



MANUAL TÉCNICO

SmartCompareMarket

Marketplace Semántico con Comparación Inteligente

Proyecto 16 - Nivel 2

Proyecto Final - Web Semántica y Ontologías

Versión 1.0

Universidad La Salle
Facultad de Ingeniería
Carrera Profesional de Ingeniería de Software

Docente	Ing. Marco Antonio Camacho Alatrista
Estudiante	Correo Electrónico
Álvaro André Machaca Meléndez	amachacam@ulasalle.edu.pe

Diciembre 2025
Ciclo Académico: 2025 - II

Índice

1. Descripción General del Sistema	3
1.1. Propósito	3
1.2. Alcance Técnico	3
2. Arquitectura del Sistema	4
2.1. Diagrama de Arquitectura	4
2.2. Flujo de Datos	4
3. Tecnologías Utilizadas	5
3.1. Stack del Backend	5
3.2. Stack del Frontend	6
3.3. Tecnologías Semánticas	8
4. Estructura del Proyecto	9
4.1. Estructura de Directorios	9
5. Ontología OWL 2	11
5.1. Características Generales	11
5.2. Jerarquía de Clases	11
5.3. Propiedades de Datos (Data Properties)	13
6. Reglas SWRL	15
6.1. Ubicación en la Ontología	15
6.2. Reglas Implementadas	15
6.2.1. Regla 1: DetectarGamer	15
6.2.2. Regla 2: EncontrarMejorPrecio	15
6.2.3. Regla 3: ClasificarPositivas	16
6.2.4. Regla 4: ClasificarNegativas	16
6.3. Ejecución de Reglas	16
7. API REST - Endpoints	17
7.1. Base URL	17
7.2. Endpoints de Productos	17
7.2.1. GET /products	17
7.2.2. GET /products/{product_id}/relationships	17
7.3. Endpoint de Comparación	19
7.3.1. POST /compare	19
7.4. Endpoint de Recomendaciones	20
7.4.1. POST /recommendations	20
7.5. Endpoint de Búsqueda SPARQL	22
7.5.1. GET /search	22
7.6. Endpoints SWRL	23
7.6.1. GET /swrl/gaming-laptops	23
7.7. Endpoint de Validación	25
7.7.1. GET /validate/all	25
7.8. Documentación Interactiva	26

8. Servicios del Backend	28
8.1. ComparisonService	28
8.1.1. Algoritmo de Scoring	28
8.1.2. Cálculo del Score	28
8.2. RecommendationService	30
8.2.1. Sistema de Scoring para Recomendaciones	30
8.3. ValidationService	30
9. Instalación para Desarrolladores	31
9.1. Prerrequisitos	31
9.2. Clonar el Repositorio	31
9.3. Configuración del Backend	31
9.4. Configuración del Frontend	32
10. Ejecución y Despliegue	33
10.1. Modo Desarrollo	33
10.2. Modo Producción	34
10.3. Puertos Utilizados	34
11. Testing	35
11.1. Tests del Backend	35
11.2. Tests de la API	35
11.3. Tests Manuales Recomendados	35
12. Mantenimiento y Extensibilidad	37
12.1. Agregar Nuevos Productos	37
12.2. Agregar Nuevas Reglas SWRL	38
12.3. Modificar Pesos de Comparación	38
13. Resumen de Archivos Importantes	39
14. Información del Proyecto	40
14.1. Autor	40
14.2. Repositorio	40
14.3. Historial de Versiones	40

1. Descripción General del Sistema

1.1. Propósito

SmartCompareMarket es un sistema de marketplace que implementa tecnologías de **Web Semántica** para proporcionar:

- Comparación inteligente de productos electrónicos
- Clasificación automática mediante razonamiento OWL
- Recomendaciones personalizadas basadas en ontologías
- Búsquedas semánticas con SPARQL
- Validación de consistencia de datos

1.2. Alcance Técnico

El sistema implementa los siguientes requisitos funcionales del **Proyecto 16 - Nivel 2**:

Req.	Descripción	Estado
RF1	Ontología de productos con jerarquías complejas	[OK] 100 %
RF2	Modelado de equivalencias semánticas	[OK] 100 %
RF3	Reglas de inferencia para compatibilidades	[OK] 100 %
RF4	Motor de comparación con razonamiento	[OK] 100 %
RF5	Búsqueda con filtros semánticos (SPARQL)	[OK] 100 %
RF6	Recomendaciones basadas en perfil de usuario	[OK] 100 %
RF7	Consultas SPARQL para análisis de mercado	[OK] 100 %
RF8	Clasificación automática (subsunción OWL)	[OK] 100 %
RF9	Validación de consistencia de especificaciones	[OK] 100 %

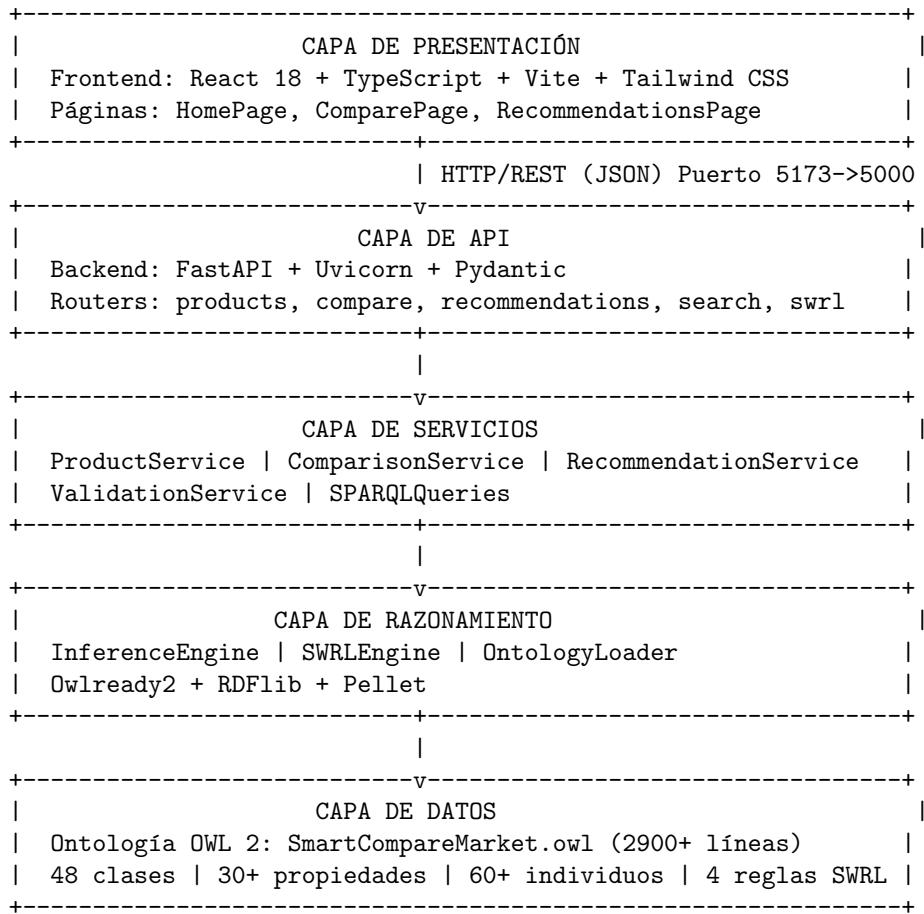
Cuadro 1: Cumplimiento de requisitos funcionales

Total: 9 de 9 requisitos completamente funcionales (100 %)

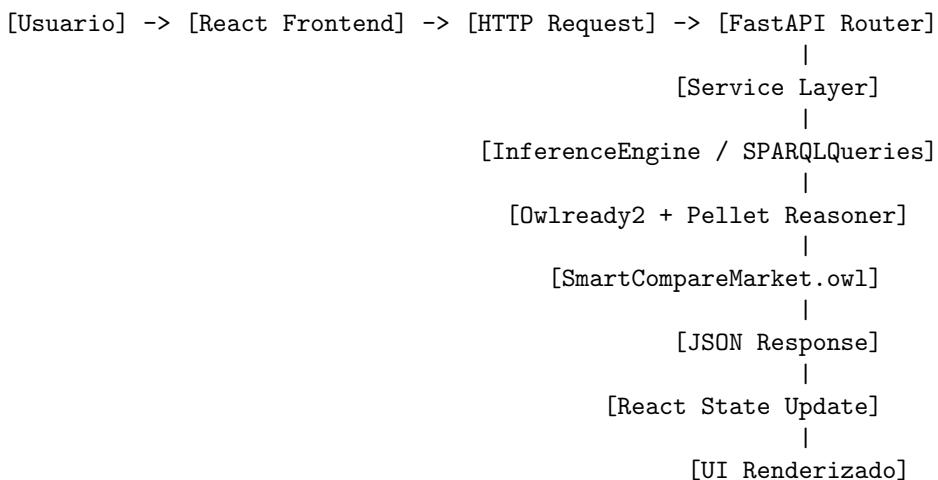
2. Arquitectura del Sistema

2.1. Diagrama de Arquitectura

El sistema implementa una arquitectura de **5 capas**:



2.2. Flujo de Datos

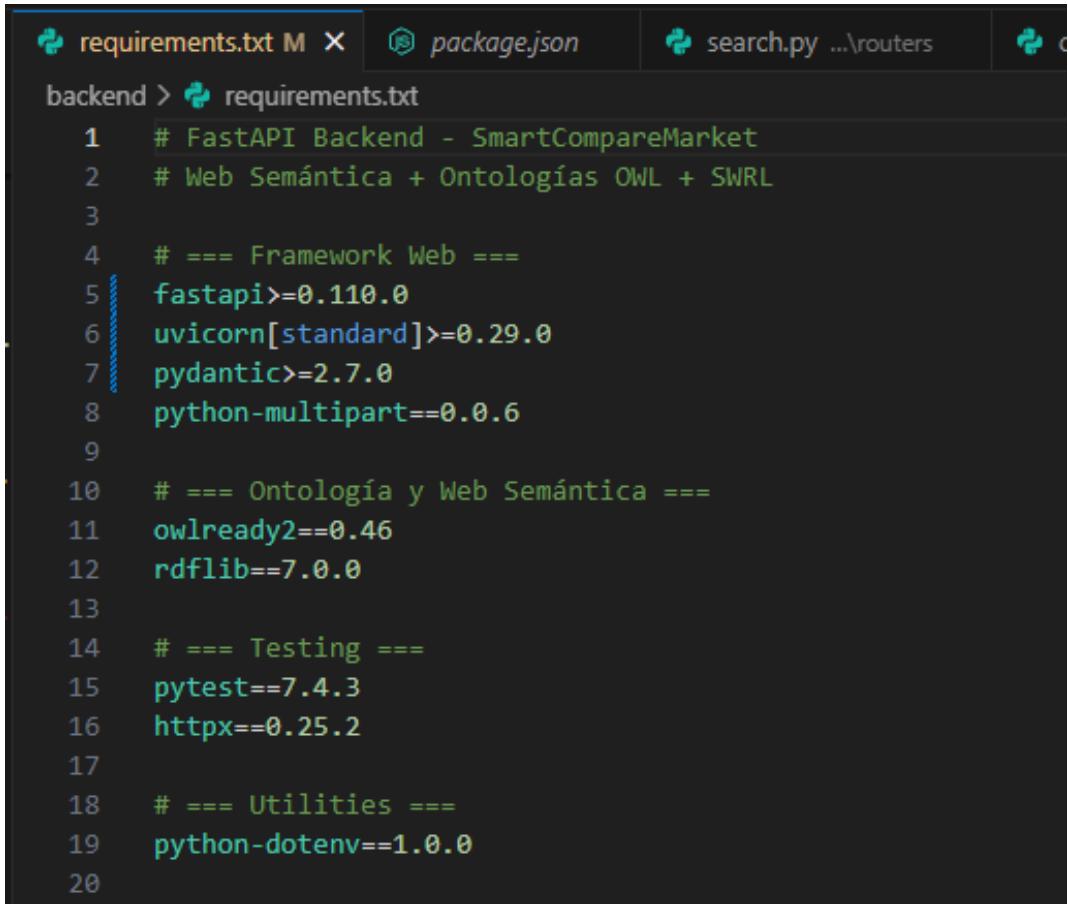


3. Tecnologías Utilizadas

3.1. Stack del Backend

Tecnología	Versión	Propósito
Python	3.11+	Lenguaje principal
FastAPI	0.109.0	Framework web REST
Uvicorn	0.27.0	Servidor ASGI
Pydantic	2.6.0	Validación de datos
Owlready2	0.46	Manipulación de ontologías OWL
RDFlib	7.0.0	Motor SPARQL
Pellet	2.4+	Razonador OWL+SWRL (requiere Java)

Cuadro 2: Stack tecnológico del backend



```
backend > requirements.txt
1 # FastAPI Backend - SmartCompareMarket
2 # Web Semántica + Ontologías OWL + SWRL
3
4 # === Framework Web ===
5 fastapi>=0.110.0
6 uvicorn[standard]>=0.29.0
7 pydantic>=2.7.0
8 python-multipart==0.0.6
9
10 # === Ontología y Web Semántica ===
11 owlready2==0.46
12 rdflib==7.0.0
13
14 # === Testing ===
15 pytest==7.4.3
16 httpx==0.25.2
17
18 # === Utilities ===
19 python-dotenv==1.0.0
20
```

Figura 1: Archivo requirements.txt mostrando las dependencias

3.2. Stack del Frontend

Tecnología	Versión	Propósito
React	18.3.1	Framework UI
TypeScript	5.8.3	Tipado estático
Vite	5.4.19	Build tool y dev server
Tailwind CSS	3.4.17	Framework CSS
Shadcn/UI	latest	Componentes UI
React Router	6.30.1	Enrutamiento SPA
Axios	1.13.2	Cliente HTTP
Lucide React	0.462.0	Iconografía

Cuadro 3: Stack tecnológico del frontend

```
frontend > package.json > {} dependencies
13  "dependencies": {
46    "cmdk": "^1.1.1",
47    "date-fns": "^3.6.0",
48    "embla-carousel-react": "^8.6.0",
49    "input-otp": "^1.4.2",
50    "lucide-react": "^0.462.0",
51    "next-themes": "^0.3.0",
52    "react": "^18.3.1",
53    "react-day-picker": "^8.10.1",
54    "react-dom": "^18.3.1",
55    "react-hook-form": "^7.61.1",
56    "react-resizable-panels": "^2.1.9",
57    "react-router-dom": "^6.30.1",
58    "recharts": "^2.15.4",
59    "sonner": "^1.7.4",
60    "tailwind-merge": "^2.6.0",
61    "tailwindcss-animate": "^1.0.7",
62    "vaul": "^0.9.9",
63    "zod": "^3.25.76"
64  },
65  "devDependencies": {
66    "@eslint/js": "^9.32.0",
67    "@tailwindcss/typography": "^0.5.16",
68    "@types/node": "^22.16.5",
69    "@types/react": "^18.3.23",
70    "@types/react-dom": "^18.3.7",
71    "@vitejs/plugin-react-swc": "^3.11.0",
72    "autoprefixer": "^10.4.21",
73    "eslint": "^9.32.0",
74    "eslint-plugin-react-hooks": "^5.2.0",
75    "eslint-plugin-react-refresh": "^0.4.20",
76    "globals": "^15.15.0",
77    "lovable-tagger": "^1.1.11",
78    "postcss": "^8.5.6",
79    "tailwindcss": "^3.4.17",
80    "typescript": "^5.8.3",
81    "typescript-eslint": "^8.38.0",
82    "vite": "^5.4.19"
83  }
84}
85}
```

Figura 2: Archivo package.json mostrando las dependencias

3.3. Tecnologías Semánticas

Tecnología	Descripción
OWL 2	Web Ontology Language 2 para modelar conocimiento
SWRL	Semantic Web Rule Language para reglas de inferencia
SPARQL	Lenguaje de consultas para RDF/OWL
Pellet	Razonador que soporta OWL 2 + SWRL

Cuadro 4: Tecnologías de Web Semántica

4. Estructura del Proyecto

4.1. Estructura de Directorios

```

WebsemanticaProyect/
|-- backend/                               # Servidor API
|   |-- api/                                # Configuracion de la API
|   |-- data/                               # Datos de configuracion
|   |   |-- comparison_weights.json          # Pesos para scoring
|   |-- models/                            # Modelos Pydantic
|   |   |-- product.py                      # Modelo de Producto
|   |   |-- comparison.py                  # Modelo de Comparacion
|   |   |-- recommendation.py            # Modelo de Recomendacion
|   |-- ontology/                          # Archivos de ontologia
|   |   |-- SmartCompareMarket.owl        # ONTOLOGIA PRINCIPAL (2900+ lineas)
|   |   |-- owl_helpers.py              # Utilidades para OWL
|   |-- reasoning/                         # Capa de razonamiento
|   |   |-- inference_engine.py        # Motor de inferencias
|   |   |-- swrl_engine.py             # Motor de reglas SWRL
|   |   |-- ontology_loader.py       # Cargador de ontologia
|   |-- routers/                           # Endpoints de la API
|   |   |-- products.py                # /api/v1/products
|   |   |-- compare.py                 # /api/v1/compare
|   |   |-- recommendations.py      # /api/v1/recommendations
|   |   |-- search.py                  # /api/v1/search
|   |   |-- swrl.py                    # /api/v1/swrl
|   |   |-- validation.py              # /api/v1/validate
|   |-- services/                          # Logica de negocio
|   |   |-- product_service.py        # Motor de comparacion (445 lineas)
|   |   |-- comparison_service.py    # Sistema de recomendaciones
|   |   |-- recommendation_service.py # Validacion de datos
|   |-- sparql/                           # Consultas SPARQL
|   |   |-- queries.py                # Consultas predefinidas
|   |-- main.py                            # PUNTO DE ENTRADA
|   |-- requirements.txt                 # Dependencias Python
|
|-- frontend/                             # Aplicacion React
|   |-- src/                                # Componentes reutilizables
|   |   |-- components/                   # Paginas principales
|   |   |-- pages/                      # Catalogo de productos
|   |   |   |-- HomePage.tsx           # Comparacion inteligente
|   |   |   |-- ComparePage.tsx       # Recomendaciones
|   |   |-- services/                  # Servicios HTTP
|   |   |-- App.tsx                     # Componente raiz
|   |-- package.json                       # Dependencias Node.js
|
|-- README.md                            # Documentacion principal
|-- MANUAL_USUARIO.md                   # Manual de usuario
|-- MANUAL_TECNICO.md                   # Manual tecnico

```

Nombre	Fecha de modificación	Tipo	Tamaño
.git	6/12/2025 04:44	Carpeta de archivos	
backend	6/12/2025 05:36	Carpeta de archivos	
frontend	6/12/2025 05:00	Carpeta de archivos	
.gitignore	29/11/2025 09:57	Documento de te...	1 KB
Latex_Manual_tecnico	6/12/2025 04:30	Archivo de origen ...	45 KB
Latex_Manual_usuario	6/12/2025 04:30	Archivo de origen ...	33 KB
README	6/12/2025 04:42	Archivo de origen ...	27 KB

Figura 3: Estructura de carpetas del proyecto en explorador de archivos

5. Ontología OWL 2

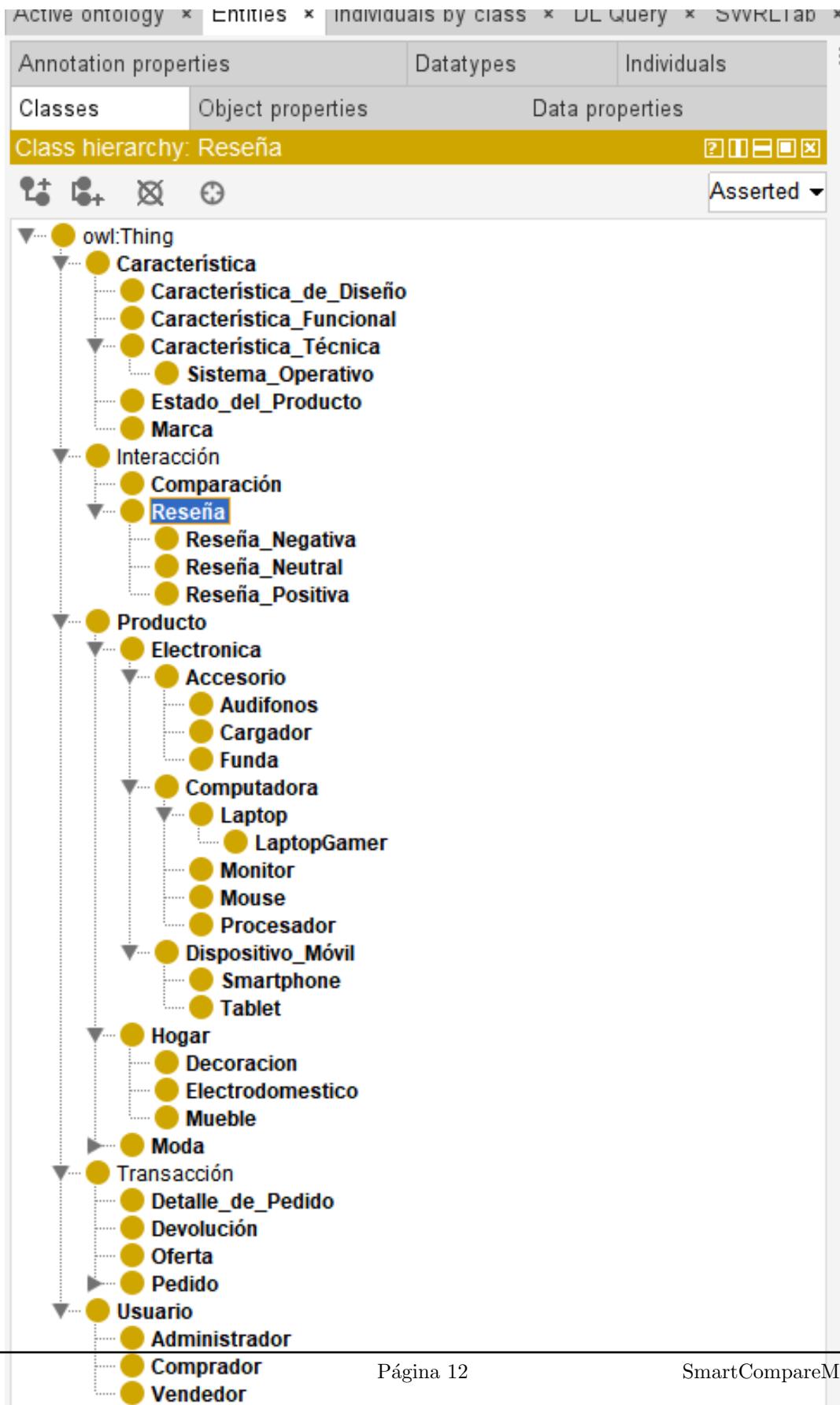
5.1. Características Generales

Propiedad	Valor
Archivo	backend/ontology/SmartCompareMarket.owl
Tamaño	~2,900 líneas
Namespace	http://smartcompare.com/ontologia#
Formato	RDF/XML

Cuadro 5: Características de la ontología

5.2. Jerarquía de Clases

```
owl:Thing
|-- Producto (raíz de productos)
|   |-- Electronica
|   |   |-- Smartphone
|   |   |-- Tablet
|   |   |-- Computadora
|   |       |-- Desktop
|   |       |-- Laptop
|   |           |-- LaptopGamer (inferida por SWRL)
|   |-- Moda
|   |   |-- Ropa
|   |   |-- Calzado
|   |-- Hogar
|       |-- Muebles
|       |-- Electrodomesticos
|
|-- Usuario
|   |-- Cliente
|   |-- Vendedor
|
|-- Resena
    |-- Resena_Positiva (inferida por SWRL)
    |-- Resena_Negativa (inferida por SWRL)
```



5.3. Propiedades de Datos (Data Properties)

Propiedad	Dominio	Rango	Descripción
tienePrecio	Producto	xsd:float	Precio en USD
tieneDescuento	Producto	xsd:float	Porcentaje de descuento
tieneRAM_GB	Electronica	xsd:integer	RAM en GB
tieneAlmacenamiento_GB	Electronica	xsd:integer	Almacenamiento en GB
tienePulgadas	Electronica	xsd:float	Tamaño de pantalla
resolucionPantalla	Electronica	xsd:string	Resolución (ej: "1920x1080")
bateriaCapacidad_mA	Electronica	xsd:integer	Capacidad de batería
procesadorModelo	Electronica	xsd:string	Modelo del procesador
procesadorVelocidad_GHz	Electronica	xsd:float	Velocidad del CPU
numeroNucleosCPU	Electronica	xsd:integer	Número de núcleos
tarjetaGrafica	Computadora	xsd:string	GPU
garantiaMeses	Producto	xsd:integer	Garantía en meses
esEquivalenteTecnico	Symmetric	Productos técnicamente equivalentes	
esMejorOpcionQue	Transitive	Producto A es mejor opción que B	
tieneMejorRAMQue	-	Comparación de RAM	

Cuadro 6: Propiedades de objeto de la ontología

- iPhone15_Barato (RAM: 6GB, 128GB, Precio: \$799)
- Samsung_Galaxy_S24 (RAM: 8GB, 256GB, Precio: \$899)
- Pixel_8_Pro (RAM: 12GB, 256GB, Precio: \$999)

Tablets:

- iPad_Pro_12 (RAM: 16GB, 512GB, Precio: \$1099)
- Samsung_Tab_S9 (RAM: 12GB, 256GB, Precio: \$849)

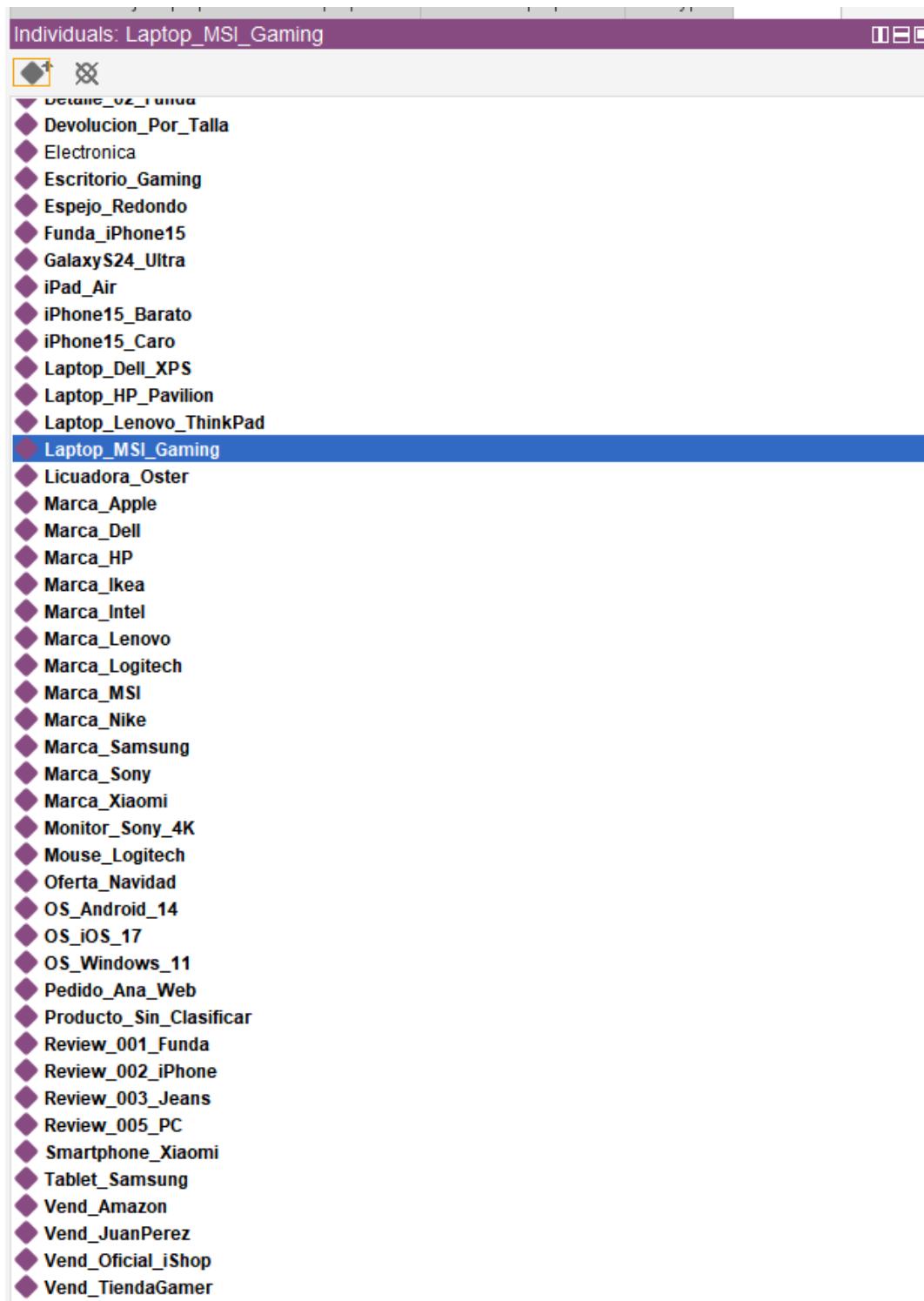


Figura 5: Individuos en Protégé, tab Individuals

6. Reglas SWRL

6.1. Ubicación en la Ontología

Las reglas SWRL se encuentran en las líneas **1964-2300** del archivo **SmartCompareMarket.owl**.

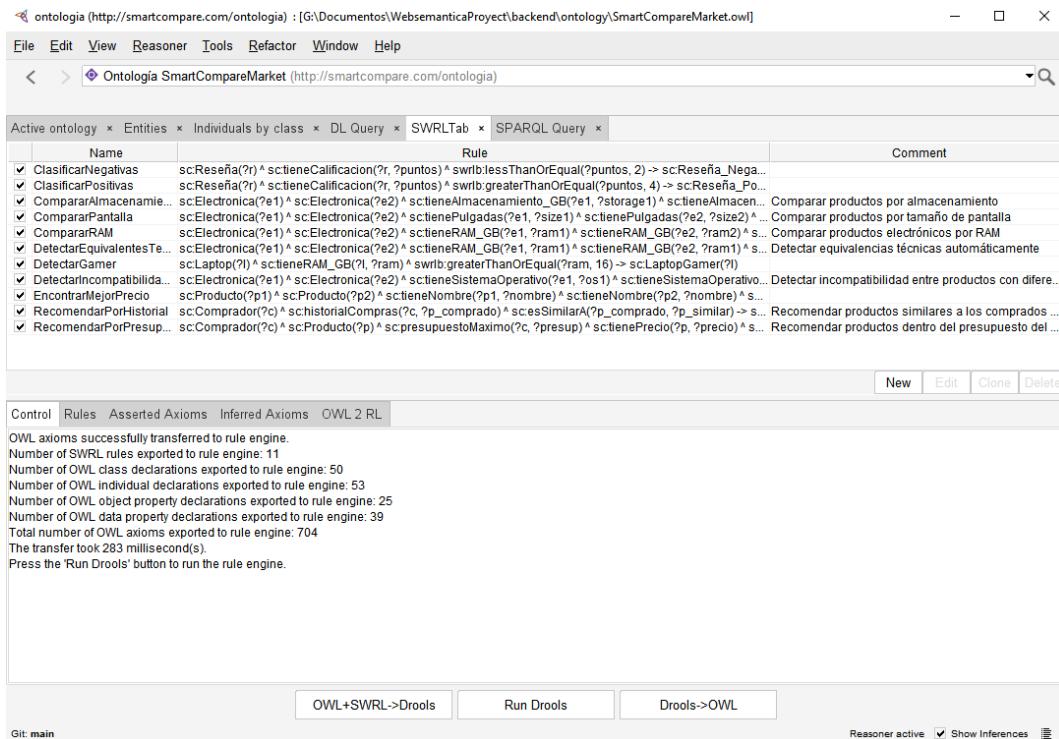
6.2. Reglas Implementadas

6.2.1. Regla 1: DetectarGamer

```
Laptop(?1) AND tieneRAM_GB(?1, ?ram) AND greaterThanOrEqual(?ram, 16)
-> LaptopGamer(?1)
```

Explicación técnica:

- **Antecedente:** Un individuo **?1** es de clase **Laptop** Y tiene propiedad **tieneRAM_GB** con valor **?ram** Y ese valor es ≥ 16
- **Consecuente:** El individuo **?1** se clasifica como miembro de la clase **LaptopGamer**
- **Efecto:** Subsunción automática por el razonador Pellet



The screenshot shows the Protégé interface with the "SWRLTab" selected. The rule "DetectarGamer" is listed in the table:

Name	Rule	Comment
DetectarGamer	sc:Laptop(?1) ^ sc:tieneRAM_GB(?1, ?ram) ^ swrlb:greaterThanOrEqual(?ram, 16) -> sc:LaptopGamer(?1)	

Below the table, the status bar indicates: "OWL axioms successfully transferred to rule engine. Number of SWRL rules exported to rule engine: 11 Number of OWL class declarations exported to rule engine: 50 Number of OWL individual declarations exported to rule engine: 53 Number of OWL object property declarations exported to rule engine: 25 Number of OWL data property declarations exported to rule engine: 39 Total number of OWL axioms exported to rule engine: 704 The transfer took 283 millisecond(s). Press the 'Run Drools' button to run the rule engine."

Figura 6: Regla DetectarGamer en Protégé, tab SWRL

6.2.2. Regla 2: EncontrarMejorPrecio

```
Producto(?p1) AND Producto(?p2) AND tieneNombre(?p1, ?n) AND tieneNombre(?p2, ?n)
AND tienePrecio(?p1, ?pr1) AND tienePrecio(?p2, ?pr2) AND lessThan(?pr1, ?pr2)
-> esMejorOpcionQue(?p1, ?p2)
```

Explicación: Dos productos con el mismo nombre pero diferente precio; el de menor precio se marca como “mejor opción que” el otro.

6.2.3. Regla 3: ClasificarPositivas

```
Resena(?r) AND tieneCalificacion(?r, ?cal) AND greaterThanOrEqual(?cal, 4)
-> Resena_Positiva(?r)
```

6.2.4. Regla 4: ClasificarNegativas

```
Resena(?r) AND tieneCalificacion(?r, ?cal) AND lessThanOrEqual(?cal, 2)
-> Resena_Negativa(?r)
```

6.3. Ejecución de Reglas

El razonador Pellet se ejecuta al iniciar el backend:

```
# backend/reasoning/ontology_loader.py
from owlready2 import sync_reasoner_pellet

def load_ontology():
    onto = get_ontology("SmartCompareMarket.owl").load()

    # Ejecutar razonador Pellet con soporte SWRL
    with onto:
        sync_reasoner_pellet(infer_property_values=True,
                             infer_data_property_values=True)

    return onto
```

```
PS G:\Documentos\WebsemanticaProyect\backend> python main.py
[OK] Ontología cargada: G:\Documentos\WebsemanticaProyect\backend\ontology\SmartCompareMarket
- Clases: 49
- Individuos: 52
- Object Properties: 24
- Data Properties: 38
[REASONER] Ejecutando razonador Pellet...
[OK] Razonador Pellet ejecutado exitosamente
[SWRL] Reglas SWRL que deberían aplicarse:
1. DetectarGamer (RAM >= 16GB -> LaptopGamer)
2. EncontrarMejorPrecio (precio menor -> esMejorOpcionQue)
3. ClasificarPositivas (cal >= 4 -> Resena_Positiva)
4. ClasificarNegativas (cal <= 2 -> Resena_Negativa)
[OK] Ontología cargada en RDFlib: 1327 triples
```

Figura 7: Log del backend mostrando Razonador Pellet ejecutado exitosamente

7. API REST - Endpoints

7.1. Base URL

<http://localhost:5000/api/v1>

7.2. Endpoints de Productos

7.2.1. GET /products

Obtener todos los productos.

```
curl http://localhost:5000/api/v1/products
```

Respuesta:

```
{
  "products": [
    {
      "id": "Laptop_Dell_XPS",
      "name": "Dell XPS 15",
      "category": "Laptop",
      "types": ["Producto", "Electronica", "Computadora", "Laptop", "LaptopGamer"],
      "price": 1599.99,
      "ram_gb": 32,
      "storage_gb": 1024,
      "rating": 4.7
    }
  ],
  "count": 60
}
```

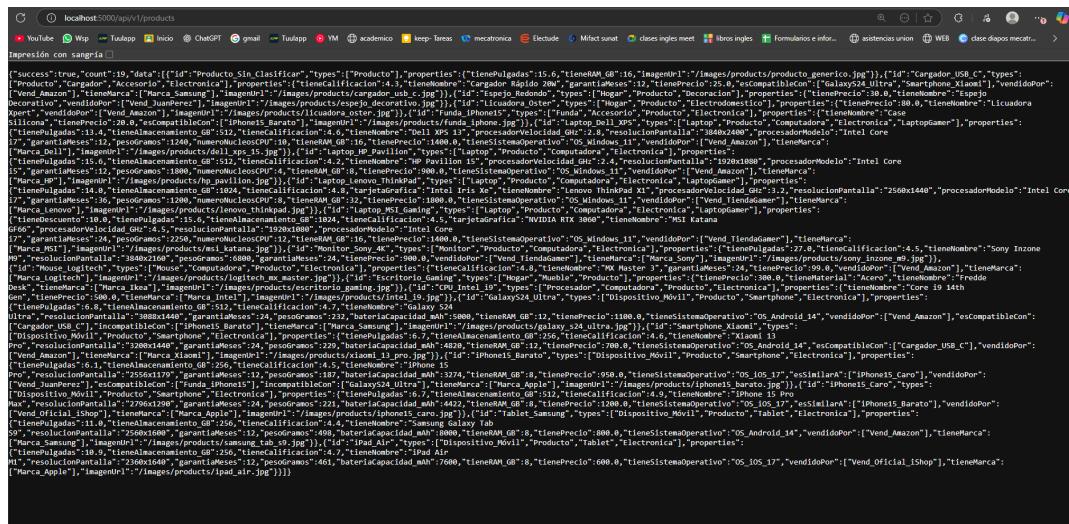


Figura 8: Respuesta JSON de /api/v1/products en navegador

7.2.2. GET /products/{product.id}/relationships

Obtener relaciones de un producto.

```
curl http://localhost:5000/api/v1/products/iPhone15_Barato/relationships
```

Respuesta:

```
{  
    "product_id": "iPhone15_Barato",  
    "compatible": ["Cargador_USB_C", "Funda_iPhone15"],  
    "incompatible": ["Cargador_MicroUSB"],  
    "similar": ["iPhone15_Pro", "Samsung_Galaxy_S24"],  
    "better_than": ["iPhone14_Base"]  
}
```

7.3. Endpoint de Comparación

7.3.1. POST /compare

```
curl -X POST http://localhost:5000/api/v1/compare \
-H "Content-Type: application/json" \
-d '{"products": ["Laptop_Dell_XPS", "Laptop_MSI_Gaming"]}'
```

Request Body:

```
{  
  "products": ["Laptop_Dell_XPS", "Laptop_MSI_Gaming", "Laptop_HP_Pavilion"]  
}
```

Respuesta:

```
{  
  "winner": "Laptop_Dell_XPS",  
  "winner_score": 87.5,  
  "comparison_table": {  
    "Precio": {"Laptop_Dell_XPS": 1599, "Laptop_MSI_Gaming": 1299},  
    "RAM (GB)": {"Laptop_Dell_XPS": 32, "Laptop_MSI_Gaming": 16},  
    "Almacenamiento (GB)": {"Laptop_Dell_XPS": 1024, "Laptop_MSI_Gaming": 512}  
  },  
  "swrl_inferences": [  
    {  
      "type": "esMejorOpcionQue",  
      "subject": "Laptop_Dell_XPS",  
      "object": "Laptop_MSI_Gaming"  
    }  
  ],  
  "reason": "Laptop_Dell_XPS gana con 87.5 puntos"  
}
```

Code	Description
200	<p>Comparación exitosa</p> <p>Media type</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;"> application/json </div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>{ "success": true, "comparison": { "product1": { "id": "iPhone15_Barato", "nombre": "iPhone 15 Pro", "precio": 950, "ram": 8 }, "product2": { "id": "iPhone15_Caro", "nombre": "iPhone 15 Pro", "precio": 1200, "ram": 8 }, "winner": "iPhone15_Barato", "reason": "Mejor precio para el mismo producto", "swrl_inference": { "esMejorOpcionQue": true, "rule": "EncontrarMejorPrecio" } } }</pre>

Figura 9: Respuesta de comparación mostrando ganador

7.4. Endpoint de Recomendaciones

7.4.1. POST /recommendations

```
curl -X POST http://localhost:5000/api/v1/recommendations \
-H "Content-Type: application/json" \
-d '{"budget": 1500, "preferred_category": "Laptop", "min_ram": 16}'
```

Respuesta:

```
{
  "recommendations": [
    {
      "product_id": "Laptop_MSI_Gaming",
      "name": "MSI GF65 Thin",
      "score": 92.5,
      "match_percentage": 95,
      "reason": "Laptop Gamer detectado (SWRL) + Excelente relacion calidad-precio",
    }
  ]
}
```

```
        "price": 1299
    }
],
"total_matches": 5
}
```

Code	Description
200	Successful Response Media type <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> Controls Accept header. Example Value Schema
422	Validation Error

Figura 10: Respuesta de recomendaciones con scores y razones

7.5. Endpoint de Búsqueda SPARQL

7.5.1. GET /search

```
curl
"http://localhost:5000/api/v1/search?category=Laptop&min_price=1000&max_price=1500&min_ram=16"
```

Parámetros:

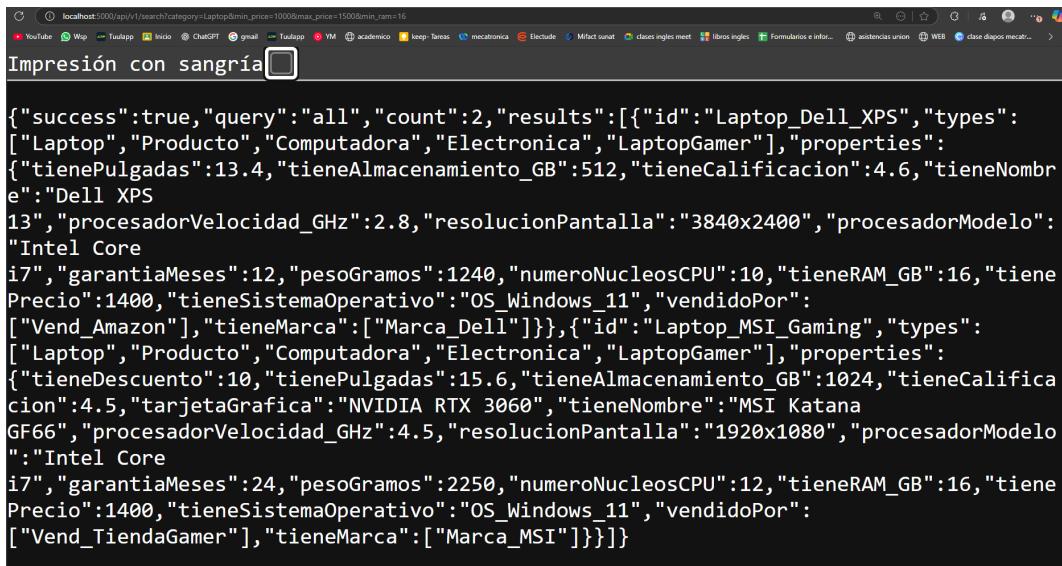
Parámetro	Tipo	Descripción
text	string	Búsqueda de texto libre
category	string	Categoría (Laptop, Smartphone, Tablet)
min_price	float	Precio mínimo
max_price	float	Precio máximo
min_ram	integer	RAM mínima en GB

Cuadro 7: Parámetros del endpoint de búsqueda

Consulta SPARQL generada internamente:

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

SELECT ?product ?name ?price ?ram WHERE {
  ?product rdf:type ns:Laptop .
  ?product ns:tienePrecio ?price .
  ?product ns:tieneRAM_GB ?ram .
  FILTER (?price >= 1000 && ?price <= 1500)
  FILTER (?ram >= 16)
}
```



```
{"success":true,"query":"all","count":2,"results":[{"id":"Laptop_Dell_XPS","types":["Laptop","Producto","Computadora","Electronica","LaptopGamer"],"properties":{"tienePulgadas":13.4,"tieneAlmacenamiento_GB":512,"tieneCalificacion":4.6,"tieneNombre":"Dell XPS","procesadorVelocidad_GHz":2.8,"resolucionPantalla":"3840x2400","procesadorModelo":"Intel Core i7","garantiaMeses":12,"pesoGramos":1240,"numeroNucleosCPU":10,"tieneRAM_GB":16,"tienePrecio":1400,"tieneSistemaOperativo":"OS_Windows_11","vendidoPor":[{"Vend_Amazon"}, {"tieneMarca":["Marca_Dell"]}], {"id":"Laptop_MSI_Gaming","types":["Laptop","Producto","Computadora","Electronica","LaptopGamer"],"properties":{"tieneDescuento":10,"tienePulgadas":15.6,"tieneAlmacenamiento_GB":1024,"tieneCalificacion":4.5,"tarjetaGrafica":"NVIDIA RTX 3060","tieneNombre":"MSI Katana GF66","procesadorVelocidad_GHz":4.5,"resolucionPantalla":"1920x1080","procesadorModelo":"Intel Core i7","garantiaMeses":24,"pesoGramos":2250,"numeroNucleosCPU":12,"tieneRAM_GB":16,"tienePrecio":1400,"tieneSistemaOperativo":"OS_Windows_11","vendidoPor":[{"Vend_TiendaGamer"}, {"tieneMarca":["Marca_MSI"]}]}}]}
```

Figura 11: Respuesta de búsqueda filtrada

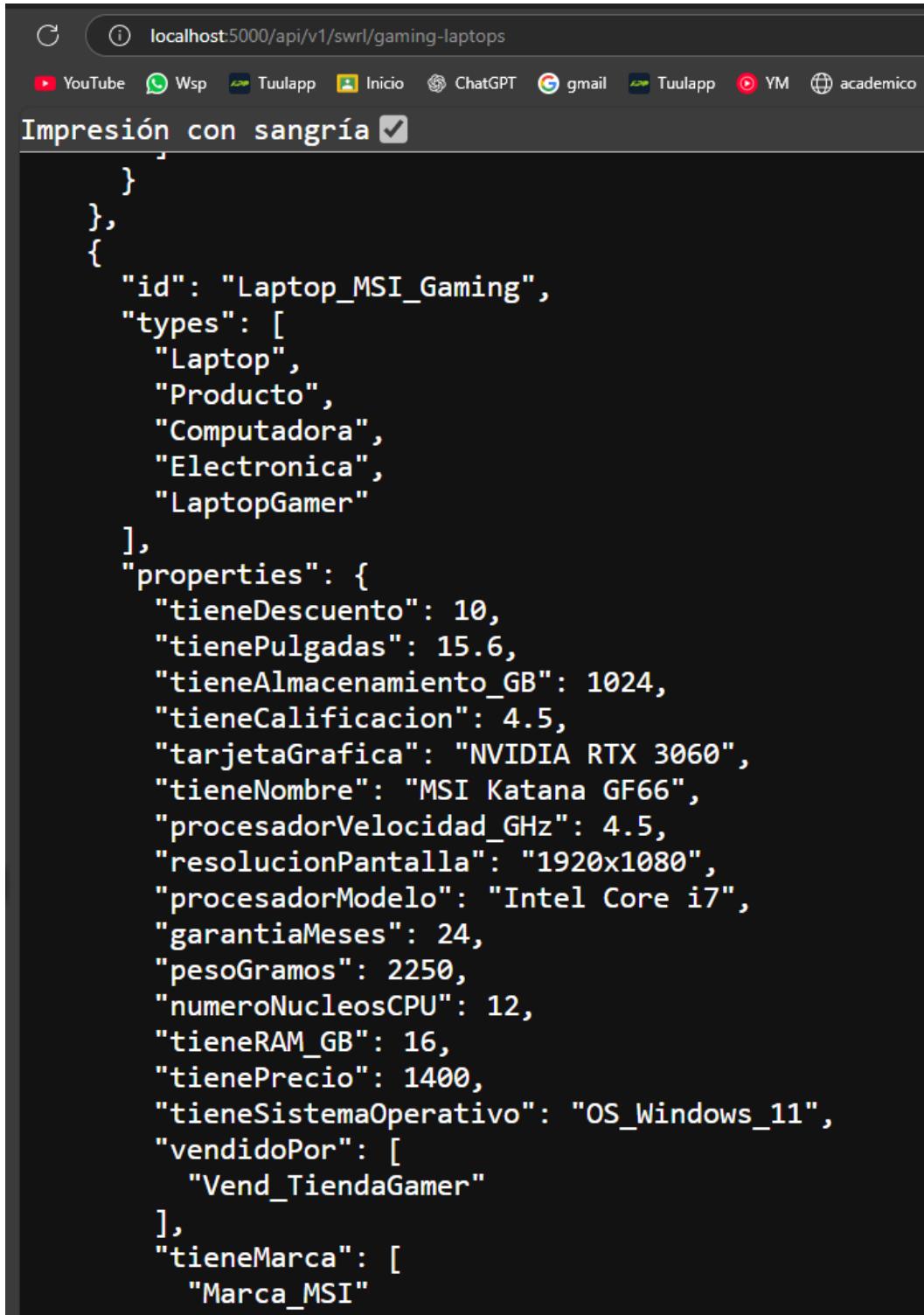
7.6. Endpoints SWRL

7.6.1. GET /swrl/gaming-laptops

```
curl http://localhost:5000/api/v1/swrl/gaming-laptops
```

Respuesta:

```
{  
  "rule": "DetectarGamer",  
  "description": "Laptops con RAM >= 16GB clasificadas como LaptopGamer",  
  "results": [  
    {"id": "Laptop_Dell_XPS", "ram": 32, "classified_as": "LaptopGamer"},  
    {"id": "Laptop_MSI_Gaming", "ram": 16, "classified_as": "LaptopGamer"}  
,  
  "count": 3  
}
```



```
Impresión con sangría ✓
{
  "id": "Laptop_MS1_Gaming",
  "types": [
    "Laptop",
    "Producto",
    "Computadora",
    "Electronica",
    "LaptopGamer"
  ],
  "properties": {
    "tieneDescuento": 10,
    "tienePulgadas": 15.6,
    "tieneAlmacenamiento_GB": 1024,
    "tieneCalificacion": 4.5,
    "tarjetaGrafica": "NVIDIA RTX 3060",
    "tieneNombre": "MSI Katana GF66",
    "procesadorVelocidad_GHz": 4.5,
    "resolucionPantalla": "1920x1080",
    "procesadorModelo": "Intel Core i7",
    "garantiaMeses": 24,
    "pesoGramos": 2250,
    "numeroNucleosCPU": 12,
    "tieneRAM_GB": 16,
    "tienePrecio": 1400,
    "tieneSistemaOperativo": "OS_Windows_11",
    "vendidoPor": [
      "Vend_TiendaGamer"
    ],
    "tieneMarca": [
      "Marca_MS1"
    ]
  }
}
```

Figura 12: Respuesta de gaming-laptops mostrando productos clasificados

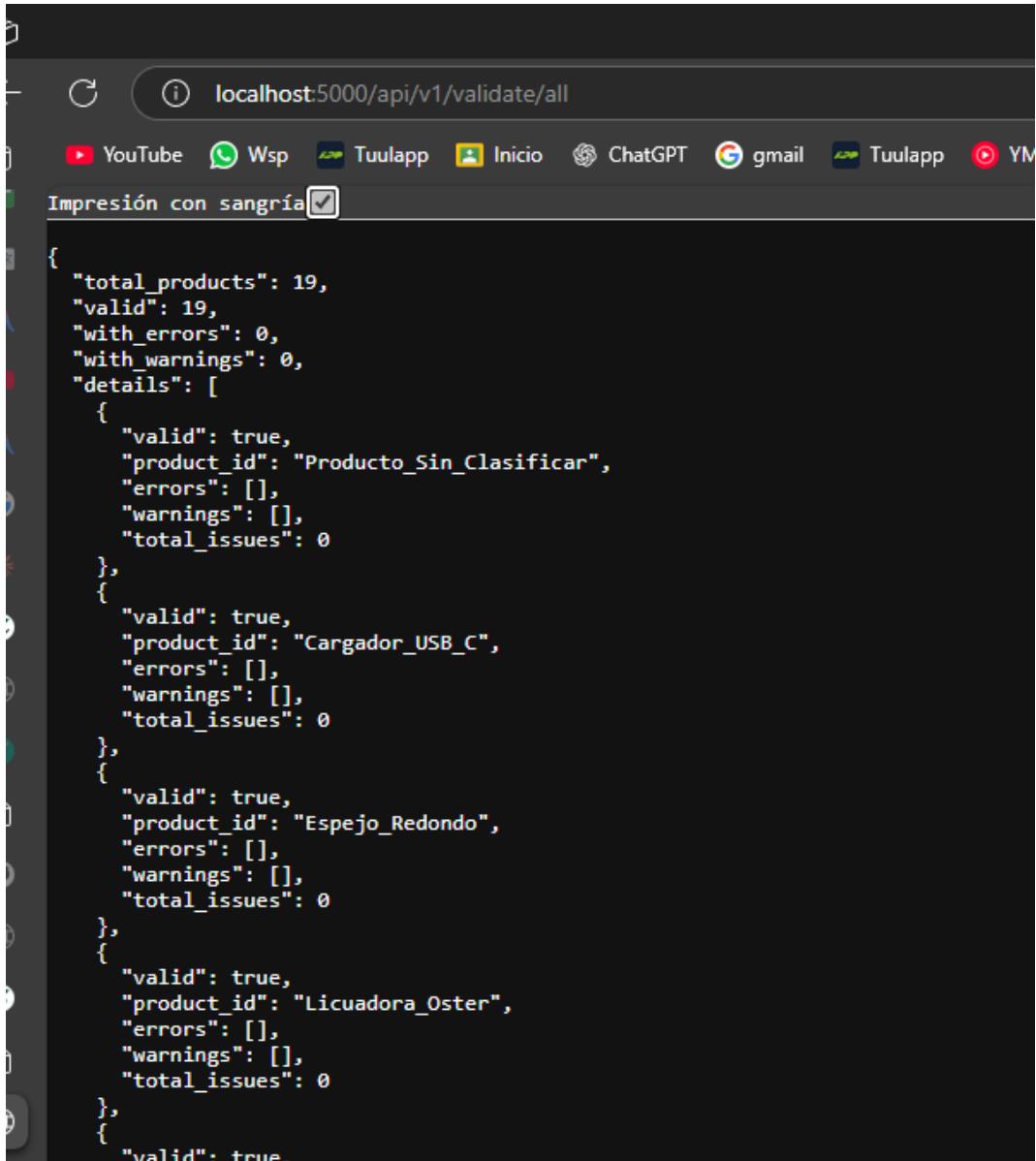
7.7. Endpoint de Validación

7.7.1. GET /validate/all

```
curl http://localhost:5000/api/v1/validate/all
```

Respuesta:

```
{  
  "summary": {  
    "total_products": 60,  
    "valid": 58,  
    "with_errors": 1,  
    "with_warnings": 3  
  },  
  "details": [...]  
}
```



```
localhost:5000/api/v1/validate/all

Impresión con sangría
```

```
{  
    "total_products": 19,  
    "valid": 19,  
    "with_errors": 0,  
    "with_warnings": 0,  
    "details": [  
        {  
            "valid": true,  
            "product_id": "Producto_Sin_Clasicar",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Cargador_USB_C",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Espejo_Redondo",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Licuadora_Oster",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Televisor_Smart",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Monitor_Periferico",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Altavoz_Stereo",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Proyector_Multimedios",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Cable_Adaptador",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Funda_Portatil",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Memoria_SSD",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "CPU_Procesador",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Placa_Madre",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "GPU_Grafica",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Fuente_Poder",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Caja_Ratn",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Raton_Periferico",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Teclado_Periferico",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Alimentador",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Unidad_Optica",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        },  
        {  
            "valid": true,  
            "product_id": "Bandeja",  
            "errors": [],  
            "warnings": [],  
            "total_issues": 0  
        }  
    ]  
}
```

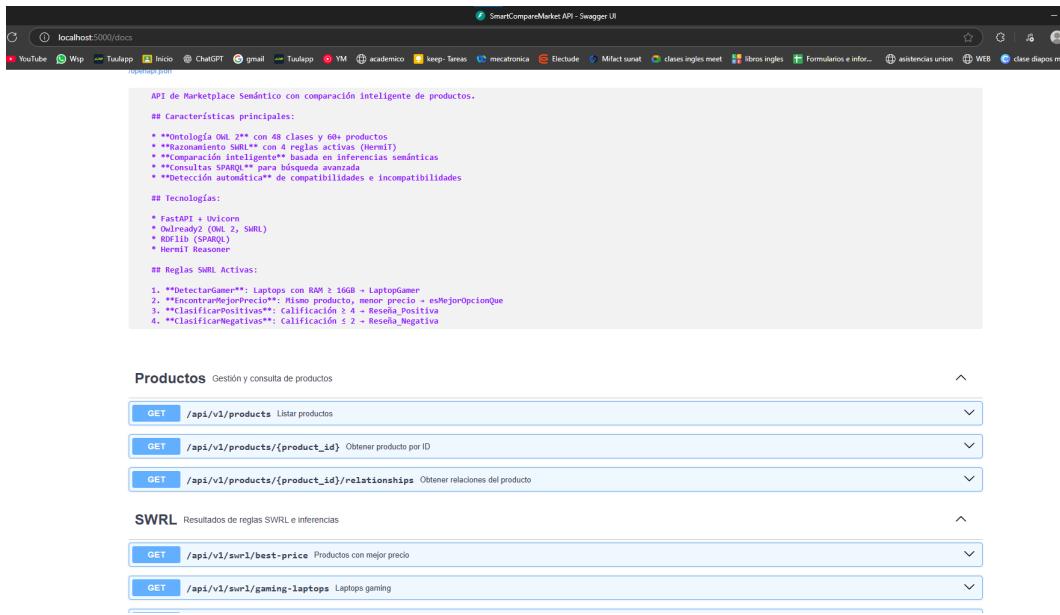
Figura 13: Respuesta de validación masiva

7.8. Documentación Interactiva

FastAPI genera documentación automática:

URL	Descripción
http://localhost:5000/docs	Swagger UI (interactiva)
http://localhost:5000/redoc	ReDoc (documentación)

Cuadro 8: URLs de documentación de la API



The screenshot shows the Swagger UI interface for the SmartCompareMarket API. At the top, there's a navigation bar with links like YouTube, Wip, Tuleapp, Inicio, ChatGPT, gmail, Tuleapp, YM, academico, keep-tareas, mecatronica, Reciente, Millet sunat, clases ingles meet, libros ingles, Formularios e informes, asistencias union, WEB, clase diapositivas, and reportes pdf.

The main content area is titled "SmartCompareMarket API - Swagger UI". It displays the API documentation for the "Marketplace Semántico con comparación inteligente de productos".

Key sections shown include:

- API de Marketplace Semántico con comparación inteligente de productos.**
- Características principales:**
 - **Ontología OML 2** con 48 clases y 60+ productos
 - **Razonamiento SHDL** con 4 reglas activas (HermiT)
 - **Integración completa** basada en reglas y relaciones semánticas
 - **Consultas SPARQL** para consulta avanzada
 - **Detección automática** de compatibilidades e incompatibilidades
- Tecnologías:**
 - FactotPI + Unicorn
 - ÓrdenePI 2 (OML 2, SHDL)
 - RDQL (SPARQL)
 - HermiT Reasoner
- Reglas SWRL Activas:**
 - **DetectarGame**: Laptops con RAM ≥ 16GB → LaptopGamer
 - **EncontrarMejorPrecio**: Misma producto, menor precio → esMejorOpcionQue
 - **ClasificarPositivo**: Calificación ≥ 4 → Reseña_Positiva
 - **ClasificarNegativo**: Calificación ≤ 2 → Reseña_Negativa

Below these sections, there are two expandable sections:

- Productos** (Gestión y consulta de productos)
 - GET /api/v1/products** Listar productos
 - GET /api/v1/products/{product_id}** Obtener producto por ID
 - GET /api/v1/products/{product_id}/relationships** Obtener relaciones del producto
- SWRL** (Resultados de reglas SWRL e inferencias)
 - GET /api/v1/swrl/best-price** Productos con mejor precio
 - GET /api/v1/swrl/gaming-laptops** Laptops gaming

Figura 14: Página de Swagger UI en /docs mostrando todos los endpoints

8. Servicios del Backend

8.1. ComparisonService

Ubicación: backend/services/comparison_service.py (445 líneas)

Responsabilidad: Motor de comparación inteligente con scoring multi-factor.

8.1.1. Algoritmo de Scoring

```
WEIGHTS = {
    "battery": 0.20,      # Mayor es mejor
    "rating": 0.18,       # Mayor es mejor
    "price": 0.14,        # MENOR es mejor (invertido)
    "resolution": 0.10,   # Mayor es mejor
    "ram": 0.10,          # Mayor es mejor
    "storage": 0.10,      # Mayor es mejor
    "warranty": 0.07,     # Mayor es mejor
    "screen": 0.06,       # Mayor es mejor
    "weight": 0.05        # MENOR es mejor (invertido)
}

REFERENCE_VALUES = {
    "battery": 10000,     # 10000 mAh = 100 puntos
    "rating": 5.0,         # 5 estrellas = 100 puntos
    "price": 3000,         # $3000 = 0 puntos (precio maximo)
    "ram": 64,             # 64GB = 100 puntos
    "storage": 2048,       # 2TB = 100 puntos
}
```

8.1.2. Cálculo del Score

```
def _calculate_score(self, product: dict) -> float:
    score = 0

    for factor, weight in WEIGHTS.items():
        value = product.get(factor, 0)
        reference = REFERENCE_VALUES[factor]

        if factor in ["price", "weight"]: # Menor es mejor
            normalized = max(0, 100 - (value / reference * 100))
        else: # Mayor es mejor
            normalized = min(100, value / reference * 100)

        score += normalized * weight

    # Bonus por reglas SWRL
    if self._has_swrl_bonus(product):
        score += 2 # Bonus por cada relacion "esMejorOpcionQue"

    return round(score, 2)
```

```

backend > services > comparison_service.py > ...
1 """
2 Servicio de Comparación Inteligente - DÍA 2
3 Motor de comparación entre productos usando inferencias semánticas
4 """
5 import sys
6 from pathlib import Path
7 from typing import List, Dict, Any, Optional
8
9 sys.path.insert(0, str(Path(__file__).resolve().parent.parent))
10
11 from ontology.loader import get_ontology
12 from reasoning.inference_engine import InferenceEngine
13 from services.product_service import ProductService
14 from utils.owl_helpers import individual_to_dict
15
16
17 class ComparisonService:
18 """
19     Servicio para comparación inteligente de productos.
20
21     Utiliza:
22         - InferenceEngine para relaciones SWRL
23         - ProductService para datos de productos
24         - Scoring basado en múltiples factores
25 """
26
27     def __init__(self):
28         self.onto = get_ontology()
29         self.inference_engine = InferenceEngine(self.onto)
30         self.product_service = ProductService()
31
32     def compare_products(self, product_ids: List[str]) -> Dict[str, Any]:
33 """
34     Compara múltiples productos y determina el ganador.
35
36     Args:
37         product_ids: Lista de IDs de productos a comparar
38
39     Returns:
40         Diccionario con comparación completa
41 """

```

Figura 15: Código de calculate score en comparisonservice.py

8.2. RecommendationService

Ubicación: backend/services/recommendation_service.py (277 líneas)

Responsabilidad: Sistema de recomendaciones personalizadas.

8.2.1. Sistema de Scoring para Recomendaciones

```
def _calculate_recommendation_score(self, product: dict, preferences: dict) -> float:
    score = 0

    # Factor 1: Presupuesto (30 puntos)
    if product["price"] <= preferences["budget"]:
        budget_usage = product["price"] / preferences["budget"]
        score += 30 * budget_usage

    # Factor 2: Calificación (25 puntos)
    score += product["rating"] * 5 # 5 puntos por estrella

    # Factor 3: RAM (15 puntos)
    if product["ram"] >= preferences.get("min_ram", 0):
        score += 15

    # Factor 4: Almacenamiento (10 puntos)
    if product["storage"] >= preferences.get("min_storage", 0):
        score += 10

    # Bonus SWRL
    if "LaptopGamer" in product.get("types", []):
        score += 10 # Bonus por ser Laptop Gamer

    return score
```

8.3. ValidationService

Ubicación: backend/services/validation_service.py (152 líneas)

Reglas de Validación:

Tipo	Condición
Error	Precio negativo
Error	RAM negativa o > 512GB
Error	Almacenamiento negativo o > 10TB
Error	Calificación fuera de rango 0-5
Advertencia	Precio > \$100,000
Advertencia	RAM > 128GB

Cuadro 9: Reglas de validación implementadas

9. Instalación para Desarrolladores

9.1. Prerrequisitos

Software	Versión	Verificación
Python	3.11+	<code>python --version</code>
Node.js	18+	<code>node --version</code>
npm	9+	<code>npm --version</code>
Java JDK	11+	<code>java -version</code>
Git	Cualquiera	<code>git --version</code>

Cuadro 10: Prerrequisitos de instalación

```
PS G:\Documentos\python --versionoyect\backend>
Python 3.13.3
PS G:\Documentos\WebsemanticaProyect\backend> node --version
v22.15.0
PS G:\Documentos\WebsemanticaProyect\backend> npm --version
10.9.2
PS G:\Documentos\WebsemanticaProyect\backend> java -version
java version "21.0.1" 2023-10-17 LTS
Java(TM) SE Runtime Environment (build 21.0.1+12-LTS-29)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.1+12-LTS-29, mixed mode, sharing)
PS G:\Documentos\WebsemanticaProyect\backend> git --version
git version 2.47.1.windows.1
PS G:\Documentos\WebsemanticaProyect\backend> █
```

Figura 16: Terminal mostrando todos los comandos de verificación

9.2. Clonar el Repositorio

```
git clone https://github.com/AlvaroMachaca0503/Web_Semantica.git
cd Web_Semantica
```

9.3. Configuración del Backend

```
# 1. Navegar al backend
cd backend

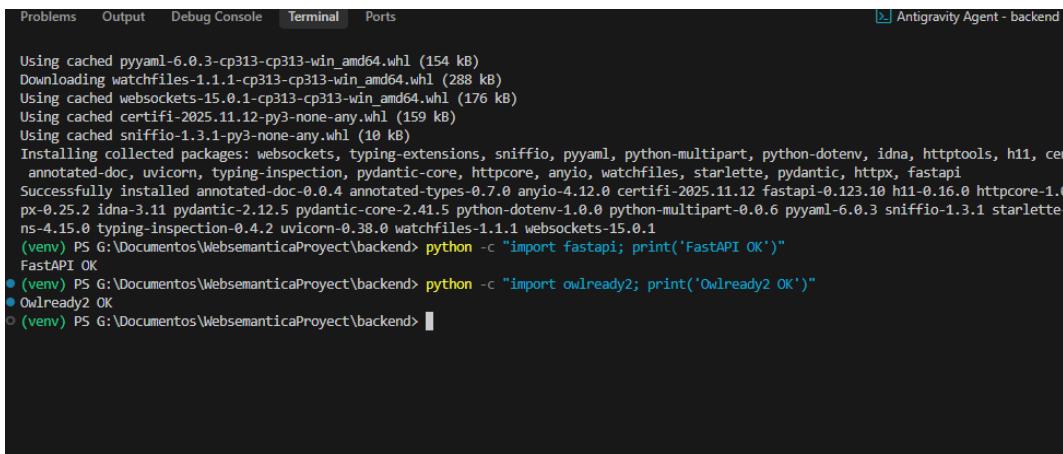
# 2. Crear entorno virtual
python -m venv venv

# 3. Activar entorno virtual
# Windows:
```

```
venv\Scripts\activate
# Linux/Mac:
source venv/bin/activate

# 4. Instalar dependencias
pip install -r requirements.txt

# 5. Verificar instalacion
python -c "import owlready2; print('Owlready2 OK')"
python -c "import fastapi; print('FastAPI OK')"
```



Terminal output:

```
Using cached pyyaml-6.0.3-cp313-cp313-win_amd64.whl (154 kB)
Downloading watchfiles-1.1.1-cp313-cp313-win_amd64.whl (288 kB)
Using cached websockets-15.0.1-cp313-cp313-win_amd64.whl (176 kB)
Using cached certifi-2025.11.12-py3-none-any.whl (159 kB)
Using cached sniffio-1.3.1-py3-none-any.whl (10 kB)
Installing collected packages: websockets, typing-extensions, sniffio, pyyaml, python-multipart, python-dotenv, idna, httptools, h11, certifi, annotated-doc, uvicorn, typing-inspection, pydantic-core, httpcore, aiohttp, watchfiles, starlette, pydantic, httpx, fastapi
SuccessFully installed annotated-doc-0.0.4 annotated-types-0.7.0 aiohttp-4.12.0 certifi-2025.11.12 fastapi-0.123.10 h11-0.16.0 httpcore-1.0.0 px-0.25.2 idna-3.11 pydantic-2.12.5 pydantic-core-2.41.5 python-dotenv-1.0.0 python-multipart-0.0.6 pyyaml-6.0.3 sniffio-1.3.1 starlette-ns-4.15.0 typing-inspection-0.4.2 uvicorn-0.38.0 watchfiles-1.1.1 websockets-15.0.1
(venv) PS G:\Documentos\WebsemanticaProyect\backend> python -c "import fastapi; print('FastAPI OK')"
FastAPI OK
● (venv) PS G:\Documentos\WebsemanticaProyect\backend> python -c "import owlready2; print('Owlready2 OK')"
Owlready2 OK
(venv) PS G:\Documentos\WebsemanticaProyect\backend>
```

Figura 17: Terminal con entorno virtual activado y verificaciones exitosas

9.4. Configuración del Frontend

```
# 1. Navegar al frontend
cd frontend

# 2. Instalar dependencias
npm install

# 3. Verificar instalacion
npm list react
```

10. Ejecución y Despliegue

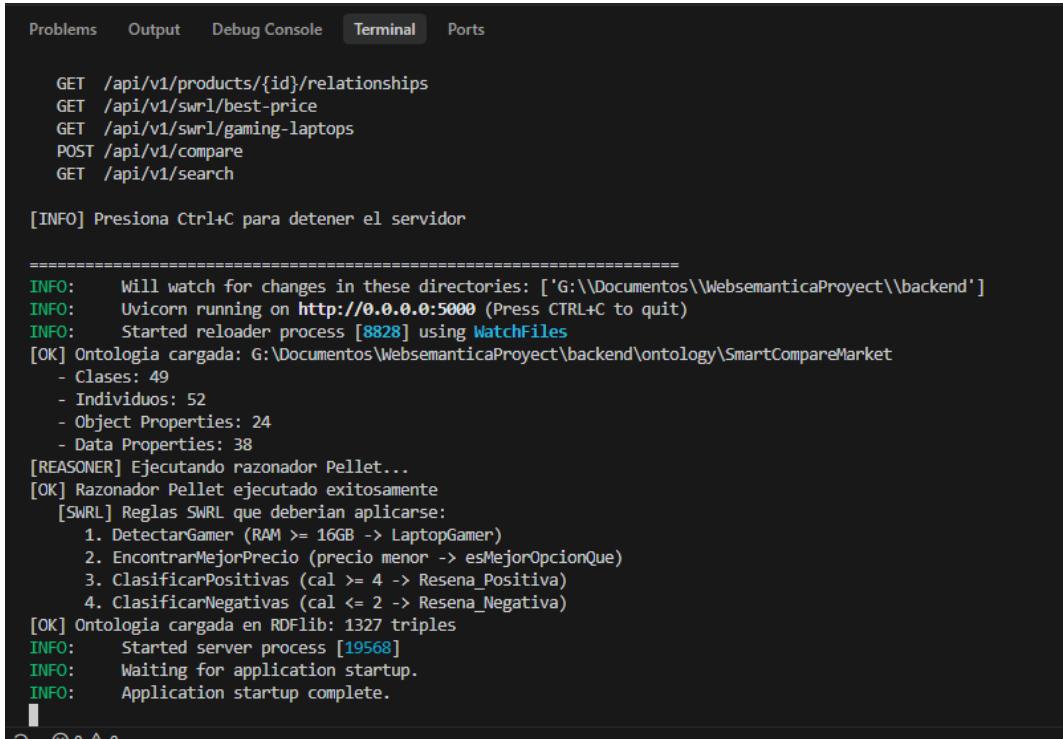
10.1. Modo Desarrollo

Terminal 1 - Backend:

```
cd backend
venv\Scripts\activate # Windows
python main.py
```

Logs esperados:

```
[INFO] Cargando ontología SmartCompareMarket.owl...
[OK] Ontología cargada: 60 productos encontrados
[INFO] Ejecutando razonador Pellet...
[OK] Razonador Pellet ejecutado exitosamente
[OK] Reglas SWRL aplicadas: DetectarGamer, EncontrarMejorPrecio...
INFO:     Uvicorn running on http://0.0.0.0:5000
```



```
Problems   Output   Debug Console   Terminal   Ports

GET /api/v1/products/{id}/relationships
GET /api/v1/swrl/best-price
GET /api/v1/swrl/gaming-laptops
POST /api/v1/compare
GET /api/v1/search

[INFO] Presiona Ctrl+C para detener el servidor
=====
INFO: Will watch for changes in these directories: ['G:\\Documentos\\Websemanticaproject\\backend']
INFO: Uvicorn running on http://0.0.0.0:5000 (Press CTRL+C to quit)
INFO: Started reloader process [8828] using WatchFiles
[OK] Ontología cargada: G:\\Documentos\\Websemanticaproject\\backend\\ontology\\SmartCompareMarket
- Clases: 49
- Individuos: 52
- Object Properties: 24
- Data Properties: 38
[REASONER] Ejecutando razonador Pellet...
[OK] Razonador Pellet ejecutado exitosamente
[SWRL] Reglas SWRL que deberían aplicarse:
  1. DetectarGamer (RAM >= 16GB -> LaptopGamer)
  2. EncontrarMejorPrecio (precio menor -> esMejorOpcionQue)
  3. ClasificarPositivas (cal >= 4 -> Resena_Positiva)
  4. ClasificarNegativas (cal <= 2 -> Resena_Negativa)
[OK] Ontología cargada en RDFlib: 1327 triples
INFO: Started server process [19568]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

Figura 18: Terminal del backend con todos los logs de inicio exitosos

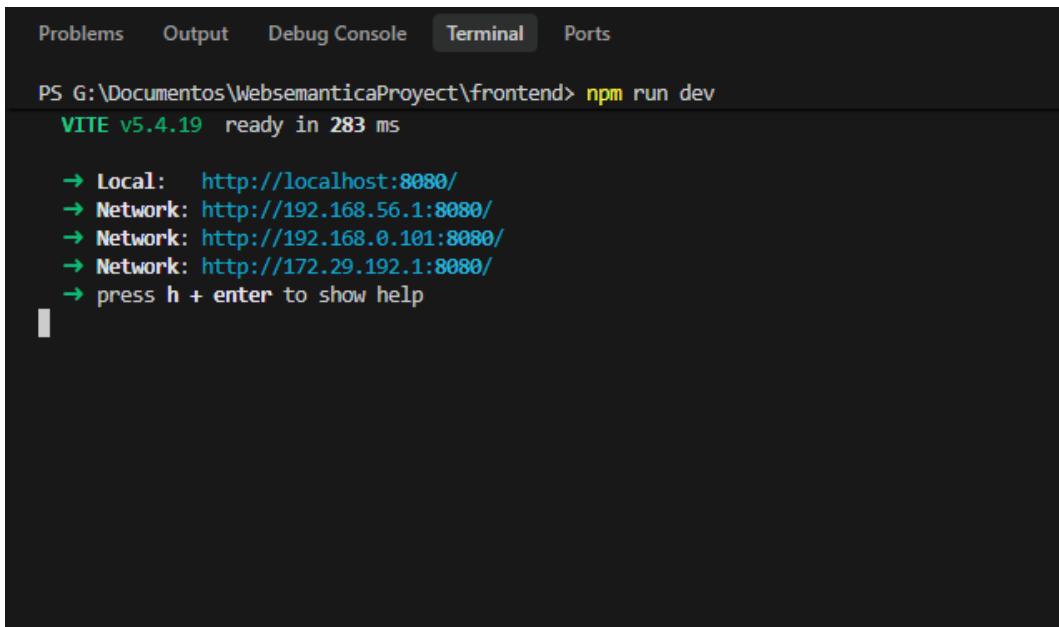
Terminal 2 - Frontend:

```
cd frontend
npm run dev
```

Logs esperados:

```
VITE v5.4.19 ready in 500 ms
```

```
-> Local: http://localhost:8080/
```



A screenshot of a terminal window with a dark background. At the top, there are tabs: 'Problems', 'Output', 'Debug Console', 'Terminal' (which is highlighted in red), and 'Ports'. Below the tabs, the terminal shows the command: 'PS G:\Documentos\WebsemanticaProyecto\frontend> npm run dev'. It then displays the output of the Vite command: 'VITE v5.4.19 ready in 283 ms'. The output includes five entries under the heading '→': 'Local: http://localhost:8080/' (in green), followed by four 'Network:' entries (also in green) with different IP addresses: 'http://192.168.56.1:8080/', 'http://192.168.0.101:8080/', 'http://172.29.192.1:8080/', and 'press h + enter to show help' (in green). There is a small white vertical bar at the bottom left of the terminal window.

Figura 19: Terminal del frontend con Vite ejecutándose

10.2. Modo Producción

Backend:

```
cd backend
uvicorn main:app --host 0.0.0.0 --port 5000 --workers 4
```

Frontend:

```
cd frontend
npm run build
npm run preview
```

10.3. Puertos Utilizados

Servicio	Puerto	URL
Backend API	5000	http://localhost:5000
Frontend Dev	5173	http://localhost:5173
Swagger Docs	5000	http://localhost:5000/docs

Cuadro 11: Puertos utilizados por el sistema

11. Testing

11.1. Tests del Backend

```
cd backend

# Ejecutar todos los tests
pytest

# Con cobertura
pytest --cov=.

# Tests específicos
pytest test_comparison.py
```

11.2. Tests de la API

```
# Test basico de salud
Invoke-RestMethod -Uri "http://localhost:5000/api/v1/products"

# Test de comparacion
Invoke-RestMethod -Uri "http://localhost:5000/api/v1/compare" -
-Method Post -
-ContentType "application/json" -
-Body '{
    "products": [
        "Laptop_Dell_XPS",
        "Laptop_MSI_Gaming"
    ]
}'
```

11.3. Tests Manuales Recomendados

Test	Comando	Resultado Esperado
Listar productos	GET /api/v1/products	JSON con 60+ productos
Gaming laptops	GET /api/v1/swrl/gaming-laptops	3+ laptops con RAM \geq 16GB
Comparación	POST /api/v1/compare	Ganador con score
Validación	GET /api/v1/validate/all	Resumen de validación

Cuadro 12: Tests manuales recomendados

```

● (venv) PS G:\Documentos\WebsemanticaProyect\backend> pytest test_comparison.py
===== test session starts =====
platform win32 -- Python 3.13.3, pytest-7.4.3, pluggy-1.6.0
rootdir: G:\Documentos\WebsemanticaProyect\backend
plugins: anyio-4.12.0, cov-7.0.0
collected 5 items

test_comparison.py ......

===== 5 passed in 0.09s =====

● (venv) PS G:\Documentos\WebsemanticaProyect\backend> pytest
===== test session starts =====
platform win32 -- Python 3.13.3, pytest-7.4.3, pluggy-1.6.0
rootdir: G:\Documentos\WebsemanticaProyect\backend
plugins: anyio-4.12.0, cov-7.0.0
collected 8 items

test_classifier.py .
test_comparison.py .....
test_equivalences.py .
test_market_analysis.py .

===== 8 passed in 1.38s =====

● (venv) PS G:\Documentos\WebsemanticaProyect\backend> pytest --cov=
===== test session starts =====
platform win32 -- Python 3.13.3, pytest-7.4.3, pluggy-1.6.0
rootdir: G:\Documentos\WebsemanticaProyect\backend
plugins: anyio-4.12.0, cov-7.0.0
collected 8 items

test_classifier.py .
test_comparison.py .....
test_equivalences.py .
test_market_analysis.py .

===== tests coverage =====
coverage: platform win32, python 3.13.3-final-0



| Name                       | Stmts | Miss | Cover |
|----------------------------|-------|------|-------|
| __init__.py                | 0     | 0    | 100%  |
| api\__init__.py            | 0     | 0    | 100%  |
| api\middlewares.py         | 0     | 0    | 100%  |
| api\routes\__init__.py     | 0     | 0    | 100%  |
| api\routes\compare.py      | 0     | 0    | 100%  |
| api\routes\products.py     | 40    | 40   | 0%    |
| api\routes\search.py       | 0     | 0    | 100%  |
| api\routes\swr1.py         | 35    | 35   | 0%    |
| api\routes\transactions.py | 0     | 0    | 100%  |


```

Figura 20: Resultados de pytest mostrando todos los tests pasando

12. Mantenimiento y Extensibilidad

12.1. Agregar Nuevos Productos

1. Abrir backend/ontology/SmartCompareMarket.owl en Protégé
2. Tab “Individuals” → Click “+” para agregar individuo
3. Seleccionar clase (ej: Laptop)
4. Agregar propiedades de datos:
 - tienePrecio
 - tieneRAM_GB
 - tieneAlmacenamiento_GB
5. Guardar archivo
6. Reiniciar backend

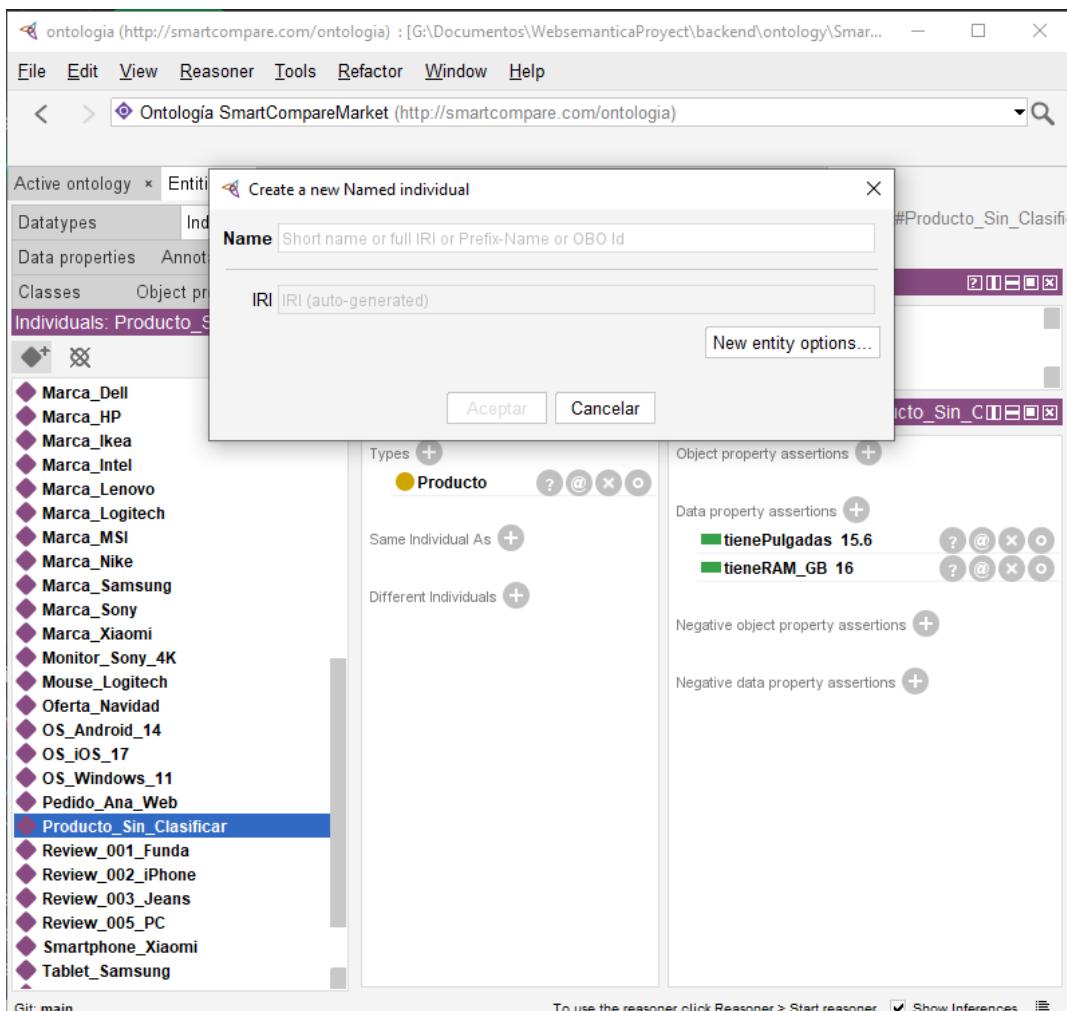


Figura 21: Protégé mostrando cómo agregar un nuevo individuo

12.2. Agregar Nuevas Reglas SWRL

1. En Protégé → Tab “SWRL”
2. Click “+” para nueva regla
3. Escribir regla en formato SWRL
4. Guardar
5. Reiniciar backend

Ejemplo de nueva regla:

```
Smartphone(?s) AND tieneRAM_GB(?s, ?ram) AND greaterThan(?ram, 8)
-> SmartphoneGamaAlta(?s)
```

12.3. Modificar Pesos de Comparación

Editar archivo backend/data/comparison_weights.json:

```
{
  "battery": 0.20,
  "rating": 0.18,
  "price": 0.14,
  "resolution": 0.10,
  "ram": 0.10,
  "storage": 0.10,
  "warranty": 0.07,
  "screen": 0.06,
  "weight": 0.05
}
```

13. Resumen de Archivos Importantes

Archivo	Líneas	Descripción
SmartCompareMarket.owl	2,900+	Ontología principal
comparison_service.py	445	Motor de comparación
recommendation_service.py	277	Sistema de recomendaciones
queries.py	267	Consultas SPARQL
validation_service.py	152	Validación de datos
main.py	~200	Punto de entrada
ComparePage.tsx	500+	Página de comparación

Cuadro 13: Archivos principales del proyecto

14. Información del Proyecto

Campo	Información
Universidad	Universidad La Salle
Facultad	Facultad de Ingeniería
Carrera	Carrera Profesional de Ingeniería de Software
Curso	Web Semántica y Ontologías
Docente	Ing. Marco Antonio Camacho Alatrista
Ciclo Académico	2025 - II
Fecha	Diciembre 2025
Proyecto	Proyecto 16 - Nivel 2
Nombre del Sistema	SmartCompareMarket
Versión del Manual	1.0

Cuadro 14: Información académica del proyecto

14.1. Autor

Nombre	Correo Electrónico
Álvaro André Machaca Meléndez	amachacam@ulasalle.edu.pe

Cuadro 15: Autor del proyecto

14.2. Repositorio

- GitHub: https://github.com/AlvaroMachaca0503/Web_Semantica

14.3. Historial de Versiones

Versión	Fecha	Cambios
1.0	Diciembre 2024	Versión inicial del manual técnico

Cuadro 16: Control de versiones del manual