | --- | --- |
| Propiedad de Datos | Dominio | Rango | Descripción |
| dcterms:identifier | schema:VideoGame | xsd:string | Identificador único. |
| schema:alternateName | schema:VideoGame | xsd:string | Nombre alternativo URL-friendly |
| schema:name | schema:VideoGame | xsd:string | Nombre del videojuego. |
| schema:aggregateRating | schema:VideoGame | schema:AggregateRating | Rating agregado (e.g., Metacritic). |
| schema:datePublished | schema:VideoGame | xsd:date | Fecha de publicación. |
| rawg:toBeAnnounced | schema:VideoGame | xsd:boolean | Fecha por anunciarse. |
| dcterms:modified | schema:VideoGame | xsd:dateTime | Fecha de modificación. |
| schema:url | schema:VideoGame | xsd:anyURI | URL oficial. |
| schema:ratingValue | schema:VideoGame | xsd:decimal | Valor del rating. |
| schema:bestRating | schema:VideoGame | xsd:decimal | Mejor rating posible. |
| vgo:averagePlayTime | schema:VideoGame | xsd:integer | Tiempo promedio de juego. |
| rawg:achievementCount | schema:VideoGame | xsd:integer | Número de logros. |
| schema:ratingCount | schema:VideoGame | xsd:integer | Número de ratings. |
| rawg:suggestionCount | schema:VideoGame | xsd:integer | Sugerencias de juegos similares. |
| rawg:gameSeriesCount | schema:VideoGame | xsd:integer | Juegos en la serie. |
| schema:reviewCount | schema:VideoGame | xsd:integer | Número de reseñas. |
| schema:gamePlatform | schema:VideoGame | schema:VideoGamePlatform | Plataforma del juego. |
| schema:developer | schema:VideoGame | schema:Organization | Desarrollador. |
| schema:genre | schema:VideoGame | schema:Genre | Género |
| schema:publisher | schema:VideoGame | schema:Organization | Publicador |
| schema:contentRating | schema:VideoGame | schema:GameRating | Clasificación de contenido. |
| rawg:addedStatusYet | schema:VideoGame | xsd:integer | Usuarios sin estado específico. |
| rawg:addedStatusOwned | schema:VideoGame | xsd:integer | Usuarios que poseen el juego. |
| rawg:addedStatusBeaten | schema:VideoGame | xsd:integer | Usuarios que terminaron el juego. |
| rawg:addedStatusToPlay | schema:VideoGame | xsd:integer | Usuarios que planean jugar. |
| rawg:addedStatusDropped | schema:VideoGame | xsd:integer | Usuarios que abandonaron el juego. |
| rawg:addedStatusPlaying | schema:VideoGame | xsd:integer | Usuarios jugando actualmente. |

* Restricciones y Axiomas
* Cardinalidad: dcterms:identifier (exactamente 1); schema:developer, schema:publisher, schema:gamePlatform (múltiple).
  1. **Implementación de la ontología**

Para la implementación de la ontología se seleccionó Protégé como herramienta central, aprovechando su capacidad como editor visual de ontologías OWL desarrollado por la Universidad de Stanford. Este software open-source (versión 5.5.0) permite modelar de manera intuitiva todos los componentes ontológicos mediante una interfaz gráfica organizada en pestañas especializadas: Clases para definir la jerarquía taxonómica, Object Properties para relaciones entre individuos, Data Properties para atributos literales y Axioms para restricciones lógicas (cardinalidades). Protégé destaca por integrar reasoners como HermiT, que verifica en tiempo real la consistencia del modelo al detectar contradicciones (ej. conflictos en herencias múltiples), y ofrece funcionalidades avanzadas como la generación automática de documentación en HTML y serialización flexible a formatos como Turtle (.ttl) o RDF/XML, garantizando compatibilidad con estándares semánticos. Su elección se fundamenta en su robustez para proyectos académicos/industriales, soporte para importación de vocabularios externos (FOAF, Schema.org), y capacidad colaborativa mediante control de versiones integrado, lo que lo posiciona como el entorno óptimo para construir ontologías OWL 2 DL sin requerir programación manual de triples.

# **Instanciación de la Ontología, proceso de conversión**

* 1. **Selección del Dataset**

El dataset seleccionado para instanciar la ontología se denomina “rawg\_games.csv”, que contiene datos detallados sobre videojuegos extraídos de la plataforma RAWG, fue elegido por su riqueza en información relevante al dominio de los videojuegos y por su formato estructurado, lo que facilita el proceso de mapeo y conversión a la ontología. A continuación, se describen sus características clave:

* **Origen**: El dataset proviene de RAWG, una conocida base de datos de videojuegos que recopila información exhaustiva sobre títulos, plataformas, desarrolladores, y más.
* **Formato**: El formato se lo encuentra explícitamente en formato CSV lo que permite una manipulación sencilla.
* **Columnas** **principales**: Incluye columnas como id, slug, name, metacritic, released, tba, updated, website, rating, rating\_top, playtime, achievements\_count, ratings\_count, suggestions\_count, game\_series\_count, reviews\_count, platforms, developers, genres, publishers, esrb\_rating, y varios campos de added\_status que indican la interacción de los usuarios con los juegos.
* **Características Importantes**: El dataset es completo y cubre tanto metadatos básicos (nombre, fecha de lanzamiento) como datos específicos del dominio (tiempo de juego promedio, número de logros). Además, incluye información sobre las relaciones entre videojuegos y otras entidades, como plataformas, géneros, y organizaciones (desarrolladores y publicadores), lo que lo hace ideal para instanciar una ontología rica en relaciones semánticas.
  1. **Proceso de Mapeo e Instanciación**

Para nuestro proceso de mapeo e instanciación de los datos del csv a los individuos de la ontología se llevó a cabo mediante el uso de un script en Python, construido específicamente para transformar la información contenida en un archivo TTL compatible con la ontología definida. Este script emplea bibliotecas como pandas para la lectura y manipulación de datos, urllib.parse para el manejo de URIs, y expresiones regulares (re) para la limpieza de cadenas. El objetivo principal fue tomar los datos del dataset de videojuegos de RAWG, estructurados en un CSV, y mapearlos a las clases y propiedades definidas en la ontología, generando instancias individuales para plataformas, desarrolladores, editores, géneros, ratings ESRB y los propios videojuegos.

El proceso comienza con la lectura del archivo CSV mediante pandas, lo que permite cargar el dataset en un DataFrame para su procesamiento. Dado que el dataset contiene campos con múltiples valores separados por "||", como plataformas o géneros, se implementó una función específica, parse\_delimited\_field, para analizar estos campos y extraer los valores individuales en una lista de diccionarios. Por ejemplo, un campo como "PC||PlayStation 4" se convierte en [{'name': 'PC'}, {'name': 'PlayStation 4'}]. Esta función también maneja casos de valores nulos o vacíos, asegurando robustez en el procesamiento.

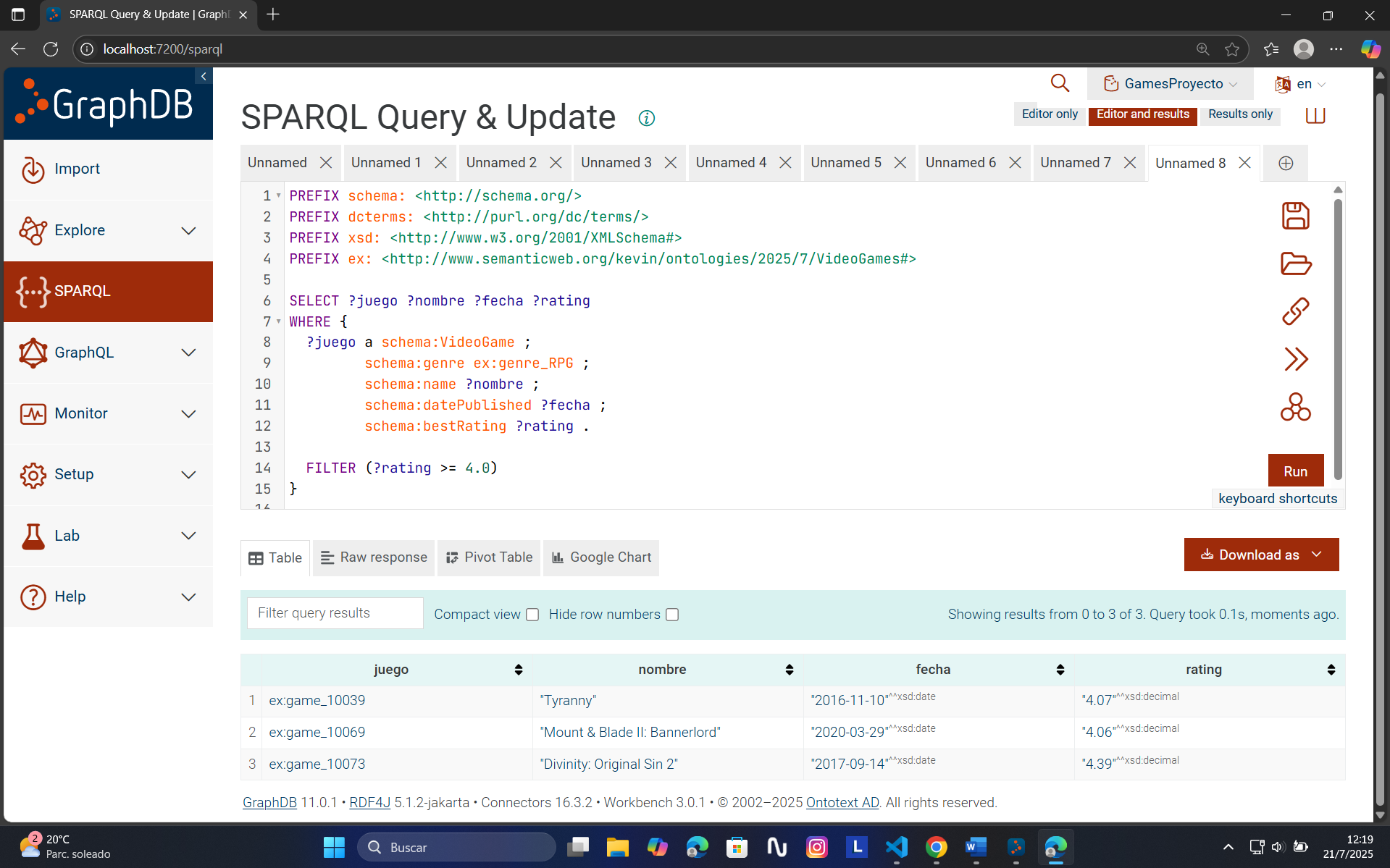
Antes de generar las instancias de los videojuegos, el script realiza un primer pase sobre los datos para recopilar entidades únicas (plataformas, desarrolladores, editores, géneros y ratings ESRB) y almacenarlas en conjuntos, evitando duplicados. Para cada entidad, se limpian los nombres mediante funciones como clean\_uri\_string, que elimina caracteres no deseados y reemplaza espacios por guiones bajos, y clean\_literal\_string, que escapa caracteres especiales para su uso en literales TTL. Estas entidades se escriben en el archivo TTL como individuos de clases específicas, por ejemplo, :platform\_PC rdf:type schema:VideoGamePlatform ; schema:name "PC" ..

Posteriormente, el script procesa cada fila del CSV para generar las instancias de los videojuegos. Cada juego se identifica con un URI único basado en su ID (:game\_10092 para "Minion Masters") y se asocia con propiedades como schema:name, schema:datePublished, y métricas numéricas como schema:ratingCount o rawg:achievementCount. Las relaciones con entidades auxiliares, como las plataformas (schema:gamePlatform) o desarrolladores (schema:developer), se establecen utilizando los URIs generados previamente. El resultado es un archivo TTL estructurado que representa fielmente los datos del CSV en el marco de la ontología.

* 1. **Herramientas para conversión**

Aunque el mapeo principal se realizó con el script en Python descrito, se mencionan herramientas como Ontotext Refine y GraphDB en el contexto del proyecto. Ontotext Refine es una herramienta poderosa para la limpieza y transformación de datos, que permite a los usuarios importar datasets en diversos formatos (como CSV), aplicar transformaciones interactivas y mapearlos a estructuras RDF basadas en ontologías. En este caso, aunque el mapeo se llevó a cabo programáticamente con Python, Ontotext Refine podría haber sido utilizada en una etapa previa para preparar el dataset, por ejemplo, corrigiendo inconsistencias en los datos (valores nulos, formatos incorrectos) o estandarizando los nombres de las entidades antes de pasarlos al script. Su capacidad para generar RDF directamente también sugiere que podría servir como alternativa al script en otros escenarios, especialmente para usuarios menos familiarizados con la programación.

Por otro lado, GraphDB desempeñó un papel clave en la fase de almacenamiento y visualización. GraphDB es un almacén de datos RDF que soporta la carga de archivos TTL, permitiendo almacenar, visualizar y consultar datos semánticos mediante SPARQL. Una vez generado el archivo TTL con el script en Python, este se subió a GraphDB, donde los datos se organizaron como un grafo semántico. Esto facilitó la visualización de las relaciones entre videojuegos, plataformas, desarrolladores y otros individuos, así como la ejecución de consultas SPARQL para explorar el dataset. Por ejemplo, se podrían listar todos los juegos de un género específico o verificar las plataformas asociadas a un juego, aprovechando la estructura del grafo para análisis avanzados.

****

<https://github.com/ByronCast09/Ontolog-aRepr.Conocimiento.git>

* 1. **Validación de la Instanciación**

La validación de la instanciación se realizó mediante una combinación de métodos manuales y automatizados para garantizar que los datos se mapearan correctamente a la ontología. En primer lugar, se llevó a cabo una verificación manual de un subconjunto de los datos instanciados en el archivo TTL. Esto implicó revisar visualmente las tripletas generadas para juegos específicos, como :game\_10092 ("Minion Masters"), asegurándose de que propiedades como schema:name, schema:datePublished y relaciones como schema:gamePlatform coincidieran con los valores originales del CSV. Esta inspección también comprobó que los literales estuvieran correctamente formateados y que las relaciones apuntaran a los URIs esperados.

Además, se utilizó GraphDB para ejecutar consultas SPARQL como parte del proceso de validación. Por ejemplo, una consulta como SELECT ?game ?platform WHERE { ?game schema:gamePlatform ?platform } permitió verificar que las plataformas estuvieran correctamente asociadas a los juegos, mientras que SELECT ?game ?name WHERE { ?game rdf:type schema:VideoGame ; schema:name ?name } confirmó la presencia de todos los juegos con sus nombres correspondientes. Estas consultas ayudaron a detectar posibles omisiones o errores en el mapeo, como relaciones faltantes o valores mal interpretados.

El script en Python también incluyó una función de diagnóstico, diagnose\_csv\_format, que se ejecutó antes de la generación del TTL para examinar el formato de los campos en el CSV. Esta herramienta mostró muestras de datos crudos y sus resultados parseados, identificando problemas potenciales como separadores inconsistentes o valores malformados. Aunque no se especifican pruebas unitarias explícitas, el diseño modular del script (con funciones separadas para limpieza, parseo y generación) sugiere que se podrían haber implementado pruebas para verificar cada componente. En conjunto, estas estrategias aseguraron que la instanciación fuera precisa y consistente con la ontología definida, permitiendo un uso confiable del grafo en GraphDB.

# **Consultas SPARQL y Verificación del Modelo**

¿Qué juegos del género RPG lanzados desde 2022 tienen un rating mayor o igual a 4,0?

No hay juegos desde el 2022 que cumplan un rating mayor a 4, por eso se removió la fecha

PREFIX schema: <http://schema.org/>

PREFIX dcterms: <http://purl.org/dc/terms/>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ex: <http://www.semanticweb.org/kevin/ontologies/2025/7/VideoGames#>

SELECT ?juego ?nombre ?fecha ?rating

WHERE {

?juego a schema:VideoGame ;

schema:genre ex:genre\_RPG ;

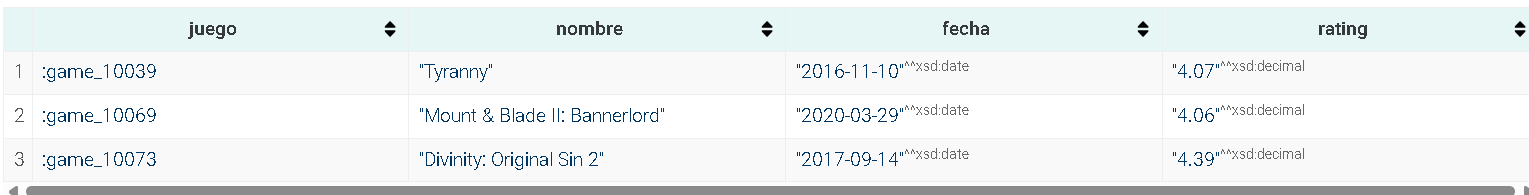
schema:name ?nombre ;

schema:datePublished ?fecha ;

schema:bestRating ?rating .

FILTER (?rating >= 4.0)

}



**Estamos viendo que solo 3 juegos de los que existen en el grafo tienen una nota mayor o igual a 4, sin embargo, se tubo que sacar la fecha ya que no existía en el año que e deseaba ver la consulta algún juego con la calificación que sale aquí con el nombre además usamos la propiedad FILTER para que se cumpla esta calificación.**

¿Qué desarrolladores han creado más de 5 juegos del género ‘Action’?

No hay desarrolladores que han hecho mas de 5, así que pusimos 2  
PREFIX schema: <http://schema.org/>

PREFIX ex: <http://www.semanticweb.org/kevin/ontologies/2025/7/VideoGames#>

SELECT ?desarrollador (COUNT(?juego) AS ?cantidadJuegos)

WHERE {

?juego a schema:VideoGame ;

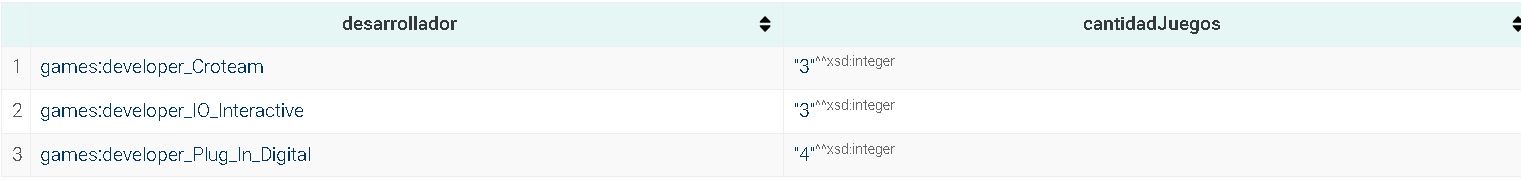
schema:genre ex:genre\_Action ;

schema:developer ?desarrollador .

}

GROUP BY ?desarrollador

HAVING (COUNT(?juego) > 2)



**La consulta sirve bien ya que nos trae las entidades que desarrollaron el videojuego y la cantidad de videojuegos que han realizado en este caso nos salen 3 porque filtramos a los que hayan realizado mas de 2**

¿Qué plataformas comparten juegos desarrollados por Ubisoft?

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX schema: <http://schema.org/>

PREFIX dcterms: <http://purl.org/dc/terms/>

PREFIX vgo: <http://purl.org/net/VideoGameOntology#>

PREFIX rawg: <http://rawg.io/ontology#>

PREFIX games: <http://www.semanticweb.org/kevin/ontologies/2025/7/VideoGames#>

SELECT DISTINCT ?plataforma ?nombrePlataforma

WHERE {

?game rdf:type schema:VideoGame ;

schema:developer ?desarrollador ;

schema:gamePlatform ?plataforma .

# Filtrar por Ubisoft (buscar en URI o nombre)

FILTER(CONTAINS(LCASE(STR(?desarrollador)), "ubisoft"))

# Obtener nombre de la plataforma

OPTIONAL { ?plataforma schema:name ?nombrePlataforma }

BIND(COALESCE(?nombrePlataforma, STRAFTER(STR(?plataforma), "#platform\_")) AS ?finalnombrePlataforma)

}

ORDER BY ?finalnombrePlataforma

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**El código está diseñado para encontrar todas las plataformas de juego en las que Ubisoft (o una entidad cuyo identificador contenga "Ubisoft") ha lanzado videojuegos, y luego presenta los nombres de esas plataformas de manera legible, ordenadas alfabéticamente.**

¿Qué juegos combinan los géneros ‘Adventure’ y ‘Puzzle’?

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX schema: <http://schema.org/>

PREFIX dcterms: <http://purl.org/dc/terms/>

PREFIX vgo: <http://purl.org/net/VideoGameOntology#>

PREFIX rawg: <http://rawg.io/ontology#>

PREFIX games: <http://www.semanticweb.org/kevin/ontologies/2025/7/VideoGames#>SELECT ?game ?name ?datePublished

WHERE {

?game rdf:type schema:VideoGame ;

schema:name ?name ;

schema:genre games:genre\_Adventure ;

schema:genre games:genre\_Puzzle .

OPTIONAL { ?game schema:datePublished ?datePublished }

}

ORDER BY ?name

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

**La consulta cumple con lo queremos si vemos usamos propiedade como schema:genre (Adventure, Puzzle) y nos muestra todos los juegos que tengan a esos dos géneros en su categoría además de mostrar el id del videojuego, el nombre y la fecha donde se publico.**

¿Qué estudio desarrollador tiene más juegos en el dataset?

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX schema: <http://schema.org/>

PREFIX dcterms: <http://purl.org/dc/terms/>

PREFIX vgo: <http://purl.org/net/VideoGameOntology#>

PREFIX rawg: <http://rawg.io/ontology#>

PREFIX games: <http://www.semanticweb.org/kevin/ontologies/2025/7/VideoGames#>

SELECT ?developer ?developerName (COUNT(?game) AS ?totalGames)

WHERE {

?game rdf:type schema:VideoGame ;

schema:developer ?developer .

# Obtener el nombre del desarrollador

OPTIONAL { ?developer schema:name ?developerName }

OPTIONAL { ?developer dcterms:title ?developerName2 }

BIND(COALESCE(?developerName, ?developerName2, STRAFTER(STR(?developer), "#developer\_")) AS ?finalName)

}

GROUP BY ?developer ?developerName

ORDER BY DESC(?totalGames)

LIMIT 10

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Aquí vuelta estamos trayendo todos los desarrolladore con más juegos desarrollados y lo ordenamos tenemos varios OPTIONAL para que igual se presente aun así alguna de las columnas este en nulo podemos ver que el más desarrollado aquí es Plug in Digital, pero estamos limitando a 10 la vista para que se muestren los 10 primeros.**

Algunas **Limitaciones** que estamos teniendo son los datos porque tenemos preguntas como:

* ¿Qué juegos tienen más de 10000 usuarios que los tienen como "Owned"?

El problema es que la consulta se ejecuta, pero no había ningún juego con mas de 10000 usuario como OWNED

PREFIX schema: <http://schema.org/>

PREFIX rawg: <http://rawg.io/ontology#> # O el prefijo de tu ontología donde definas esta propiedad

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?gameName ?ownedUsers

WHERE {

?game a schema:VideoGame ;

schema:name ?gameName ;

rawg:addedStatusOwned ?ownedUsers . # Asumo esta propiedad para los usuarios "Owned"

FILTER (?ownedUsers > 10000)

}

ORDER BY DESC(?ownedUsers)

**De ahí todas las consultas se ejecutaron bien el modelo es robusto y eficiente para lo que se buscó hacer.**

# **Conclusiones**

El desarrollo de la ontología semántica basada en el dataset RAWG Game Dataset

cumplió exitosamente con los objetivos establecidos. Se logró representar de manera

estructurada y coherente los datos de videojuegos, permitiendo consultas semánticas

avanzadas mediante SPARQL que respondieron a preguntas de competencia clave, como

identificar juegos RPG con calificaciones altas, desarrolladores con múltiples títulos de

acción o plataformas asociadas a Ubisoft. La ontología, implementada en OWL 2 DL

y serializada en Turtle, demostró ser consistente y funcional, validada a través de her-

ramientas como Protégé y GraphDB. Los hallazgos destacan la capacidad de la ontología

para organizar datos complejos, facilitar la interoperabilidad con vocabularios estándares

como Schema.org y Dublin Core, y soportar aplicaciones prácticas como sistemas de re-

comendación y análisis de tendencias en la industria de videojuegos.

Entre las lecciones aprendidas, destaca la importancia de una limpieza rigurosa de datos

durante el mapeo, ya que el dataset RAWG presentaba inconsistencias como valores nu-

los o formatos múltiples (e.g., valores separados por "||"). El uso de scripts en Python,

combinado con herramientas como Ontotext Refine, permitió superar estos desafíos, pero

requirió un diseño modular para manejar errores de manera eficiente. La validación man-

ual y automatizada (con SPARQL y reasoners como HermiT) fue crucial para garantizar

la calidad de la instanciación, enseñando la necesidad de pruebas iterativas para detectar

omisiones o relaciones incorrectas. Además, la elección de METHONTOLOGY como

metodología resultó efectiva, pero su aplicación demandó un equilibrio entre la especifici-

dad del dominio y la reusabilidad de vocabularios externos, lo que reforzó la importancia

de una planificación detallada en la fase de conceptualización.

Para futuras mejoras, se propone integrar datos adicionales de otras fuentes, como Steam

o Metacritic, para enriquecer la ontología con información sobre jugadores, reseñas de-

talladas o especificaciones técnicas de hardware, ampliando así su alcance. La incorpo-

ración de nuevas propiedades, como mecánicas de juego específicas o compatibilidad con

tecnologías emergentes (e.g., VR), podría hacer la ontología más relevante para desar-

rolladores de sistemas de recomendación avanzados.

# **Referencias**

[1] Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Proceedings of the AAAI-97 Spring Symposium Series on Ontological Engineering, 33–40.

[2] W3C OWL Working Group. (2012). OWL 2 Web Ontology Language Document Overview (Second Edition). Retrieved from <https://www.w3.org/TR/owl2-overview/>

[3] Guha, R. V., Brickley, D., & Macbeth, S. (2016). Schema.org: Evolution of Structured Data on the Web. Communications of the ACM, 59(2), 44–51. <https://schema.org>

[4] Dublin Core Metadata Initiative. (2020). Dublin Core Metadata Element Set, Version 1.1. Retrieved from <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/>

[5] Musen, M. A. (2015). The Protégé Project: A Look Back and a Look Forward. AmMatters, 1(4), 4–12. <https://protege.stanford.edu>

[6] Ontotext. (2023). GraphDB: RDF Triplestore for Knowledge Graphs. Retrieved from <https://www.ontotext.com/products/graphdb/>

[7] RAWG. (2023). RAWG Video Games Database. Retrieved from <https://rawg.io>

[8] W3C SPARQL Working Group. (2013). SPARQL 1.1 Query Language. Retrieved from <https://www.w3.org/TR/sparql11-query/>