MANUAL DE USUARIO

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. Introducción

1.1. ¿Qué es SmartCompareMarket?

SmartCompareMarket es una plataforma web inteligente para comparar y recibir recomendaciones de productos electrónicos (Laptops, Smartphones, Tablets). A diferencia de los comparadores tradicionales, este sistema utiliza tecnologías de **Web Semántica** e **Inteligencia Artificial** para:

- **Entender** las características de los productos mediante ontologías OWL 2
- **Clasificar automáticamente** productos usando reglas SWRL (ej: detectar si una laptop es "Gamer")
- **Comparar inteligentemente** productos con un motor de scoring multi-factor
- **Recomendar** productos personalizados según las preferencias del usuario
- **Validar** que las especificaciones de productos sean consistentes
- **Buscar** productos usando consultas SPARQL semánticas

1.2. Características Principales

Característica	Descripción
Razonamiento Automático	Clasifica productos automáticamente (LaptopGamer si RAM $\geq$ 16GB)
Comparación Inteligente	Motor que calcula scores basados en 9 factores y reglas SWRL
Consultas SPARQL	Búsqueda semántica con filtros avanzados
Recomendaciones	Sistema basado en perfil de usuario y razonamiento
Validación	Detecta errores e inconsistencias en especificaciones
Relaciones	Detecta compatibilidad, incompatibilidad y equivalencias

Cuadro 1: Características principales del sistema

1.3. ¿Para quién es este manual?

Este manual está diseñado para **usuarios finales** que desean:

- Instalar y ejecutar el sistema en su computadora
- Usar la plataforma para comparar productos
- Obtener recomendaciones personalizadas

Nota: No se requieren conocimientos técnicos avanzados. Las instrucciones están escritas paso a paso.

2. Requisitos del Sistema

2.1. Requisitos de Hardware

Componente	Mínimo	Recomendado
Procesador	Dual Core 2.0 GHz	Quad Core 2.5+ GHz
Memoria RAM	4 GB	8 GB o más
Espacio en Disco	2 GB libres	5 GB libres
Conexión Internet	Requerida (instalación)	Requerida

Cuadro 2: Requisitos de hardware del sistema

2.2. Requisitos de Software

Antes de comenzar, debes tener instalados los siguientes programas:

Software	Versión	Comando de Verificación
Python	3.11 o superior	`python --version`
Node.js	18.0 o superior	`node --version`
npm	9.0 o superior	`npm --version`
Java JDK	11 o superior	`java -version`
Git (opcional)	Cualquiera	`git --version`

Cuadro 3: Software necesario y comandos de verificación

IMPORTANTE: Java es necesario para el razonador semántico Pellet.

2.3. Navegadores Compatibles

Navegador	Versión	Estado
Google Chrome	90+	Recomendado
Mozilla Firefox	88+	Compatible
Microsoft Edge	90+	Compatible
Safari	14+	Compatible

Cuadro 4: Navegadores compatibles

3. Instalación Paso a Paso

3.1. Paso 1: Verificar Prerrequisitos

Abre una **terminal/consola de comandos** y ejecuta estos comandos uno por uno:

```
python --version
```

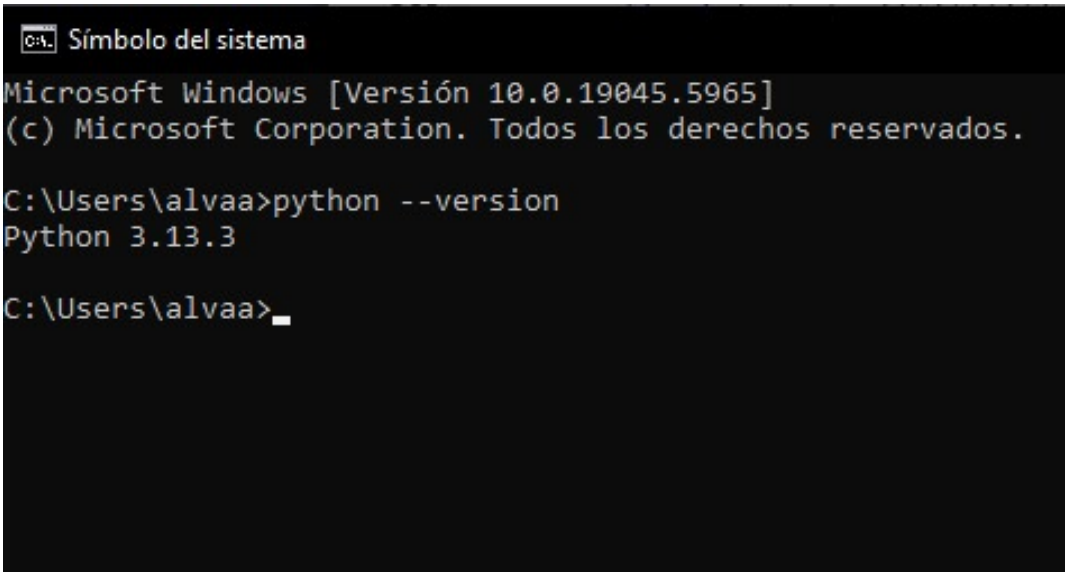


Figura 1: Terminal mostrando Python 3.11.x o superior

```
node --version
```

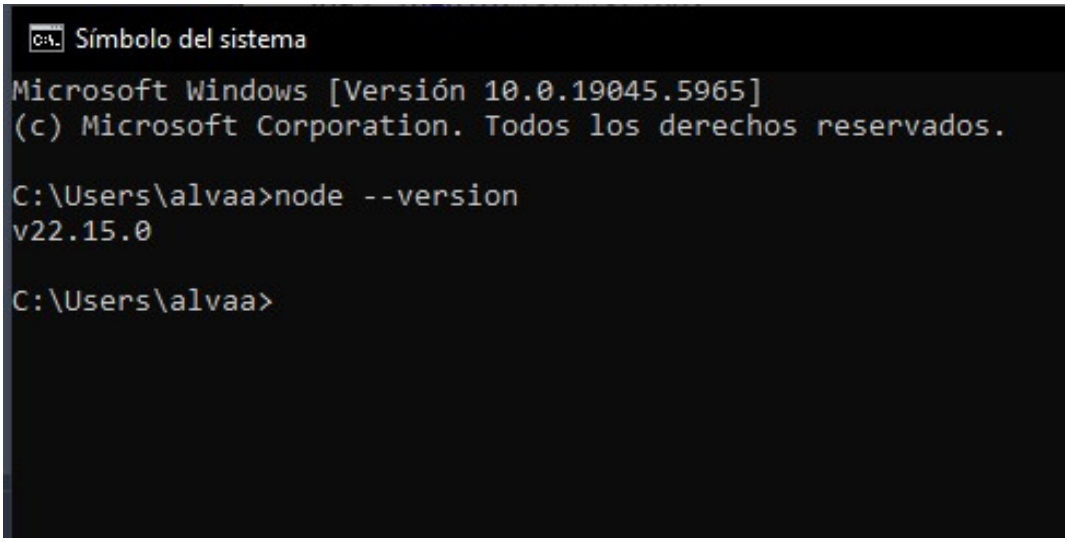


Figura 2: Terminal mostrando v18.x.x o superior

```
npm --version
```

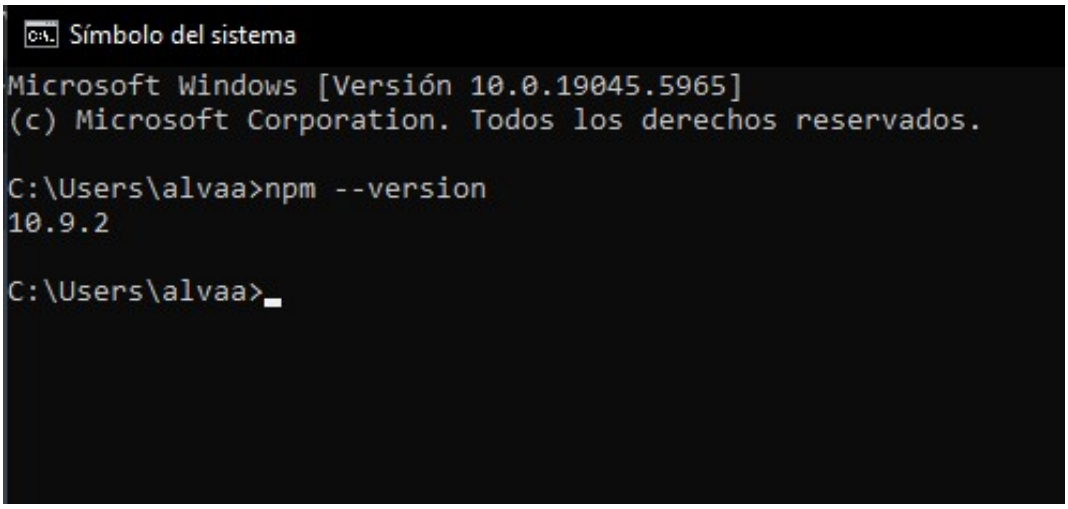


Figura 3: Terminal mostrando 9.x.x o superior

```
java -version
```

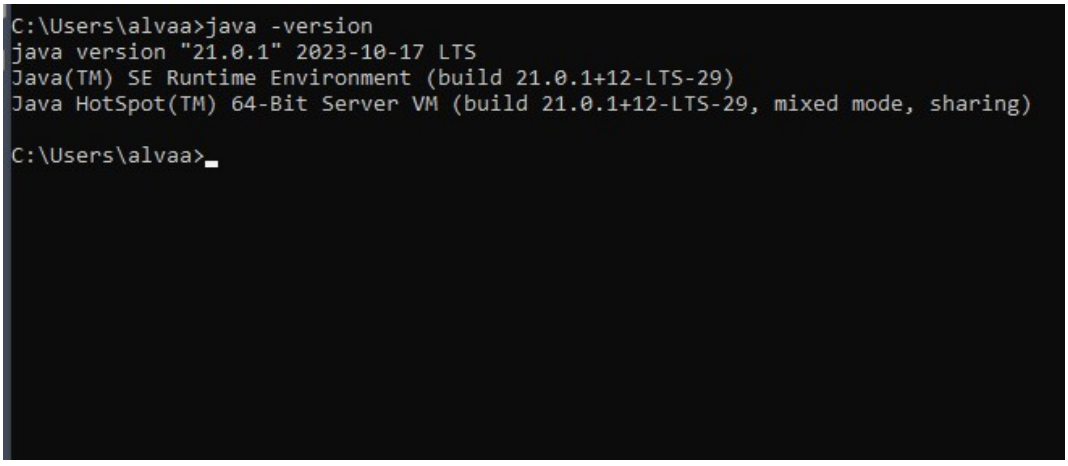


Figura 4: Terminal mostrando openjdk version 11.x.x o similar

3.2. Paso 2: Descargar el Proyecto

3.2.1. Opción A: Usando Git (Recomendado)

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

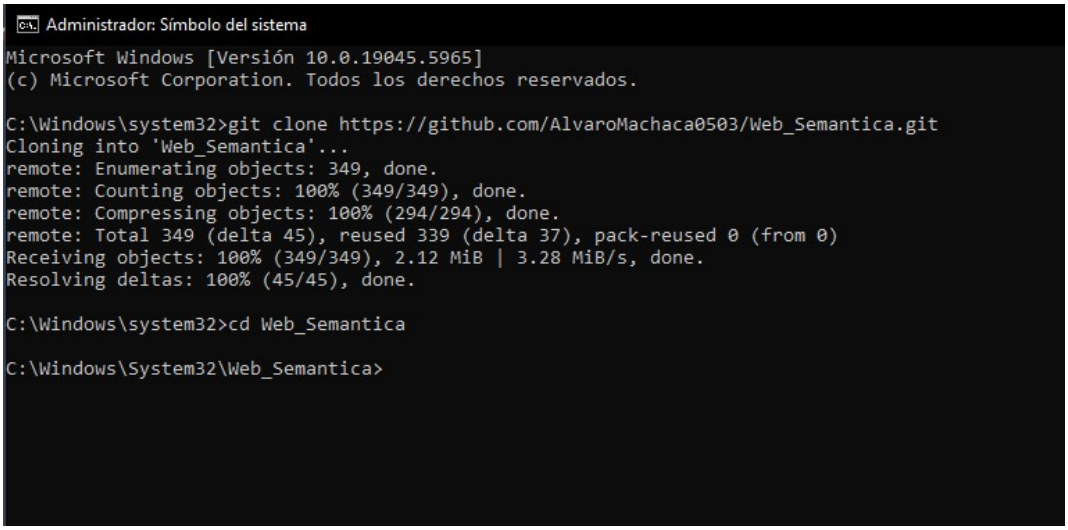
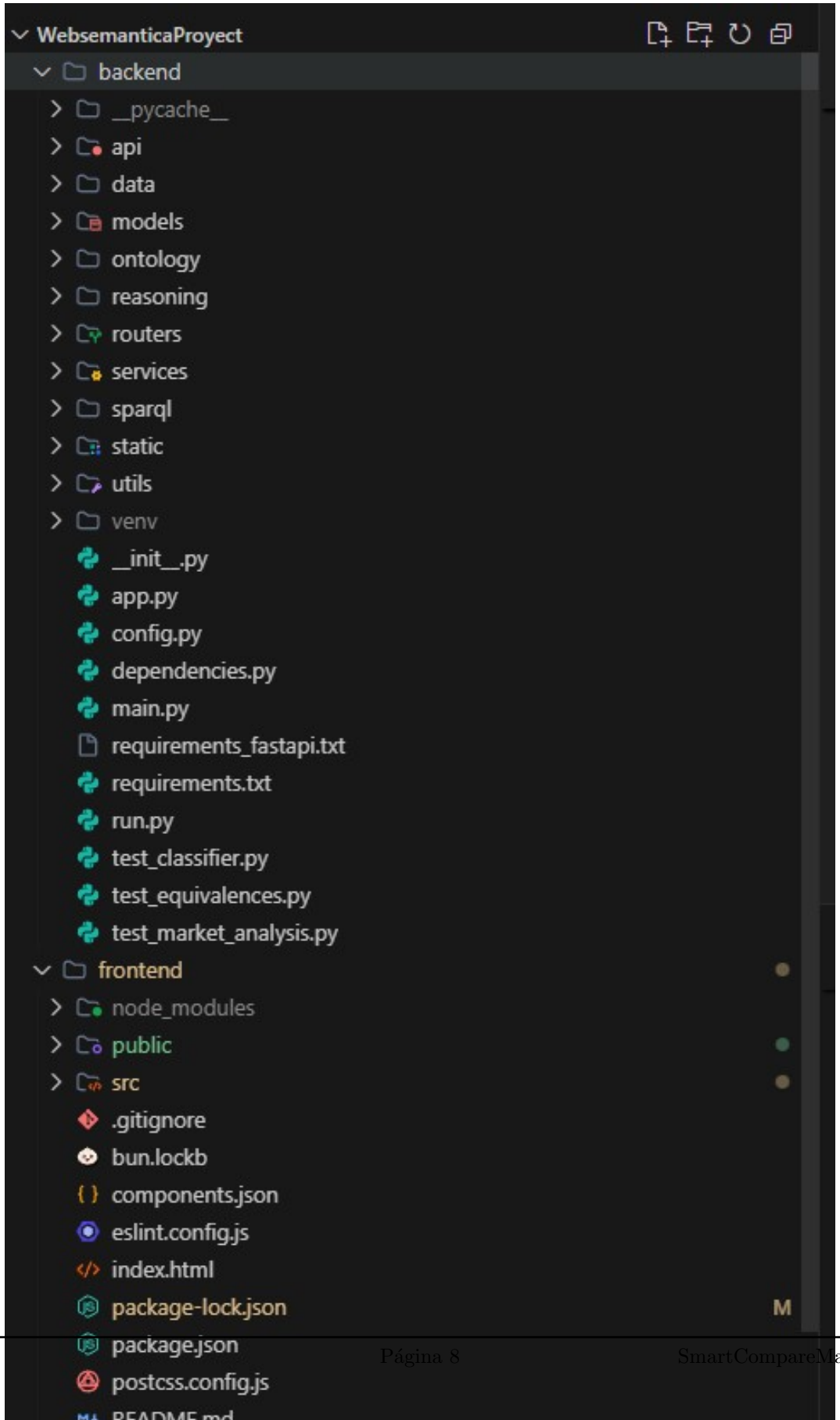


Figura 5: Terminal mostrando el proceso de clonación del repositorio

3.2.2. Opción B: Descarga Manual

1. Ve a la página del repositorio en GitHub
2. Haz clic en el botón verde **"Code"**
3. Selecciona **"Download ZIP"**
4. Extrae el archivo ZIP en una carpeta de tu elección
5. Abre una terminal en esa carpeta


WebsemanticaProyect
backend
__pycache__
api
data
models
ontology
reasoning
routers
services
sparql
static
utils
venv
__init__.py
app.py
config.py
dependencies.py
main.py
requirements_fastapi.txt
requirements.txt
run.py
test_classifier.py
test_equivalences.py
test_market_analysis.py
frontend
node_modules
public
src
.gitignore
bun.lockb
components.json
eslint.config.js
index.html
package-lock.json
M
package.json
postcss.config.js

3.3. Paso 3: Instalar Dependencias del Backend

1. Navega a la carpeta del backend:

```
cd backend
```

2. Crea un entorno virtual de Python:

   En Windows:

```
python -m venv venv
venv\Scripts\activate
```

   En Mac/Linux:

```
python3 -m venv venv
source venv/bin/activate
```

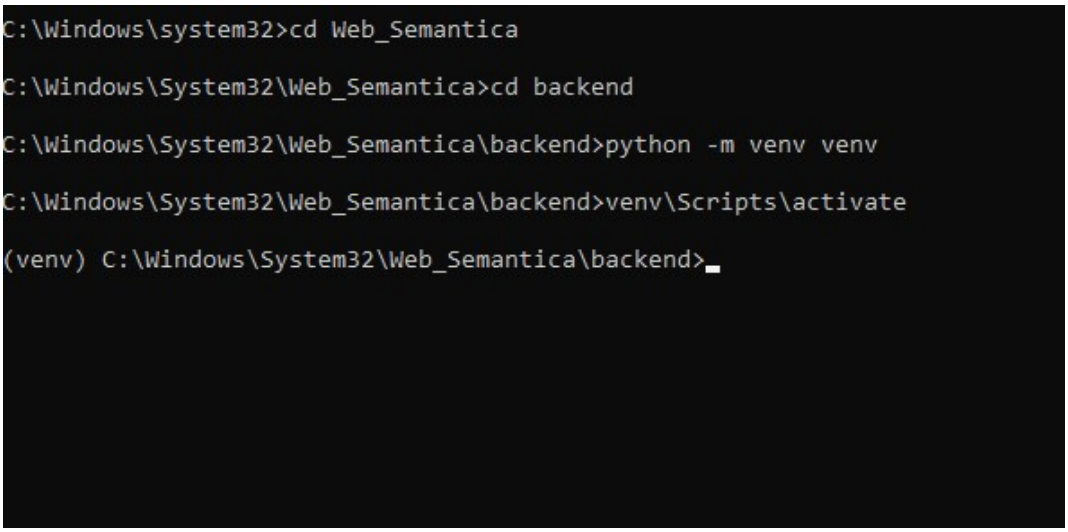


Figura 7: Terminal mostrando (venv) al inicio

3. Instala las dependencias:

```
pip install -r requirements.txt
```

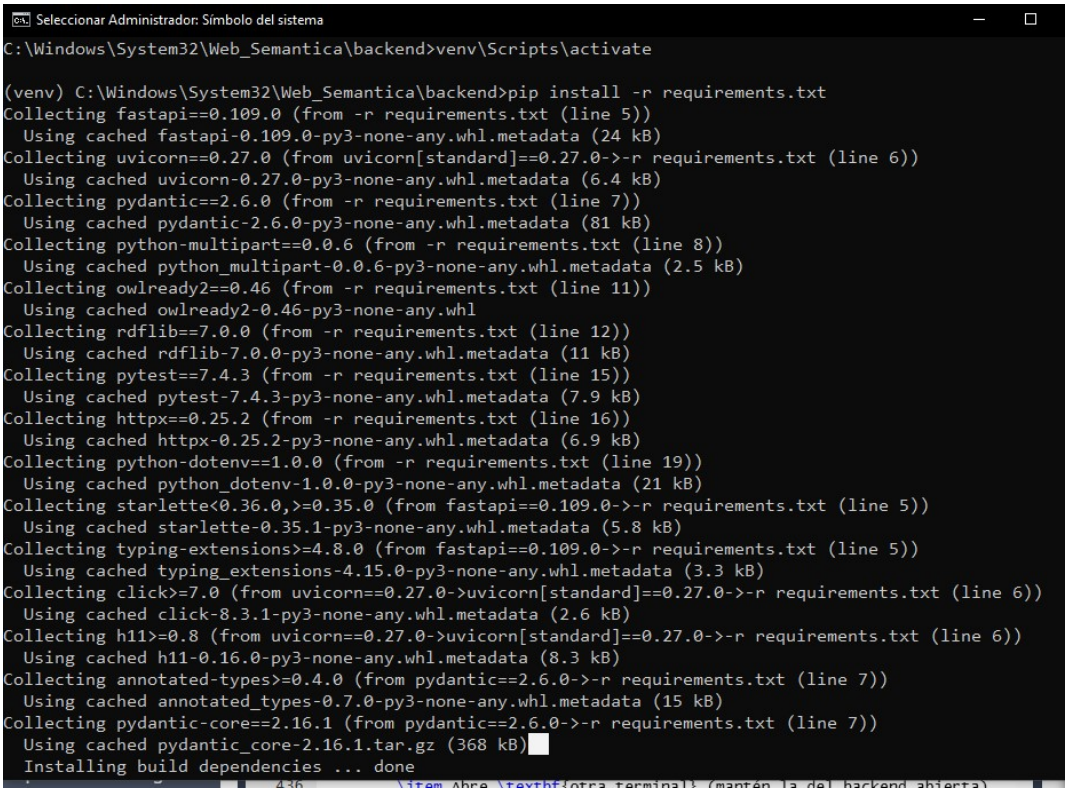


Figura 8: Terminal mostrando dependencias instaladas exitosamente

3.4. Paso 4: Instalar Dependencias del Frontend

1. Abre **otra terminal** (mantén la del backend abierta)
2. Navega a la carpeta del frontend:

```
cd frontend
```

3. Instala las dependencias:

```
npm install
```

Nota: Este proceso puede tardar 2-5 minutos dependiendo de tu conexión.

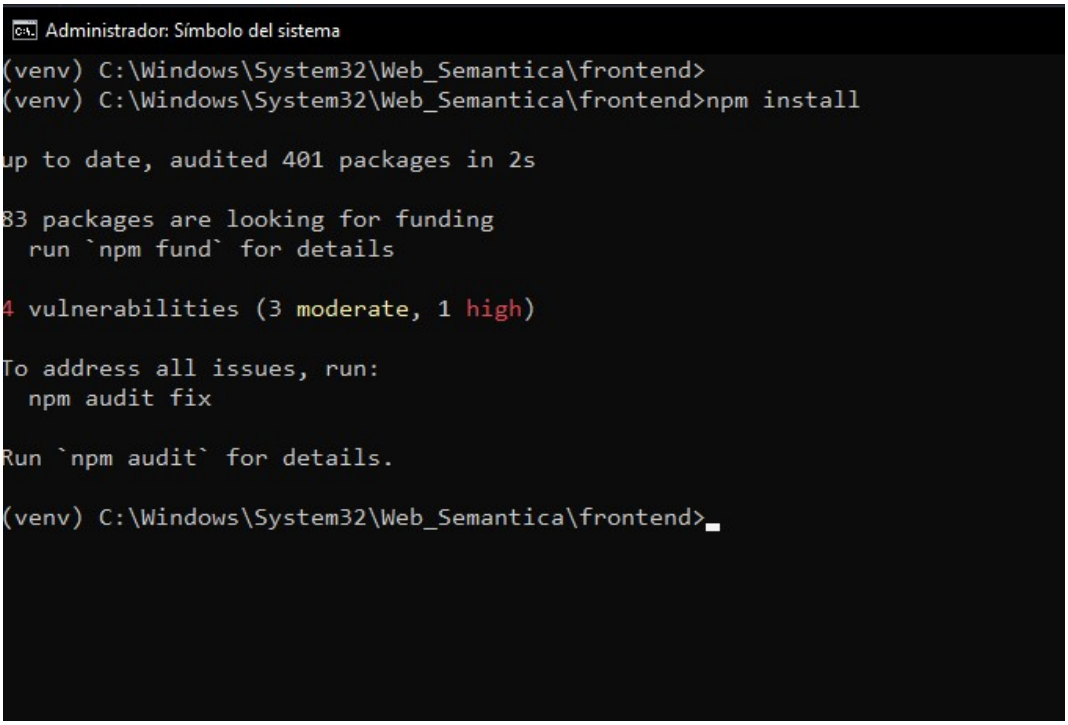


Figura 9: Terminal mostrando paquetes de Node.js instalados

4. Ejecución del Proyecto

4.1. Iniciar el Backend (Servidor API)

1. Asegúrate de estar en la carpeta `backend` con el entorno virtual activado
2. Ejecuta:

```
python main.py
```

3. **Espera** hasta ver estos mensajes de éxito:

```
[OK] Ontologia cargada: 60+ productos
[OK] Razonador Pellet ejecutado exitosamente
[OK] Reglas SWRL aplicadas
INFO:    Uvicorn running on http://0.0.0.0:5000
```

```
PS G:\Documentos\WebsemanticaProyect> cd .\backend\
PS G:\Documentos\WebsemanticaProyect\backend> python main.py
[OK] Ontologia 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 deberian aplicarse:
      1. DetectarGamer (RAM >= 16GB -> LaptopGamer)
      2. EncontrarMejorPrecio (precio menor -> esMejorOpcionQue)
      3. ClasificarPositivas (cal >= 4 -> Resena_Positiva)
      4. ClasificarNegativas (cal <= 2 -> Resena_Negativa)
[OK] Ontologia cargada en RDFlib: 1327 triples
======================================================================
[START] SmartCompareMarket Backend (FastAPI) - Iniciando...
======================================================================

[OK] Servidor corriendo en: http://0.0.0.0:5000
[DOCS] Documentacion Swagger: http://localhost:5000/docs
[DOCS] Documentacion ReDoc: http://localhost:5000/redoc

[API] Endpoints disponibles:
   GET  /api/v1/products
   GET  /api/v1/products/{id}
   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\\WebsemanticaProyect\\backend']
INFO:     Uvicorn running on http://0.0.0.0:5000 (Press CTRL+C to quit)
INFO:     Started reloader process [15520] using StatReload
[OK] Ontologia 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 deberian aplicarse:
      1. DetectarGamer (RAM >= 16GB -> LaptopGamer)
      2. EncontrarMejorPrecio (precio menor -> esMejorOpcionQue)
      3. ClasificarPositivas (cal >= 4 -> Resena_Positiva)
```

Figura 10: Terminal mostrando inicio exitoso del backend

¡NO CIERRES ESTA TERMINAL! El servidor debe permanecer ejecutándose.

4.2. Iniciar el Frontend (Interfaz Web)

1. En la **segunda terminal**, asegúrate de estar en la carpeta `frontend`
2. Ejecuta:

```
npm run dev
```

3. Espera hasta ver:

```
VITE v5.x.x ready in XXX ms

->  Local:   http://localhost:8080/
->  Network: use --host to expose
```

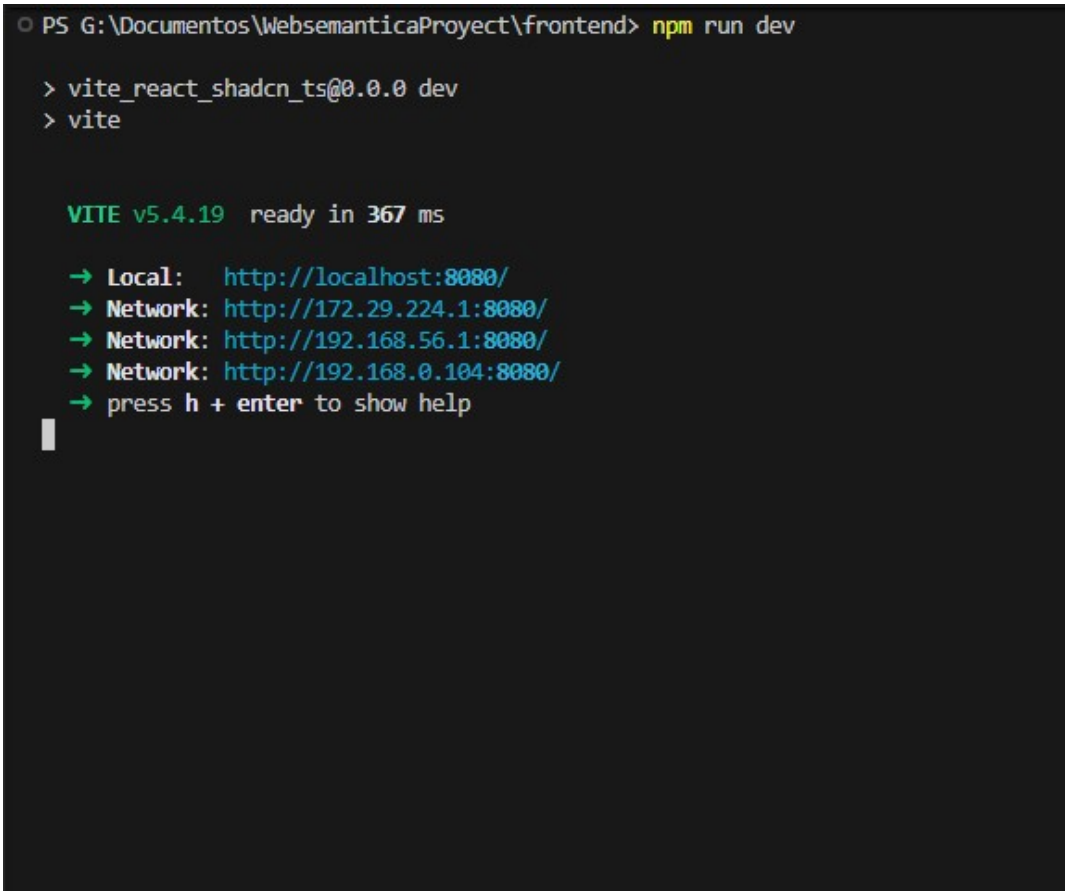


Figura 11: Terminal mostrando Vite ejecutándose

4.3. Abrir la Aplicación

1. Abre tu **navegador web** (Chrome recomendado)
2. Escribe en la barra de direcciones: `http://localhost:8080`
3. Presiona **Enter**

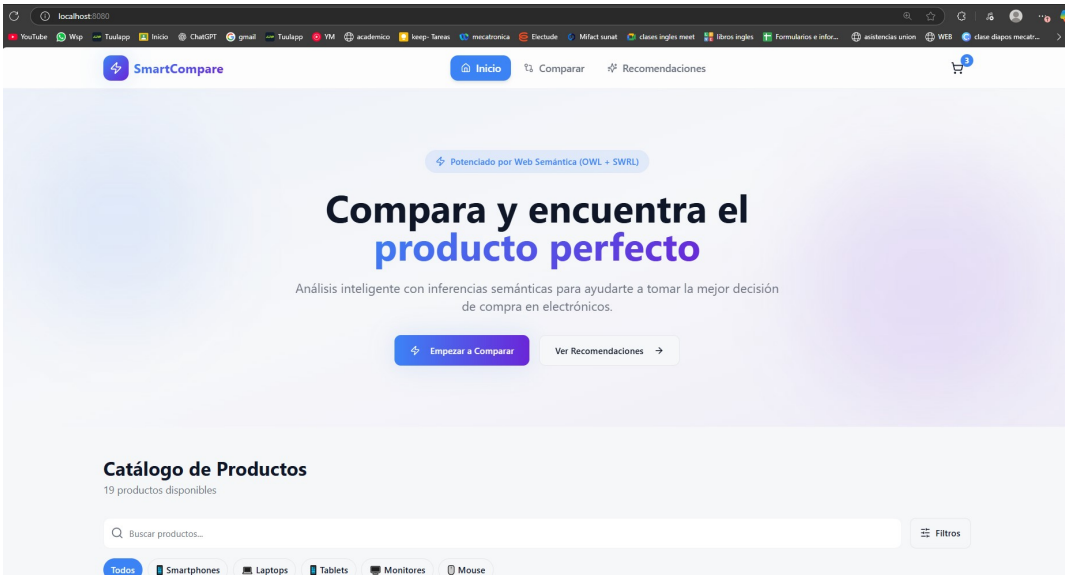


Figura 12: Página principal del sistema mostrando el catálogo de productos

4.4. URLs Importantes

Servicio	URL
Frontend (Interfaz)	http://localhost:8080
Backend (API)	http://localhost:5000
Documentación API (Swagger)	http://localhost:5000/docs

Cuadro 5: URLs principales del sistema

5. Guía de Uso del Sistema

5.1. Navegación Principal

La aplicación tiene **tres secciones principales** accesibles desde el menú superior:

Sección	Icono	Descripción
Inicio		Catálogo completo de productos
Comparar		Comparación inteligente de productos
Recomendaciones		Recomendaciones personalizadas

Cuadro 6: Secciones principales de navegación

Figura 13: Menú de navegación superior mostrando las tres opciones

5.2. Explorar el Catálogo de Productos

En la **página de Inicio**, verás tarjetas de productos. Cada tarjeta muestra:

- Imagen del producto
- Nombre y marca
- Precio (con descuento si aplica)
- Calificación de usuarios
- Especificaciones técnicas (RAM, Almacenamiento, etc.)
- Badges especiales (ej: “Laptop Gamer”)

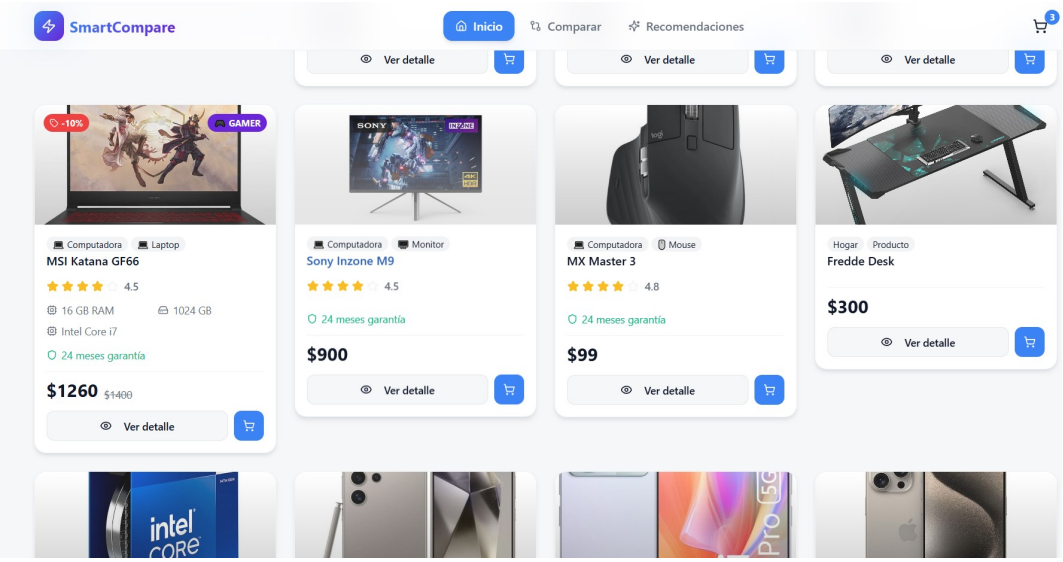


Figura 14: Tarjeta de producto mostrando nombre, precio, especificaciones y badge

5.2.1. Usando los Filtros de Búsqueda

1. Usa la **barra de búsqueda** para buscar por nombre
2. Puedes seleccionar entre **2 y 5 productos**
3. Verás una **barra flotante** en la parte inferior indicando cuántos has seleccionado

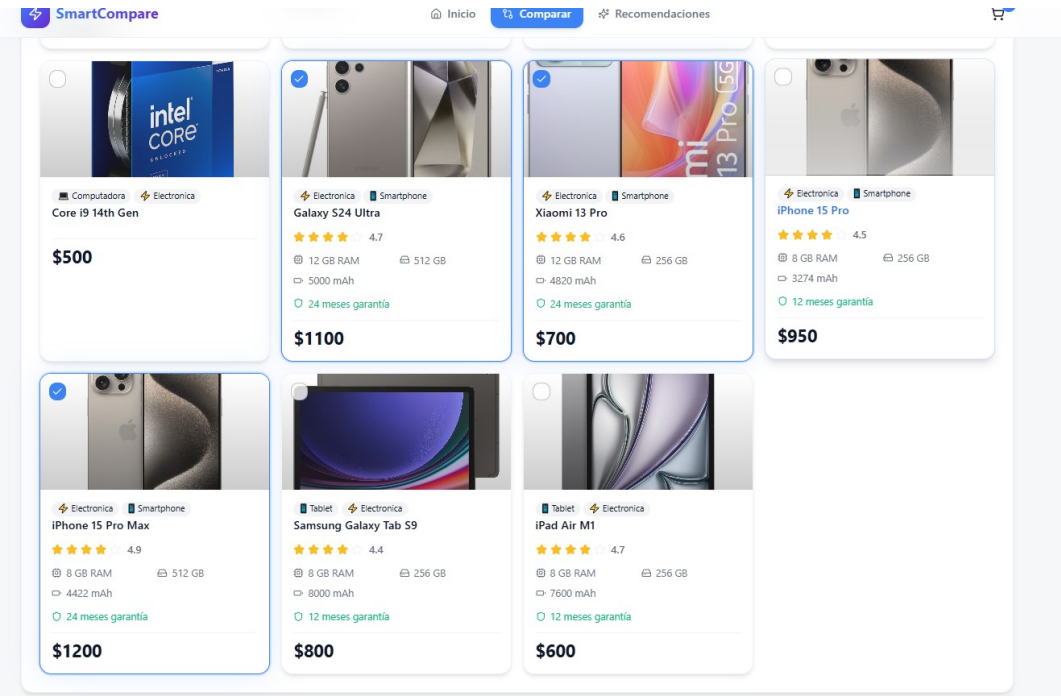


Figura 15: Tres tarjetas de productos con una seleccionada

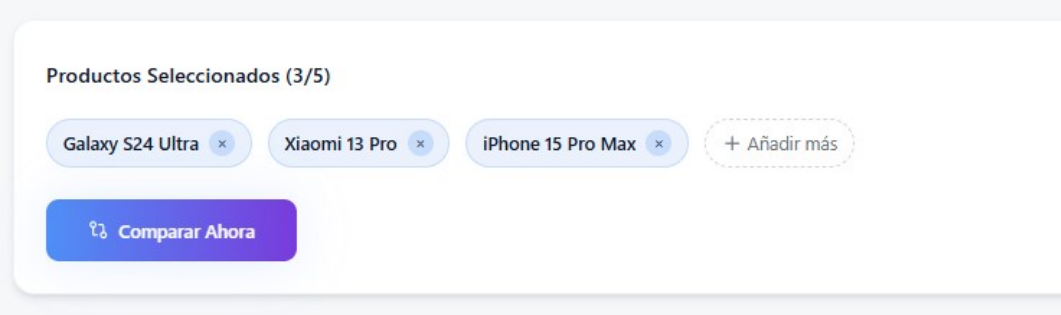


Figura 16: Barra flotante inferior de comparación

5.2.2. Paso 2: Ir a la página de comparación

Haz clic en el botón **"Comparar"** de la barra flotante.

Propiedad	iPhone 15 Pro Max Score: 80.1	Xiaomi 13 Pro Score: 85.0	Galaxy S24 Ultra Score: 86.4
Nombre	iPhone 15 Pro Max	Xiaomi 13 Pro	Galaxy S24 Ultra
$ Precio	$1,200	$700	$1,100
RAM	8 GB	12 GB	12 GB
Almacenamiento	512 GB	256 GB	512 GB
Calificación	☆ 4.9	☆ 4.6	☆ 4.7
Pantalla	6.7"	6.7"	6.8"
Batería	4422 mAh	4820 mAh	5000 mAh
Garantía	24 meses	24 meses	24 meses
Marca	Marca_Apple	Marca_Xiaomi	Marca_Samsung
Vendedor	Vend_Oficial_iShop	Vend_Amazon	Vend_Amazon
Sistema Operativo	OS_iOS_17	OS_Android_14	OS_Android_14
esCompatibleCon	N/A	Cargador_USB_C	Cargador_USB_C
esSimilarA	iPhone15_Barato	N/A	N/A
imagenUrl	/images/products/iphone15_caro.jpg	/images/products/xiaomi_13_pro.jpg	/images/products/galaxy_s24_ultra.jpg
incompatibleCon	N/A	N/A	iPhone15_Barato
Peso	221g	229g	232g
Resolución	2796x1290	3200x1440	3088x1440

Figura 17: Página de comparación mostrando la tabla comparativa

5.2.3. Paso 3: Analizar los resultados

La página de comparación muestra:
A. Ganador Global (parte superior)

- El sistema determina automáticamente el **mejor producto**

- Muestra el **score numérico** (0-100 puntos)
- Explica la **razón** de por qué ganó

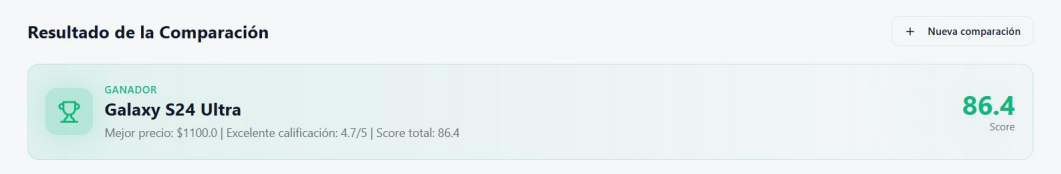


Figura 18: Sección del Ganador con nombre, score y razón

B. Tabla Comparativa (centro)

- Cada columna es un producto
- Cada fila es una característica (Precio, RAM, Batería, etc.)
- Los valores **en verde** son los mejores de cada fila
- Los valores **en amarillo** indican empate

Propiedad	Xiaomi 13 Pro Score: 85.0	Galaxy S24 Ultra Score: 86.4
Nombre	Xiaomi 13 Pro	Galaxy S24 Ultra
$ Precio	$700	$1,100
RAM	12 GB	12 GB
Almacenamiento	256 GB	512 GB
Calificación	4.6	4.7
Pantalla	6.7"	6.8"
Batería	4820 mAh	5000 mAh
Garantía	24 meses	24 meses
Marca	Marca_Xiaomi	Marca_Samsung
Vendedor	Vend_Amazon	Vend_Amazon
Sistema Operativo	OS_Android_14	OS_Android_14
esCompatibleCon	Cargador_USB_C	Cargador_USB_C
imagenUrl	/images/products/xiaomi_13_pro.jpg	/images/products/galaxy_s24_ultra.jpg
incompatibleCon	N/A	iPhone15_Barato
Peso	229g	232g
Resolución	3200x1440	3088x1440

Figura 19: Tabla comparativa con celdas resaltadas

C. Inferencias SWRL (parte inferior)

- Muestra las **reglas inteligentes** aplicadas
- Ejemplos: “ LaptopGamer detectado”, “ Es mejor opción que ProductoX”

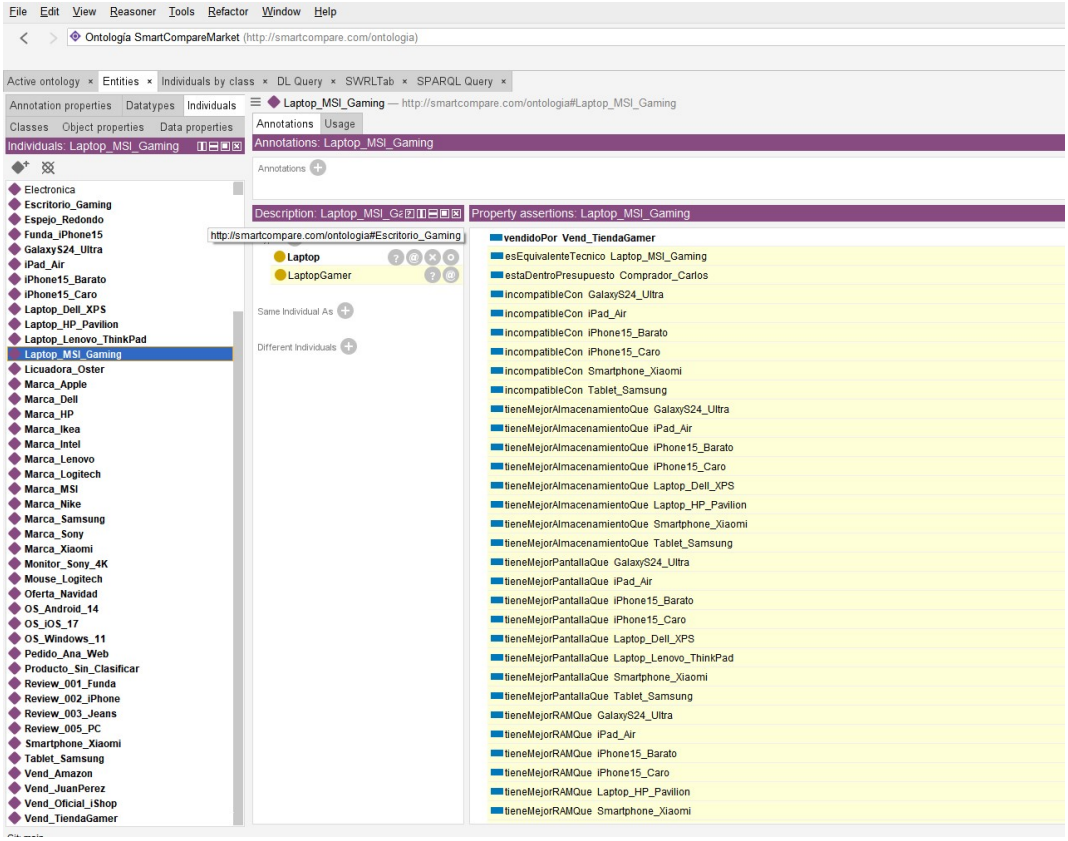


Figura 20: Sección Reglas SWRL Aplicadas mostrando inferencias

5.3. Obtener Recomendaciones Personalizadas

5.3.1. Paso 1: Ir a Recomendaciones

Haz clic en **"Recomendaciones"** en el menú superior.

5.3.2. Paso 2: Configurar preferencias

En el **panel izquierdo**, ajusta:

Preferencia	Descripción	Ejemplo
Presupuesto máximo	Cuánto puedes gastar	$1500
Categoría preferida	Tipo de producto	Laptop
RAM mínima	Memoria RAM mínima	16 GB
Almacenamiento mínimo	Disco duro mínimo	512 GB
Calificación mínima	Puntuación de usuarios	4.0

Cuadro 7: Opciones de preferencias del usuario

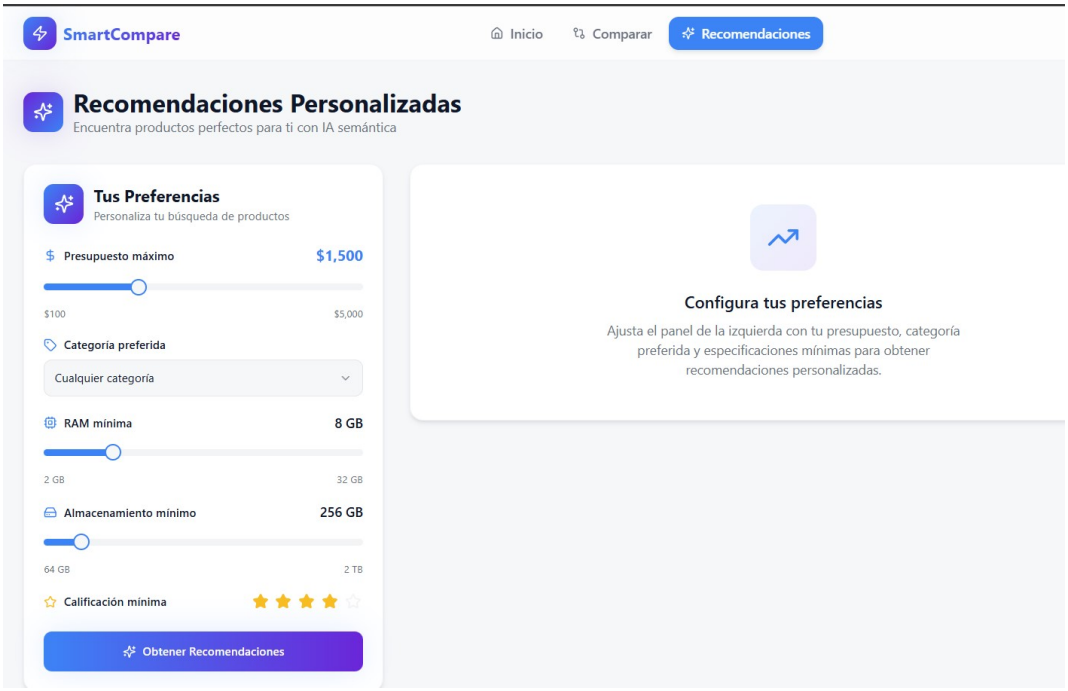


Figura 21: Panel de preferencias con sliders y selectores configurados

5.3.3. Paso 3: Ver las recomendaciones

El **panel derecho** mostrará:

1. Lista de productos ordenados por **relevancia**
2. Para cada producto:
   - **Score de match** (0-100 %)

- **Razón** de la recomendación
- Especificaciones principales

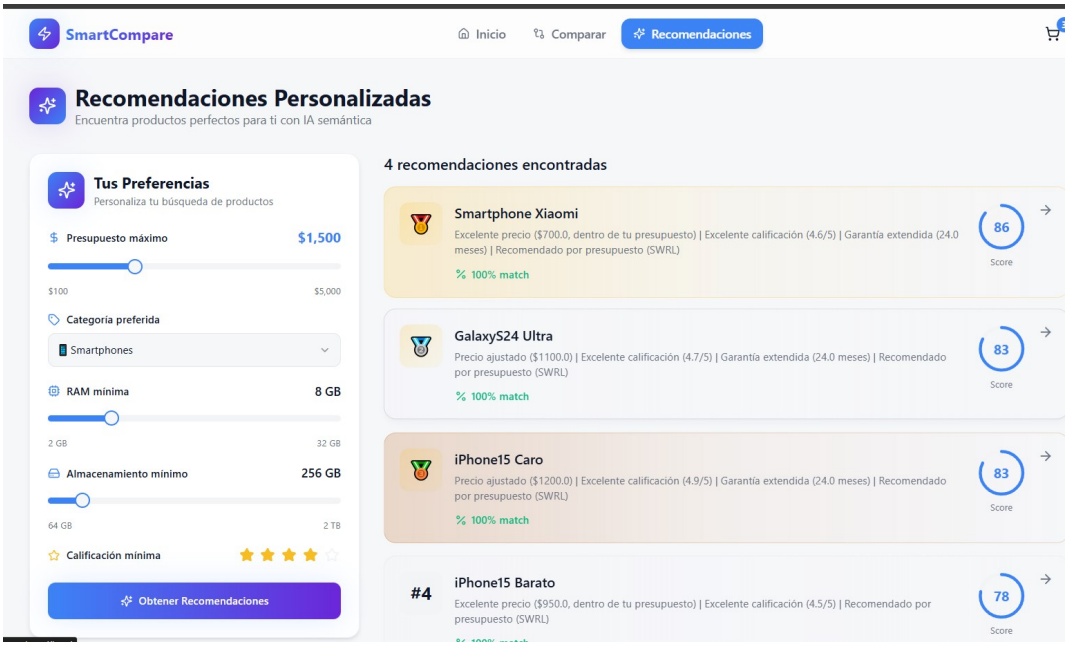


Figura 22: Lista de recomendaciones con scores y razones

6. Sistema de Comparación Inteligente

6.1. Algoritmo de Scoring

El motor de comparación evalúa productos usando **9 factores ponderados**:

Factor	Peso	Criterio
Batería	20 %	Mayor es mejor
Calificación	18 %	Mayor es mejor
Precio	14 %	**Menor es mejor**
Resolución	10 %	Mayor es mejor
RAM	10 %	Mayor es mejor
Almacenamiento	10 %	Mayor es mejor
Garantía	7 %	Mayor es mejor
Pantalla	6 %	Mayor es mejor
Peso	5 %	**Menor es mejor**

Cuadro 8: Factores de scoring y sus pesos

Bonus por reglas SWRL:

- +2 puntos si el producto "es mejor opción que" otro
- +10 puntos si es detectado como "Laptop Gamer"

6.2. Clasificación Automática (SWRL)

El sistema clasifica productos automáticamente usando reglas inteligentes:

Regla	Condición	Clasificación
DetectarGamer	Laptop con RAM $\geq$ 16GB	$\rightarrow$ LaptopGamer
EncontrarMejorPrecio	Mismo producto, menor precio	$\rightarrow$ esMejorOpcionQue
ClasificarPositivas	Reseña con calificación ≥ 4	$\rightarrow$ Reseña_Positiva
ClasificarNegativas	Reseña con calificación ≤ 2	$\rightarrow$ Reseña_Negativa

Cuadro 9: Reglas SWRL implementadas

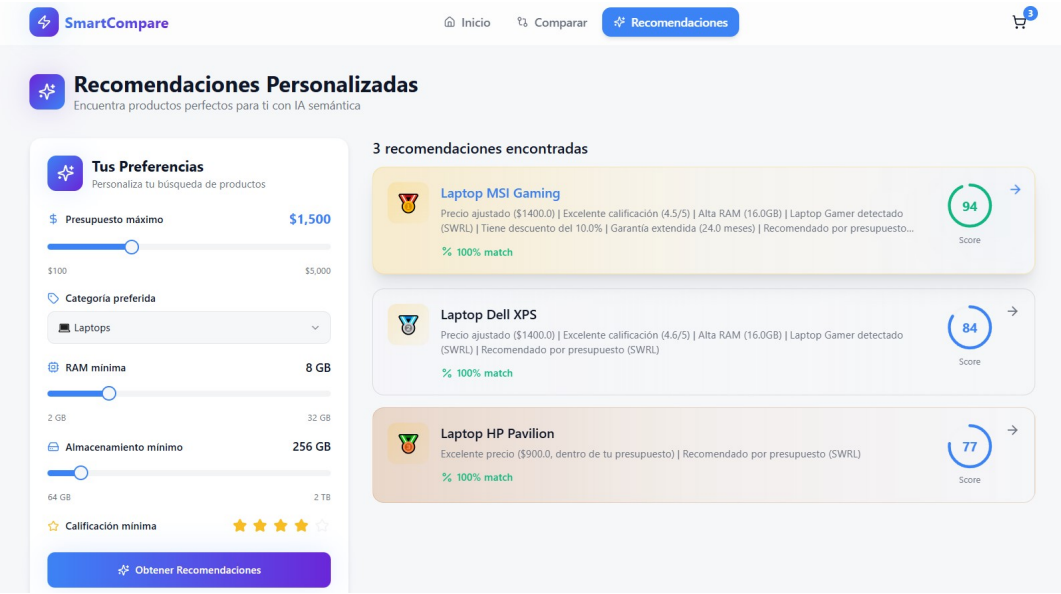


Figura 23: Producto mostrando el badge Laptop Gamer clasificado automáticamente

7. Solución de Problemas Comunes

7.1. Error: "python no se reconoce como comando"

Problema: Python no está instalado o no está en el PATH.
Solución:

1. Descarga Python desde: `https://www.python.org/downloads/`
2. Durante la instalación, **marca la casilla "Add Python to PATH"**
3. Reinicia la terminal

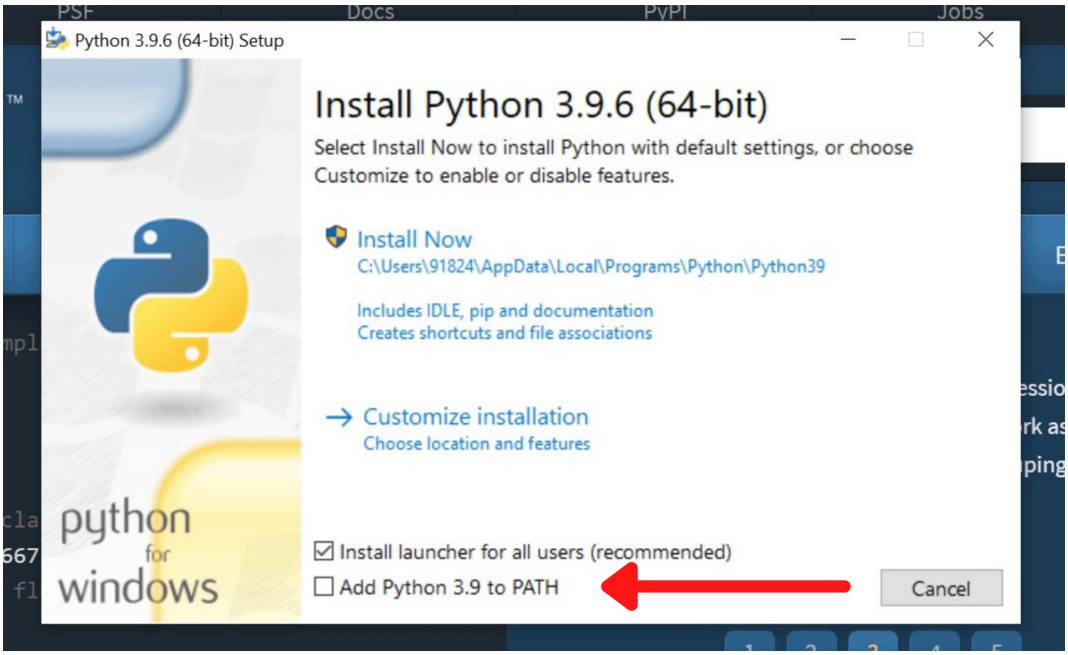


Figura 24: Instalador de Python con casilla Add Python to PATH marcada

7.2. Error: "npm no se reconoce como comando"

Problema: Node.js no está instalado.
Solución:

1. Descarga Node.js desde: `https://nodejs.org/`
2. Elige la versión **LTS** (recomendada)
3. Instala y reinicia la terminal

7.3. Error: "No se puede conectar al servidor" en el frontend

Problema: El backend no está ejecutándose.
Solución:

1. Verifica que la terminal del backend muestre:

```
INFO: Uvicorn running on http://0.0.0.0:5000
```

2. Si no, vuelve a ejecutar `python main.py`

```
PS G:\Documentos\WebsemanticaProyect\backend> python main.py
   GET  /api/v1/products/{id}
   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\\WebsemanticaProyect\\backend']
INFO:     Uvicorn running on http://0.0.0.0:5000 (Press CTRL+C to quit)
INFO:     Started reloader process [15520] using StatReload
[OK] Ontologia 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 deberian aplicarse:
      1. DetectarGamer (RAM >= 16GB -> LaptopGamer)
      2. EncontrarMejorPrecio (precio menor -> esMejorOpcionQue)
      3. ClasificarPositivas (cal >= 4 -> Resena_Positiva)
      4. ClasificarNegativas (cal <= 2 -> Resena_Negativa)
[OK] Ontologia cargada en RDFlib: 1327 triples
INFO:     Started server process [3596]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:37708 - "GET /api/v1/products HTTP/1.1" 200 OK
INFO:     127.0.0.1:37710 - "GET /api/v1/products HTTP/1.1" 200 OK
INFO:     127.0.0.1:37740 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:37740 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:37740 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:37740 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:37740 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:37797 - "GET /api/v1/products HTTP/1.1" 200 OK
INFO:     127.0.0.1:37797 - "GET /api/v1/products/Producto_Sin_Clasificar HTTP/1.1" 200 OK
INFO:     127.0.0.1:37797 - "GET /api/v1/products HTTP/1.1" 200 OK
INFO:     127.0.0.1:37798 - "GET /api/v1/products HTTP/1.1" 200 OK
INFO:     127.0.0.1:37816 - "OPTIONS /api/v1/compare HTTP/1.1" 200 OK
INFO:     127.0.0.1:37816 - "POST /api/v1/compare HTTP/1.1" 200 OK
INFO:     127.0.0.1:37844 - "POST /api/v1/compare HTTP/1.1" 200 OK
INFO:     127.0.0.1:37862 - "POST /api/v1/compare HTTP/1.1" 200 OK
INFO:     127.0.0.1:26504 - "OPTIONS /api/v1/recommendations?limit=5 HTTP/1.1" 200 OK
INFO:     127.0.0.1:26504 - "POST /api/v1/recommendations?limit=5 HTTP/1.1" 200 OK
INFO:     127.0.0.1:26505 - "POST /api/v1/recommendations?limit=5 HTTP/1.1" 200 OK
```

Figura 25: Terminal del backend ejecutándose correctamente

7.4. Error: “Error loading ontology”

Problema: Java no está instalado (necesario para Pellet).

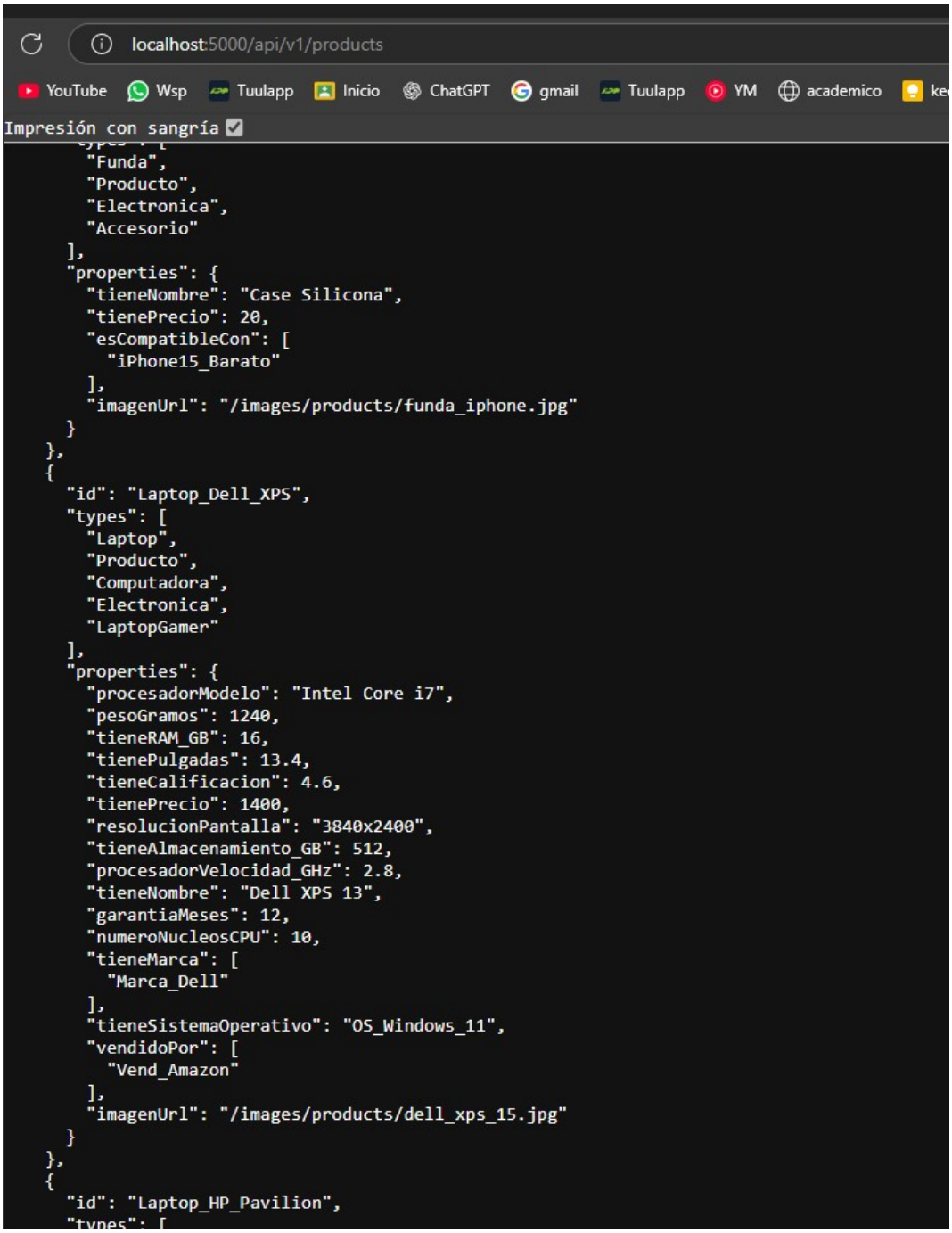


Figura 26: Navegador mostrando JSON de productos

8. Preguntas Frecuentes (FAQ)

8.1. ¿Qué significa "Inferencia SWRL"?

Son **reglas lógicas programadas** en la ontología. Por ejemplo: *"Si una Laptop tiene RAM ≥ 16GB, entonces se clasifica como LaptopGamer"*. El sistema aplica estas reglas automáticamente.

8.2. ¿Cómo se decide el "Ganador" en una comparación?

El sistema calcula un **score de 0 a 100** considerando:

1. Los 9 factores técnicos (ver Tabla 6)
2. Bonus por reglas SWRL aplicadas
3. Relación calidad-precio

El producto con el score más alto gana.

8.3. ¿Por qué algunos productos tienen el badge "Laptop Gamer"?

El sistema detectó automáticamente que tienen **RAM de 16GB o más**, lo cual es típico de laptops para gaming según las reglas SWRL.

8.4. ¿Necesito Java si el sistema es Python?

Sí. El razonador **Pellet** que ejecuta las reglas SWRL está escrito en Java. La librería Owlready2 lo necesita para el razonamiento semántico.

8.5. ¿Necesito Internet para usar el sistema?

- **Para instalar:** Sí, para descargar dependencias
- **Para usar:** No, funciona localmente una vez instalado

9. Conclusiones

Este manual ha cubierto todos los aspectos necesarios para instalar, configurar, ejecutar y utilizar **SmartCompareMarket**. Los principales logros del proyecto incluyen:

- **Ontología OWL 2 compleja:** 48 clases, 30+ propiedades, 60+ individuos
- **Reglas SWRL funcionales:** Clasificación automática de productos
- **Motor de comparación inteligente:** Scoring multi-factor + inferencias
- **Búsqueda semántica:** Consultas SPARQL con filtros avanzados
- **Recomendaciones personalizadas:** Basadas en perfil y razonamiento
- **Validación de consistencia:** Detección de errores en especificaciones
- **Frontend moderno:** React + TypeScript + Tailwind CSS
- **Backend robusto:** FastAPI + Pellet + Owlready2

9.1. Tecnologías Utilizadas

Componente	Tecnología
Backend	Python 3.11 + FastAPI + Uvicorn
Ontología	OWL 2 + SWRL + Pellet
Consultas	SPARQL + RDFlib
Frontend	React 18 + TypeScript + Vite
Estilos	Tailwind CSS + Shadcn/UI

Cuadro 10: Stack tecnológico del proyecto

Gracias por utilizar SmartCompareMarket.
¡Éxito en tu experiencia de comparación inteligente!

10. 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 11: Información académica del proyecto

10.1. Autor

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

Cuadro 12: Autor del proyecto

10.2. Repositorio

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

10.3. Historial de Versiones

Versión	Fecha	Cambios
1.0	Diciembre 2025	Versión inicial del manual de usuario

Cuadro 13: Control de versiones del manual