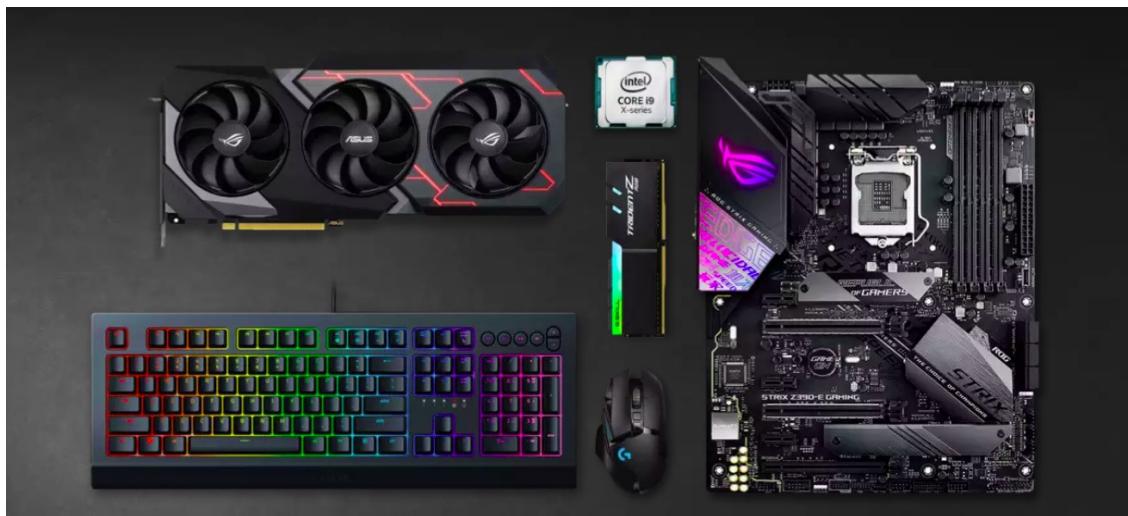


PROYECTO DE TEORÍA: CONFIGURADOR DE PC

Bases de Dades

2022-23



Pablo Arranz

Adrián Galán Pacheco

Pablo Noriega Vázquez

Ronald Ortiz de Lazcano

ÍNDICE

1. INTRODUCCIÓN	2
2. OBJETIVOS	3
3. ESQUEMA E-R	4
4. RECOPILACIÓN DE DATOS	9
5. CREACIÓN DE LA BASE DE DATOS	12
6. OBTENCIÓN DEL ORDENADOR	16
7. COMPROBACIÓN DEL RESULTADO OBTENIDO	20
8. CONCLUSIONES	27
9. ANEXO	28
9.1. ARCHIVO DE SCRAPING	28
9.2. ARCHIVO DE QUERY MYSQL	44

1. INTRODUCCIÓN

Hoy en día, el acceso a un ordenador se ha vuelto esencial para la vida cotidiana de cualquier persona, ya que la tecnología ha llegado a prácticamente todos los aspectos de nuestra vida, tanto en el ámbito personal como en el laboral y académico. Desde realizar trámites bancarios y compras en línea, hasta buscar información, comunicarse con otras personas, trabajar desde casa o estudiar en línea, la necesidad de tener acceso a un ordenador se ha vuelto imprescindible.

Además, tras la pandemia se ha acelerado esta necesidad al obligar a muchas personas a trabajar o estudiar telemáticamente, lo que ha aumentado la demanda de equipos informáticos y servicios en línea.

En las últimas décadas, los ordenadores han experimentado una evolución impresionante tanto en su hardware como en su software. El aumento constante de las prestaciones que nos pueden ofrecer los diferentes componentes ha permitido la creación de ordenadores cada vez más potentes, capaces de manejar cargas de trabajo más complejas y exigentes. De esta forma, los programas y aplicaciones de los ordenadores han evolucionado para aprovechar al máximo estas prestaciones, lo que ha llevado a la creación de programas que requieren más recursos que nunca.

A causa de la impresionante evolución de hardware, es que hoy en día tenemos una gran diversidad de componentes de ordenadores. Estos componentes pueden variar en sus prestaciones, fabricantes, precio y rendimiento. De esta forma el consumidor puede elegir entre distintos candidatos a la hora de comprar un componente para su ordenador.

En consecuencia con lo anterior, cada vez son más las personas que necesitan un ordenador, pero son pocos los que disponen de unos conocimientos informáticos para elegir los distintos componentes que formarán parte de su ordenador. Esto conlleva a que muchas personas opten por comprar un ordenador ensamblado, los cuales suelen ser de un precio muy superior al de sus componentes individuales. Además, estos ordenadores pueden no cumplir con la necesidades de algunas personas debido a sus prestaciones bastante limitadas e incluso suelen volverse obsoletos en pocos años debido a que usan algunos componentes que llevan varios años en el mercado.

Por otra parte, las personas que sí dispongan de este conocimiento es posible que se encuentren con otro problema. Este problema es que no disponen del tiempo necesario para realizar una búsqueda exhaustiva de los distintos componentes que necesita, comparar sus prestaciones y sus precios.

En este proyecto queremos ayudar a esas personas que no dispongan del conocimiento o tiempo necesario para montarse su propio ordenador. Para ello les ofreceremos los distintos componentes necesarios para montarse su ordenador según sean sus necesidades.

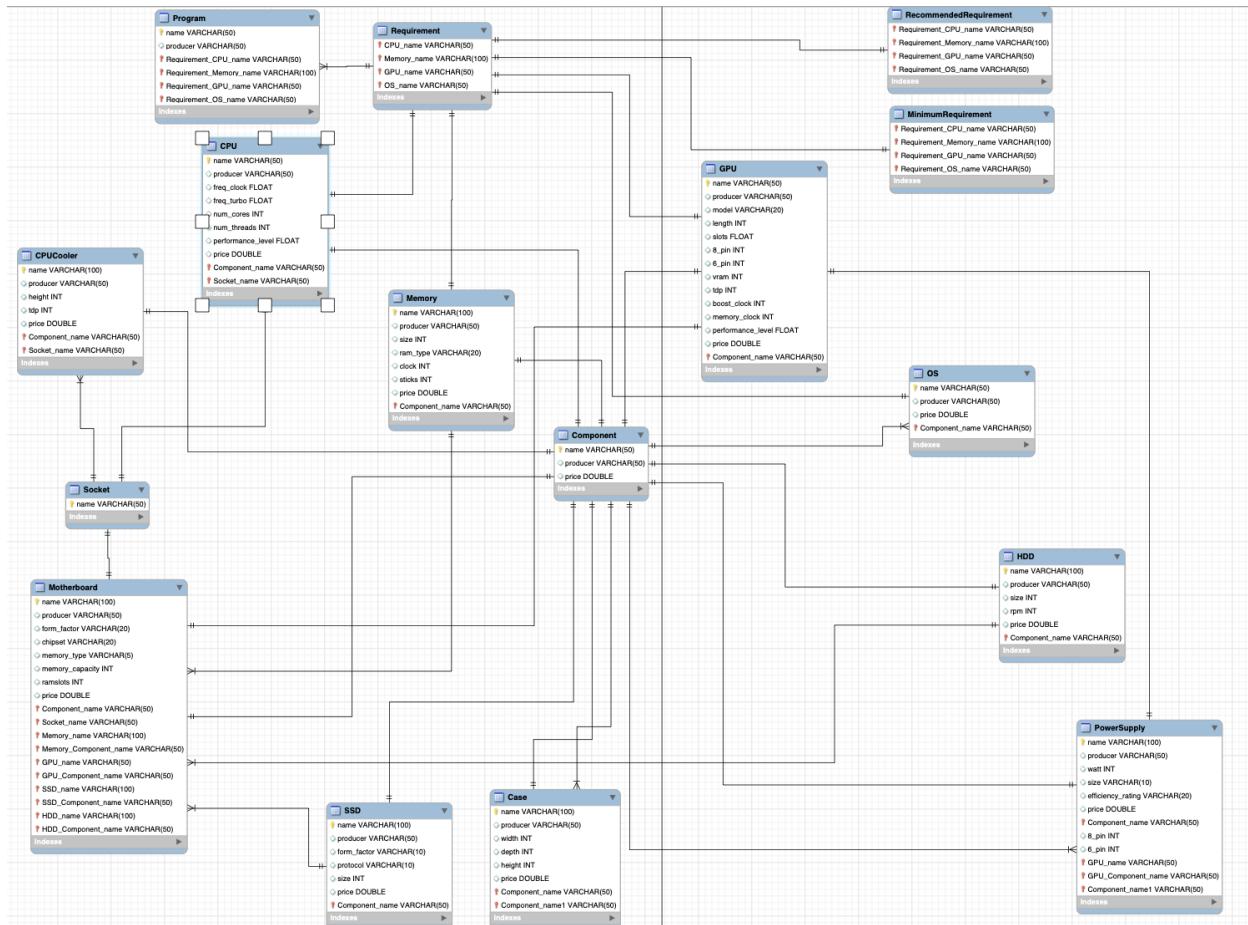
2. OBJETIVOS

Uno de los objetivos principales de este proyecto es otorgar a las personas una herramienta mediante la cual seleccionen un programa que se adapte a sus necesidades específicas y que a partir de este programa se diseñen dos ordenadores. El primero cumplirá con los requisitos mínimos requeridos por el programa y el segundo satisfacerá con los requisitos recomendados, integrando ambos computadores los componentes más económicos y compatibles entre sí que cumplan los requisitos. De esta forma, la persona puede elegir el ordenador que más se adecue a sus necesidades teniendo en cuenta que ambos están hechos a medida para su programa.

Para lograr este objetivo, necesitaremos almacenar en una base de datos información sobre los componentes y los programas. En relación a los componentes guardaremos sus prestaciones y ciertos aspectos para determinar su compatibilidad. Respecto a los programas almacenaremos sus requisitos mínimos y recomendados, de esta manera el usuario podrá elegir cuál se adapta más a sus necesidades.

Una vez dispongamos de la base de datos, necesitaremos crear un modelo que sea capaz de analizar los requisitos del programa introducido y de cada componente requerido por el mismo, con el objetivo de buscar uno con las mismas prestaciones o mejores que tenga un precio más bajo. Este modelo también ha de asegurar que la combinación de los distintos componentes sea totalmente compatible.

3. ESQUEMA E-R



Los atributos de la entidad "componente", tales como nombre, precio y fabricante, son información importante para los usuarios que buscan comprar o actualizar sus componentes. Además, la compatibilidad y las prestaciones de los componentes también son elementos clave a considerar, ya que pueden afectar al rendimiento general del ordenador y al posible no funcionamiento.

En concreto, los componentes, como la GPU, la memoria RAM, la caja del ordenador, el procesador (CPU), el motherboard, el refrigerador del procesador y la fuente de alimentación, son esenciales para el adecuado funcionamiento de un ordenador.

La GPU es responsable del procesamiento gráfico, la RAM es importante para el almacenamiento temporal de datos y la CPU es el cerebro del ordenador que ejecuta los programas. La caja del ordenador es la estructura que protege a los componentes y le da una forma física, mientras que la fuente de alimentación es la que proporciona energía a los otros componentes.

Es importante tener en cuenta la compatibilidad y las prestaciones de cada uno de estos componentes para asegurar que funcionen bien en su conjunto y para optimizar el rendimiento

global del ordenador. Por ejemplo, si se quiere jugar a videojuegos de última generación, es necesario asegurarse de que la GPU sea suficientemente potente para procesar los gráficos de manera eficiente.

RELACIONES ENTRE COMPONENTES

Para que un ordenador funcione correctamente, es importante que todos los componentes estén bien conectados y sean compatibles entre sí. Las relaciones entre componentes que son clave para asegurar el buen funcionamiento de un ordenador son las siguientes:

- **Compatibilidad de la MOTHERBOARD con la RAM:** Es importante que la MOTHERBOARD del PC sea compatible con la RAM. Esto significa que la frecuencia y la capacidad de la RAM deben ser soportadas por este componente. Si la RAM no es compatible, el equipo no funcionará correctamente.
- **Compatibilidad de la MOTHERBOARD con la GPU:** Así como con la RAM, la MOTHERBOARD también debe ser compatible con la GPU para asegurar que los gráficos se procesen correctamente. La GPU debe tener una interfaz compatible con la MOTHERBOARD, y la fuente de alimentación debe ser lo suficientemente potente para alimentar a la GPU teniendo la fuente de alimentación el cable con el número de PINES necesarios para alimentar la GPU.
- **Compatibilidad de la CPU con la MOTHERBOARD:** La MOTHERBOARD debe tener un SOCKET que sea compatible con el procesador para que puedan trabajar juntos. Es importante verificar las especificaciones del procesador y la placa base para asegurarse de que sean compatibles entre sí.
- **Compatibilidad de la fuente de alimentación con el resto de los componentes:** La fuente de alimentación es responsable de suministrar energía a los demás componentes del ordenador. Es importante asegurarse de que la fuente de alimentación sea lo suficientemente potente para alimentar a todos los componentes, incluyendo la MOTHERBOARD (incluyendo indirectamente los componentes conectados a ella como CPU y RAM) la GPU y los dispositivos de almacenamiento (SSD o HDD).
- **Compatibilidad del sistema operativo con el resto de los componentes:** Por último, es importante asegurarse de que el sistema operativo sea compatible con todos los componentes del ordenador. Esto significa que el sistema operativo debe reconocer todos los dispositivos de hardware y tener los controladores necesarios para ellos.
- **Compatibilidad del CPU COOLER con el CPU:** El CPU COOLER se encarga de disipar el calor generado por la CPU. Por esto es muy importante que el CPU COOLER tenga un TDP, que en este caso es la cantidad de calor que puede disipar, el cual sea adecuado a la cantidad de calor que genera la CPU. Este ha de conectarse mediante el SOCKET de CPU en la MOTHERBOARD, como consecuencia han de ser compatibles y tener las mismas dimensiones.

- **Compatibilidad de la MOTHERBOARD con SSD y HDD:** La MOTHERBOARD debe ser capaz de conectar unidades de almacenamiento. Estas pueden ser SSD, HDD o ambas, por lo que generalmente disponen de puertos SATA para su conexión. Por otra parte, esta compatibilidad también puede ser definida por el Form Factor que nos dice qué tipos de SSD se pueden conectar directamente a la MOTHERBOARD.
- **Compatibilidad de CASE con el resto de componentes:** CASE es el encargado de contener los distintos componentes de hardware de un ordenador. Estos tienen unas medidas determinadas que nos limitan a la hora instalar un componente en su interior a causa sus medidas han de ser compatibles principalmente su ancho y profundidad. Por esto es muy importante elegir una CASE que sea capaz de contener todos nuestros componentes. Las dimensiones GPU, MOTHERBOARD, POWER SUPPLY y CPU COOLER son los que principalmente determinarán la compatibilidad con CASE ya que son los que más espacio suelen ocupar.

REQUISITOS DE PROGRAMAS

Para asegurarse de que los componentes pueden hacer funcionar correctamente un programa concreto, es importante conocer los requisitos del sistema que este programa necesita. La mayoría de los programas tienen una lista de requisitos del sistema en su página web o en su documentación. Esta documentación la guardamos en la entidad “requirements”. Donde tenemos por una parte los requisitos mínimos y los recomendados. Los requisitos mínimos son las especificaciones técnicas mínimas necesarias que deben tener los componentes del sistema para ejecutar un programa o juego de manera básica. Estos requisitos se refieren a la cantidad mínima de memoria RAM, capacidad de almacenamiento, velocidad del CPU y GPU que necesita el programa o juego para funcionar.

Por otro lado, los requisitos recomendados son las especificaciones técnicas recomendadas para que el programa o juego se ejecute de manera óptima, brindando una experiencia de usuario más fluida y sin problemas. Estos requisitos suelen ser más elevados que los mínimos y se refieren a una mayor cantidad de memoria RAM, una GPU más potente, mayor capacidad de almacenamiento y una CPU más rápida.

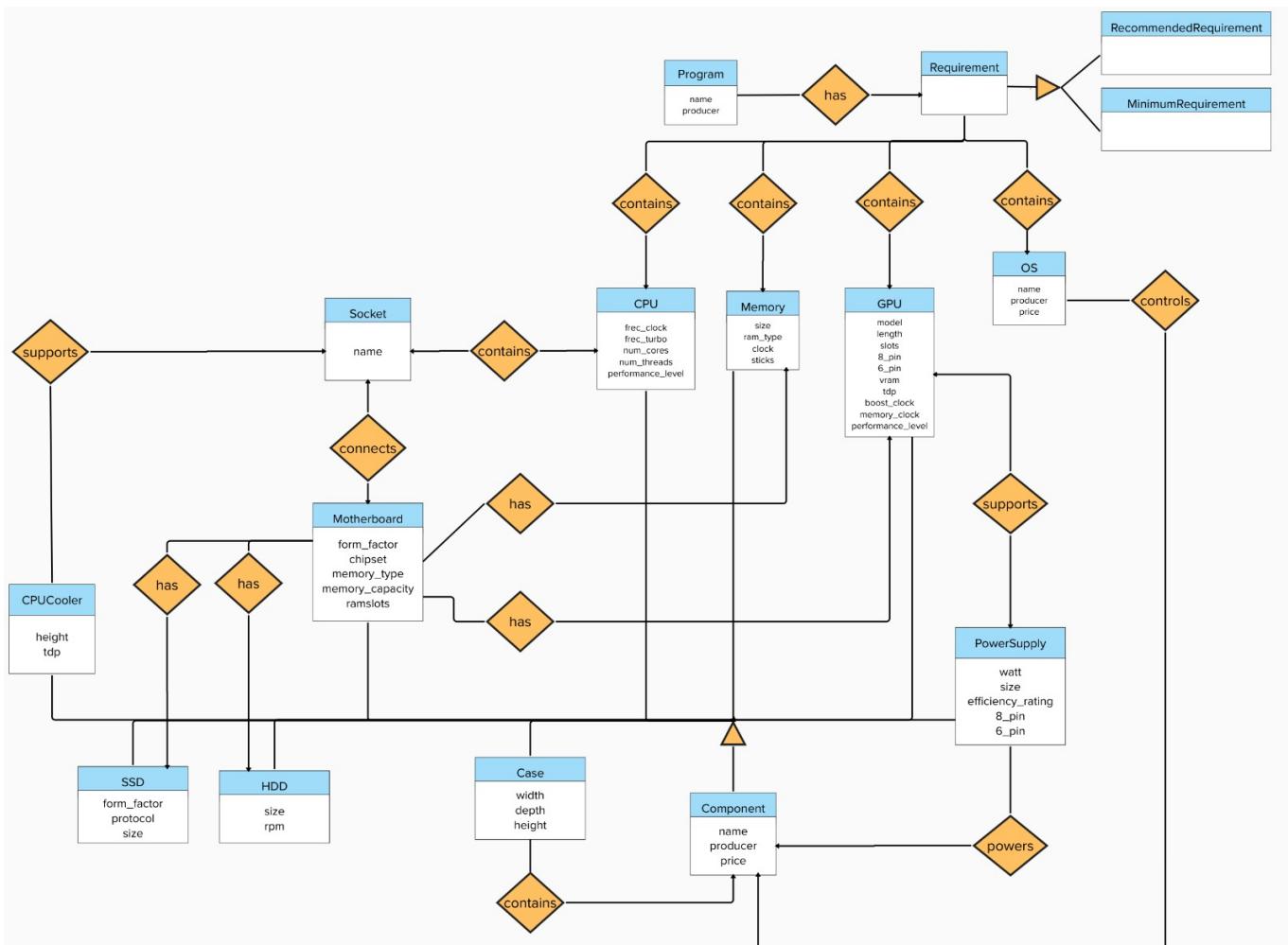
Por ejemplo, si se desea instalar un juego que requiere una GPU potente, es importante asegurar que la GPU sea suficientemente potente para procesar los gráficos del juego. Si el juego requiere una cantidad elevada de memoria RAM, es importante asegurar que la placa base soporte la capacidad de RAM necesaria y que la frecuencia de la RAM sea compatible con la placa base.

Es importante comprobar si los componentes cumplen con los requisitos mínimos del sistema para el programa que quieras ejecutar. Esto puede incluir la cantidad de memoria RAM, la capacidad de la GPU, la potencia de la fuente de alimentación y otras especificaciones. El rendimiento del programa está afectado por otros factores, como la capacidad del disco duro o la velocidad del procesador.

REQUISITOS DEL SISTEMA

MÍNIMO:	RECOMENDADO:
Requiere un procesador y un sistema operativo de 64 bits	Requiere un procesador y un sistema operativo de 64 bits
SO: 64bit Versions of Windows 7, Windows 8, Windows 10	SO: 64bit Versions of Windows 7, Windows 8, Windows 10
Procesador: Intel Core i3 2130 or AMD FX 4300	Procesador: Intel Core i5 8600K or AMD Ryzen 5 2600X
Memoria: 8 GB de RAM	Memoria: 8 GB de RAM
Gráficos: Nvidia GT 640 or AMD HD 7750	Gráficos: Nvidia GTX 1060 or AMD RX 580
DirectX: Versión 11	DirectX: Versión 11
Red: Conexión de banda ancha a Internet	Red: Conexión de banda ancha a Internet
Almacenamiento: 50 GB de espacio disponible	Almacenamiento: 50 GB de espacio disponible
Tarjeta de sonido: DirectX Compatible Soundcard	Tarjeta de sonido: DirectX Compatible Soundcard
Notas adicionales: Requires a 64-bit processor and operating system	Notas adicionales: Requires a 64-bit processor and operating system

Ejemplo de requisitos de un programa en la plataforma Steam



Hemos diseñado el esquema de entidad relación anterior en el que reflejamos las relaciones explicadas anteriormente. Vemos que todos los componentes son heredados de la entidad componente y que el componente caja contiene componentes.

También tenemos la entidad requerimiento que tiene una GPU, una CPU, una memoria RAM y un Sistema Operativo (OS) que son los requerimientos que tiene el programa para poder ejecutarse. De esta entidad heredan los requerimientos mínimos y recomendados.

4. RECOPILACIÓN DE DATOS

Para abordar este proyecto, después de haber trabajado en las relaciones de las tablas, hemos tenido que buscar la forma de llenar nuestra base de datos. Dada la gran cantidad de componentes existentes actualmente, no hemos visto viable la opción de llenar nuestra propia base de datos escribiendo a mano. Por lo tanto, hemos optado por utilizar Web Scraping como solución.

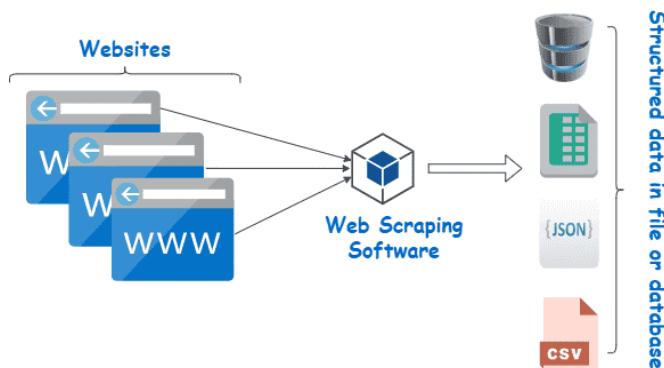
El Web Scraping es una técnica que se ha utilizado para extraer información de páginas web de forma automatizada. Ha consistido en analizar y recopilar datos de diferentes sitios web, extrayendo la información relevante y almacenándola en una base de datos u otro formato estructurado.

En nuestro caso, hemos utilizado Python como lenguaje de programación y dos bibliotecas populares: BeautifulSoup y Requests.

Beautiful Soup: Ha sido una biblioteca de Python que se ha utilizado para extraer datos de documentos HTML y XML. Ha proporcionado una forma sencilla de navegar y buscar elementos específicos en una página web y extraer el contenido deseado. BeautifulSoup también ha tenido funciones para manipular y formatear los datos extraídos.

Requests: Ha sido una biblioteca de Python que se ha utilizado para realizar solicitudes HTTP. Ha permitido enviar solicitudes a un servidor web y recibir las respuestas correspondientes. En el contexto del Web Scraping, Requests se ha utilizado para acceder a las páginas web y obtener el contenido HTML que se ha analizado y extraído con BeautifulSoup.

Utilizando estas herramientas, hemos podido automatizar la extracción de datos relevantes de los sitios web que han contenido la información de los componentes necesarios para nuestra base de datos. Esto nos ha permitido ahorrar tiempo y esfuerzo al evitar la tarea manual de recopilar y escribir los datos de forma individual.



Después de contemplar varias opciones, hemos elegido el sitio web www.pc-kombo.com para la extracción de datos debido a varias razones:

- **Cantidad de Componentes:** Este sitio web ha sido elegido por la gran cantidad de componentes que ofrece, lo que nos ha permitido obtener una amplia variedad de datos para completar nuestra base de datos.

- **Estructura de la web sencilla:** La estructura HTML de www.pc-kombo.com ha sido sencilla y bien organizada, lo que ha facilitado el proceso de extracción de datos. Hemos podido identificar fácilmente los elementos relevantes en las páginas web y extraer la información necesaria utilizando técnicas de Web Scraping.
- **Precios de los componentes:** Al extraer estos datos, hemos podido obtener información actualizada sobre los precios de los componentes, lo cual es importante para nuestro proyecto.
- **Características técnicas detalladas:** Otra gran razón que nos ha decantado por escoger esta web es porque nos ha proporcionado un alto grado de detalle en las características de los componentes. Hemos podido obtener información detallada sobre las especificaciones técnicas, compatibilidad, rendimiento y otros detalles relevantes de cada componente.

Para el desarrollo del scrapper, hemos implementado la siguiente función:

```
def make_soup(url):
    html = urlopen(url)
    return BeautifulSoup(html, 'html.parser')
```

Esta función llamada `make_soup` acepta una URL como argumento y realiza las siguientes acciones:

1. Utiliza la función `urlopen(url)` para abrir la URL especificada y obtener el contenido HTML de la página web.
2. Toma el contenido HTML obtenido y lo pasa como argumento a `BeautifulSoup`, junto con el parser '`html.parser`'.
3. La función `BeautifulSoup` procesa el contenido HTML y lo estructura en un objeto `BeautifulSoup`, que permite acceder y manipular fácilmente los elementos del HTML.
4. Por último, la función retorna el objeto `BeautifulSoup` resultante.

Posteriormente, hacemos uso de las funciones `soup.find()` y `soup.find_all()`, que son métodos utilizados en `Beautiful Soup`. Ambas funciones se utilizan para buscar elementos específicos dentro de la estructura del documento analizado.

La función `soup.find()` se utiliza para encontrar y devolver el primer elemento que coincide con los criterios de búsqueda especificados. Toma como argumento una etiqueta HTML o una lista de etiquetas, junto con cualquier atributo o clase que se desee buscar.

Por ejemplo, en una parte de nuestro código queremos encontrar el primer `span` de la clase `socket`, entonces, hemos hecho uso de la siguiente función `subtitle.find("span", {"class": "socket"})`. Esta función devuelve el primer elemento que cumpla con los criterios de búsqueda.

Por otro lado, la función `soup.find_all()` se utiliza para encontrar y devolver todos los elementos que coincidan con los criterios de búsqueda especificados. Al igual que `soup.find()`, toma como

argumento una etiqueta HTML o una lista de etiquetas, junto con atributos o clases opcionales. Por ejemplo, si queremos encontrar todos los elementos <a> con la clase "enlace", podemos usar soup.find_all('a', {'class': 'enlace'}). Esta función devuelve una lista de todos los elementos que cumplen con los criterios de búsqueda.

Estas dos funciones se pueden usar en conjunto para hacer búsquedas más avanzadas. En nuestro código, la utilizamos para recorrer listas de elementos en html.

```
for i in soup.find("ol", {"id": "hardware"}).find_all('li'):
```

Una vez realizado el Web Scraping y obtenidos los datos de interés, procedemos a importarlos al workbench de MySQL. En este caso, los datos se han preparado en formato CSV para facilitar la importación.

Por último utilizamos la misma técnica de Scraping para 'scrapear' la página de Web de Steam. Con esta técnica conseguimos también scrapear los juegos más populares de la plataforma líder en el mundo del gaming. Nos hemos encontrado con una dificultad: hay juegos que al acceder a la url te pregunta por tu edad. Para ello al scraping le pasamos una cookie con una edad para anular la aparición de dicho diálogo al entrar en la web y ver directamente el contenido.

```
cookies = {'birthtime': '568022401'}
res = requests.get(url, cookies=cookies)
res.raise_for_status()
soup = BeautifulSoup(res.text, 'html.parser')
return soup
```

5. CREACIÓN DE LA BASE DE DATOS

Una vez tenemos todos los datos de los componentes obtenidos mediante el scrapping, podemos crear nuestra base de datos. En primer lugar, lo que hicimos fue corregir los archivos csv donde estaban guardados los datos, estos contenían columnas que estaban movidas o vacías y las tuvimos que rectificar.

A continuación, creamos nuestro schema que contiene estos datos y dentro del mismo añadimos una tabla para programa y cada tipo de componente. En estas tablas añadimos las distintas columnas que contendrán la información necesaria.

Algunos componentes como las placas base y las fuentes de alimentación utilizaban estándares de referencias para definir su tamaño, por lo que tuvimos que pasar estos estándares a medidas en milímetros para poder realizar cálculos con ellos. Para añadir estas medidas de ancho, altura y profundidad usamos las siguientes conversiones:

Para **fuente de alimentación**:

- ATX (Advanced Technology eXtended):
 - Ancho: 150 mm
 - Alto: 86 mm
 - Profundidad: 140 mm
- SFX (Small Form Factor eXtended):
 - Ancho: 125 mm
 - Alto: 63.5 mm
 - Profundidad: 100 mm
- TFX (Thin Form Factor):
 - Ancho: 85 mm
 - Alto: 70 mm
 - Profundidad: 175 mm

Para **placa base**:

- ATX (Advanced Technology eXtended):
 - Ancho: 305 mm
 - Alto: 244 mm
- Micro-ATX:

- Ancho: 244 mm
- Alto: 244 mm
- Mini-ITX:
 - Ancho: 170 mm
 - Alto: 170 mm
- E-ATX (Extended ATX):
 - Ancho: 305 mm
 - Alto: 330 mm
- XL-ATX (eXtended Length ATX):
 - Ancho: 343 mm
 - Alto: 330 mm
- MINI-STX (Mini Socket Technology Extended):
 - Ancho: 147 mm
 - Alto: 140 mm
- CEB (Compact Electronics Bay):
 - Ancho: 305 mm
 - Alto: 267 mm

Una vez hecho todos estos cambios, obtuvimos el siguiente schema y tablas:

Tablas del schema:

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length
case	InnoDB	10	Dynamic	1240	132	160.0 KIB
cooler	InnoDB	10	Dynamic	251	261	64.0 KIB
cpu	InnoDB	10	Dynamic	78	210	16.0 KIB
gpu	InnoDB	10	Dynamic	486	202	96.0 KIB
hdd	InnoDB	10	Dynamic	199	246	48.0 KIB
motherboard	InnoDB	10	Dynamic	617	159	96.0 KIB
power	InnoDB	10	Dynamic	195	252	48.0 KIB
program	InnoDB	10	Dynamic	17	963	16.0 KIB
ram	InnoDB	10	Dynamic	767	149	112.0 KIB
ssd	InnoDB	10	Dynamic	270	182	48.0 KIB

Tabla de case:

Column	Type
depth	float
height	float
max	float
name	varchar(120)
price	float
producer	varchar(30)
width	float

Tabla de cooler:

Column	Type
height	float
name	varchar(140)
price	float
producer	varchar(30)
sockets	varchar(140)

Tabla de CPU:

Column	Type
clock	decimal(2,1)
cores	int
name	varchar(80)
performances	float
price	float
producer	varchar(20)
socket	varchar(20)
threads	int
turbo	decimal(2,1)

Tabla de GPU:

Column	Type
6pin	int
8pin	int
bclock	int
length	float
mclock	int
name	varchar(80)
perfomance	float
price	float
producer	varchar(20)
serie	varchar(45)
slots	decimal(3,2)
tdp	int
vram	int

Tabla de HDD:

Column	Type
name	varchar(80)
price	float
producer	varchar(40)
rpm	int
size	int

Tabla de SSD:

Column	Type
ffactor	varchar(10)
name	varchar(80)
price	float
producer	varchar(20)
protocol	varchar(10)
size	int

Tabla de motherboard:

Column	Type
chipset	varchar(30)
height	float
memorycapacity	int
memorytype	varchar(10)
name	varchar(80)
price	float
producer	varchar(20)
ramslots	int
size	varchar(20)
socket	varchar(10)
width	float

Tabla de power:

Column	Type
6pin	int
8pin	int
depth	float
efrating	varchar(45)
height	float
name	varchar(120)
price	float
producer	varchar(45)
size	varchar(20)
watts	int
width	float

Tabla de program:

Column	Type
mcpu	varchar(45)
mgpu	varchar(45)
mram	int
mstorage	int
name	varchar(100)
rcpu	varchar(45)
rgpu	varchar(45)
rram	int
rstorage	int

Tabla de RAM:

Column	Type
clock	int
name	varchar(80)
price	float
producer	varchar(20)
size	int
sticks	int
type	varchar(20)

6. OBTENCIÓN DEL ORDENADOR

Una vez tenemos nuestra base de datos creada con la información de los distintos componentes y programas, podemos llevar a cabo el objetivo de este proyecto. Para cumplirlo necesitamos obtener el ordenador más barato dado unos requisitos mínimos o recomendados de un programa. Por lo tanto, tenemos que hacer un query sobre nuestra base de datos que nos devuelva un pc compuesto por sus diferentes componentes que son: el procesador, la gráfica, la ventilación, la placa base, la memoria RAM, SSD y HDD, la fuente de alimentación y la caja que contendrá todos los componentes anteriores.

En primer lugar, necesitamos obtener los requisitos, sean mínimos o recomendados, de un programa. Estos consisten en un procesador y una gráfica que se encuentran en las tablas CPU y GPU. El primer paso es obtener estos componentes de las tablas con toda su información mediante sus nombres. De estos datos, nos quedamos con los valores de la columna performance que nos servirán como referencia para encontrar procesadores y gráficas iguales o superiores a las requeridas. Después, nos quedamos con la cantidad de memoria requerida por el programa tanto de RAM como de almacenamiento.

```
SET @program = 'Cyberpunk';
SET @gpu_base = (SELECT performance FROM gpu WHERE serie LIKE CONCAT('%', (SELECT rgpu FROM program WHERE name = @program), '%') LIMIT 1);
SET @cpu_base = (SELECT performances FROM cpu WHERE name LIKE CONCAT('%', (SELECT rcpu FROM program WHERE name = @program), '%')) ORDER BY performances LIMIT 1;
SET @ram = (SELECT rram FROM program WHERE name = @program);
-- SSD en GB
SET @ssd = (SELECT rstorage FROM program WHERE name = @program);
-- HDD en TB
SET @hdd = (SELECT rstorage FROM program WHERE name = @program);
```

En el ejemplo anterior, vemos que se están obteniendo los requisitos recomendados de un programa. Si queremos obtener los requisitos mínimos, es tan sencillo como cambiar rgpu, rcpu, rram y rstorage por mgpu, mcpu, mram y mstorage respectivamente.

La columna de performance de las tablas de GPU y CPU son unos valores de referencias del rendimiento de los componentes. Estos los hemos obtenido de la página [pc-kombo](#), específicamente de su base de datos de hardware. Esta página contiene la ratio de rendimiento en una escala sobre 10 de cada uno de los componentes, basándose en varios benchmarks.

Tras analizar esta ratio, nos pareció que era muy coherente a la hora de comparar los diferentes procesadores y gráficas, por lo que decidimos usarlo como medida de referencia.

Después de obtener los datos necesarios para el programa, procedemos a crear la consulta que devuelva el PC más económico. Inicialmente, intentamos generar esta consulta de manera directa, considerando únicamente la compatibilidad entre los componentes. Sin embargo, nos percatamos de que escalaba rápidamente y resultaba totalmente ineficiente. En cuestión de minutos, era capaz de agotar la capacidad de memoria de nuestros ordenadores debido a los datos temporales generados por la consulta.

Con el fin de mejorar su eficiencia, fue necesario descomponer esta query en subqueries más pequeñas. Estas debían tener en cuenta la compatibilidad entre los componentes y solo retener los elementos de menor precio. Para ello, identificamos las limitaciones de compatibilidad que afectaban a la combinación de los componentes, lo que nos llevó a obtener las siguientes limitaciones de compatibilidad:

- Socket: CPU, Cooler y Motherboard
- Pins (6 pins y 8 pins): GPU y Power Supply
- RAM slots: Motherboard y RAM
- Size: Case, Cooler, Motherboard, GPU y Power Supply

De estas compatibilidades, podemos ver que tanto la memoria SSD y HDD no se ven limitadas, por lo que podemos simplemente obtener la de menor precio que cumpla con los requisitos del programa y añadirlas al PC (resultado del query).

```

JOIN (SELECT name AS Ssd, price AS Price
      FROM ssd
     WHERE size >= @ssd
     ORDER BY price
    LIMIT 1) q4

JOIN(SELECT name AS Hdd, price AS Price
      FROM hdd
     WHERE size*1000 >= @hdd
     ORDER BY price
    LIMIT 1) q5
  
```

Por otra parte, podemos apreciar que la compatibilidad de tamaño que representa si los componentes caben en la caja o no, abarca casi todos los componentes. Por lo que esta será la query principal que contendrá las subqueries de SDD y HDD.

```

SELECT q3.Cpu AS CPU, q3.Cooler, q3.Motherboard,q3.Ram AS RAM, q2.Gpu AS GPU, q2.Power AS Power_Supply,
q1.CASE_ AS Case_, q4.Ssd AS SSD, q5.Hdd AS HDD,
q1.Price + q2.Price + q3.Price + q4.Price + q5.Price AS Price
 $\oplus$  FROM (SELECT DISTINCT C.name AS CASE_, C.width AS width, C.height AS height,
  
```

En esta query, se comprobará que las dimensiones de los diferentes componentes (altura, ancho y profundidad) sean menores que las dimensiones de la caja. Por lo tanto, primero necesitamos obtener la caja y, para ello, mediante una subquery, nos quedamos con la caja más barata para cada una de las diferentes dimensiones de cajas que están en nuestra tabla case.

```

FROM (SELECT DISTINCT C.name AS CASE_, C.width AS width, C.height AS height,
C.depth AS depth, C.price AS Price FROM project.case AS C
JOIN(SELECT DISTINCT width, depth, height, MIN(price) AS price FROM project.case
GROUP BY width, depth, height) q1
WHERE C.height = q1.height AND C.width = q1.width AND C.depth = q1.depth AND C.price = q1.price) q1
  
```

Por otro lado, necesitamos otras subqueries dentro de la query de size que contemplen las compatibilidades de socket, pins y ram slots. Como se puede apreciar, Power Supply y GPU

están aisladas del resto de componentes teniendo en cuenta sólo la compatibilidad de los pines de 6 y de 8.

Entonces, creamos una subquery que obtenga cuál es el conjunto de GPU y Power Supply más barato, que sean compatibles en cuanto a los pines y que el performance de la gráfica sea mayor o igual al requerido por el programa para cada una de las diferentes dimensiones de GPU y power supplies que están las tablas GPU y power.

```
JOIN(SELECT DISTINCT GPU.name AS Gpu, POWER.name AS Power, GPU.length, POWER.width,
POWER.height, POWER.depth, GPU.price + POWER.price AS Price
FROM gpu GPU
JOIN power POWER ON POWER.`6pin` >= GPU.`6pin` AND POWER.`8pin` >= GPU.`8pin`
JOIN(SELECT GPU.length, POWER.width, POWER.height, POWER.depth, MIN(GPU.price + POWER.price) AS Price
FROM gpu GPU
JOIN power POWER ON POWER.`6pin` >= GPU.`6pin` AND POWER.`8pin` >= GPU.`8pin`
WHERE GPU.performance >= @gpu_base
GROUP BY GPU.length, POWER.width, POWER.height, POWER.depth) q1
ON GPU.length = q1.length AND POWER.width = q1.width AND POWER.height = q1.height AND
POWER.depth = q1.depth AND GPU.price + POWER.price = q1.Price AND GPU.performance >= @gpu_base) q2
ON q2.length < q1.depth AND q2.width < q1.width AND q2.height < q1.height AND q2.depth < q1.depth
```

Por último, en la query de size necesitamos una subquery que contenga el resto de los componentes que faltan. Esta última query de size será la subquery que tenga en cuenta la compatibilidad de sockets. La subquery de socket contiene el componente Motherboard que como podemos ver, tiene una compatibilidad de slots con la memoria RAM, por lo que será una subquery que en conjunto con la subquery Cpu-Cooler formarán la subquery de socket. Estas subqueries de Motherboard-RAM y CPU-Cooler deberán ser compatibles en cuanto al tipo de socket.

```
+ JOIN(SELECT q1.Cpu, q1.Cooler, q2.Motherboard, q2.Ram, q1.COOLER_Height, q2.MB_Height,
q2.MB_Width, q1.Price + q2.Price AS Price
FROM
+ (SELECT DISTINCT CPU.name AS Cpu, CPU.socket AS Socket, COOLER.name AS Cooler,
+ JOIN (SELECT DISTINCT MB.name AS Motherboard, MB.height AS MB_Height,
ON q3.MB_Height < q1.height AND q3.MB_Width < q1.depth AND COOLER_Height < q1.height
```

La subquery Motherboard-RAM nos devuelve el conjunto de placa base y memoria RAM más barato, que sean compatibles en cuanto a los ram slots y que la capacidad de memoria RAM sea mayor o igual a la requerida por el programa, para cada una las diferentes dimensiones y tipos de sockets de las placas base que están en la tabla de motherboard.

```

JOIN (SELECT DISTINCT MB.name AS Motherboard, MB.height AS MB_Height,
MB.width AS MB_Width, MB.socket AS Socket, RAM.name AS Ram, (MB.price + RAM.price) AS Price
FROM motherboard MB
JOIN ram RAM ON RAM.size >= @ram AND RAM.type LIKE CONCAT('%',MB.memorytype,'%') AND MB.ramslots >= RAM.sticks
JOIN (SELECT MB.socket,MB.width,MB.height, MIN(MB.price + RAM.price) AS price
FROM motherboard MB
JOIN ram RAM ON RAM.size >= @ram AND RAM.type LIKE CONCAT('%',MB.memorytype,'%') AND MB.ramslots >= RAM.sticks
GROUP BY MB.socket,MB.width,MB.height) q1
WHERE MB.socket = q1.socket AND MB.width = q1.width AND MB.height = q1.height
AND (MB.price + RAM.price) = q1.price AND RAM.size >= @ram) q2 ON q1.Socket = q2.Socket) q3
ON q3.MB_Height < q1.height AND q3.MB_Width < q1.depth AND COOLER_Height < q1.height

```

En último lugar, la subquery de CPU-Cooler nos devuelve el conjunto de procesador y ventilación más barato, que sean compatibles en cuanto al tipo de socket y que el rendimiento del procesador sea mayor o igual al requerido por el programa, para cada uno de los distintos tipos socket y dimensiones de los procesadores y ventiladores que están en las tablas de cpu y cooler.

```

FROM
(SELECT DISTINCT CPU.name AS Cpu, CPU.socket AS Socket, COOLER.name AS Cooler,
COOLER.height AS COOLER_Height, CPU.price + COOLER.price AS Price
FROM CPU cpu
JOIN cooler COOLER ON COOLER.sockets LIKE CONCAT('%',CPU.socket,'%')
JOIN
(SELECT CPU.socket,COOLER.height,MIN(CPU.price + COOLER.price) AS price
WHERE CPU.socket = q1.socket AND COOLER.height = q1.height AND CPU.price + COOLER.price = q1.price AND CPU.performances >= @cpu_base) q1

```

Una vez tenemos todas las queries, las ordenamos por el precio, que es la suma del coste de todos los componentes. Al ejecutar la query vimos que tarda varios minutos, seguramente debido a que el resultado de la query tiene aproximadamente 13 millones de filas para los requisitos recomendados del programa Age of Empires II Definitive Edition. Por esto, decidimos limitar el número de filas devueltas a 1000 (las 1000 de menor precio) y vimos que el tiempo de ejecución se reduce considerablemente. Con este cambio, en el mismo ejemplo pasa de 475.156 segundos de duración y 29.922 segundos de fetching a 27.531 segundos de duración y 0.000 segundos de fetching.

```

ORDER BY Price
LIMIT 1000

```

7. COMPROBACIÓN DEL RESULTADO OBTENIDO

Una vez elaborada la query que devuelve un ordenador que cumple con los requisitos especificados, es necesario asegurar que los componentes proporcionados en la consulta son de igual o mejor calidad.

A continuación, mostramos un ejemplo de una consulta. En esta, se muestra la configuración proporcionada por nuestra base de datos para obtener un ordenador que se adapte a los componentes recomendados para el videojuego “Cyberpunk”.

En primer lugar, comprobaremos el procesador. La CPU que requiere el juego es el “AMD RYZEN 3 3200G”. Como se muestra en la siguiente imagen, no existe ningún procesador similar al proporcionado. En este caso, el ordenador resultante tendrá exactamente el mismo procesador de los requerimientos.

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons for database management. Below the toolbar, the main area has tabs: 'project*', 'query', and 'SQL File 4*'. The 'query' tab is active, displaying the following SQL code:

```
1 • SELECT * FROM cpu WHERE name LIKE CONCAT('%AMD Ryzen 3 3200G%')
```

Below the code, the results are displayed in a grid. The grid has columns labeled: name, socket, turbo, clock, threads, cores, performances, producer, and price. There is one row of data shown:

	name	socket	turbo	clock	threads	cores	performances	producer	price
▶	AMD Ryzen 3 3200G	AM4	4.0	3.6	4	4	3.92	AMD	99.99
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

El siguiente componente que comprobaremos será la tarjeta gráfica. La GPU requerida es “RTX 2060”. Como se muestra en la siguiente imagen, hay una gran variedad de tarjetas gráficas similares.

project* query* SQL File 4* | Limit to 100 rows | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

```
1 •  SELECT * FROM gpu WHERE name LIKE CONCAT('%RTX 2060%')
```

name	serie	vram	tdp	length	slots	8pin	6pin	bclock	mclock	producer	perfomance	price
ASUS GeForce RTX 2060 Dual Mini	GeForce RTX 2060 (6GB)	6	160	197	2.00	1	0	1755	14000	ASUS	3.76	329.99
ASUS GeForce RTX 2060 Dual O6G EVO	GeForce RTX 2060 (6GB)	6	160	242	2.50	1	0	1755	14000	ASUS	3.76	329.99
ASUS GeForce RTX 2060 ROG STRIX O6G	GeForce RTX 2060 (6GB)	6	160	300	3.00	1	1	1860	14000	ASUS	3.76	379.99
EVGA GeForce RTX 2060 SC Gaming	GeForce RTX 2060 (6GB)	6	160	189.89	2.75	1	0	1710	14000	EVGA	3.76	293.99
EVGA GeForce RTX 2060 XC Black Gaming	GeForce RTX 2060 (6GB)	6	160	172.72	3.00	1	0	1680	14000	EVGA	3.76	340.99
EVGA GeForce RTX 2060 XC Gaming	GeForce RTX 2060 (6GB)	6	160	172.72	3.00	1	0	1755	14000	EVGA	3.76	360.99
EVGA GeForce RTX 2060 XC Ultra Black Gaming	GeForce RTX 2060 (6GB)	6	160	269.2	2.00	1	0	1680	14000	EVGA	3.76	389.99
EVGA GeForce RTX 2060 XC Ultra Gaming	GeForce RTX 2060 (6GB)	6	160	269.2	2.00	1	0	1830	14000	EVGA	3.76	379.99
Gigabyte Aorus GeForce RTX 2060 Xtreme 6G	GeForce RTX 2060 (6GB)	6	190	290	2.00	1	1	1845	14000	Gigabyte	3.76	399.99
Gigabyte GeForce RTX 2060 D6	GeForce RTX 2060 (6GB)	6	160	225.6	2.00	1	0	1680	14000	Gigabyte	3.76	259.99
Gigabyte GeForce RTX 2060 Gaming OC 6G	GeForce RTX 2060 (6GB)	6	160	280	2.50	1	0	1830	14000	Gigabyte	3.76	379.99

En este caso, nos quedaríamos con la tarjeta gráfica que tenga el peor performance level. A partir de esta, buscaríamos una con mejor rendimiento y un precio menor. Como resultado, nuestra query devuelve la GPU “Gigabyte Radeon RX 6600 Eagle”.

Para verificar que la elección es correcta utilizamos una página web llamada [versus](#), que permite comparar componentes de hardware teniendo en cuenta sus características.

La comparación obtenida entre la tarjeta gráfica requerida y la proporcionada es la siguiente:



Gigabyte Radeon RX 6600 Eagle

57 Puntos

CAMPEÓN DE LA COMPARACIÓN

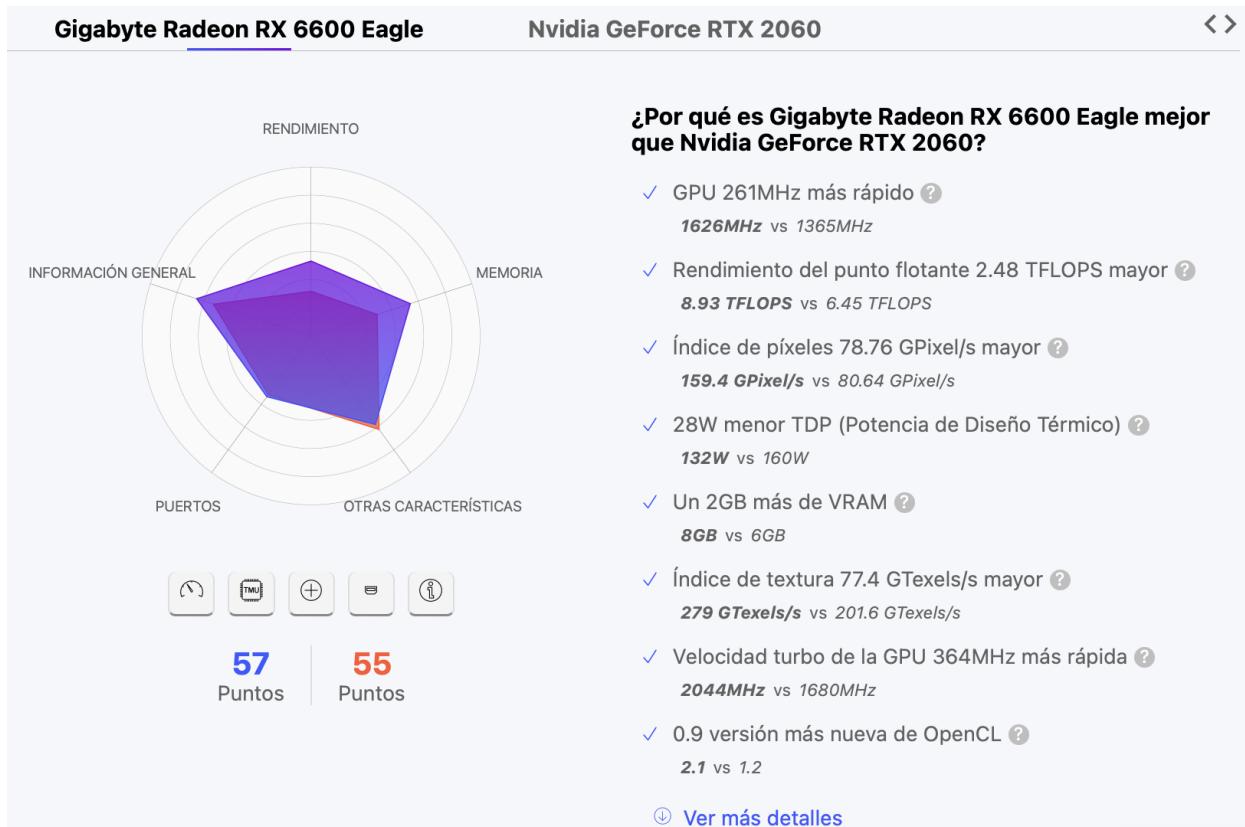
€ 249 amazon



Nvidia GeForce RTX 2060

55 Puntos

€ 320 amazon



Se puede apreciar que la tarjeta gráfica proporcionada por nuestra consulta es mejor elección en relación calidad-precio.

El próximo paso es comprobar la RAM, en este caso, el juego requiere de 16GB de memoria. A continuación, se muestra una imagen con diversos componentes con características similares.

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
1 •  SELECT * FROM ram WHERE size = 16
```

The results grid displays the following data:

	name	size	type	clock	sticks	producer	price
▶	G.Skill Aegis DDR4 DDR4-2133 16GB	16	DDR4-2133	2133	2	G.Skill	34.99
	G.Skill Aegis DDR4 DDR4-3000 16GB	16	DDR4-3000	3000	2	G.Skill	39.99
	Corsair Vengeance LPX Series red DDR4-3200	16	DDR4-3200	3200	2	Corsair	41.99
	Corsair Vengeance LPX Series black DDR4-3200	16	DDR4-3200	3200	2	Corsair	42.99
	G.Skill Ripjaws V DDR4-3200 16GB	16	DDR4-3200	3200	2	G.Skill	43.99
	Corsair Value Select DDR4-2133 CL15	16	DDR4-2133	2133	1	Corsair	44.99
	G.Skill RipJaws V Series	16	DDR4-3200	3200	2	G.Skill	46.99
	G.Skill Sniper X Series	16	DDR4-3200	3200	2	G.Skill	46.99
	Corsair Vengeance LPX Series white	16	DDR4-3200	3200	2	Corsair	47.99
	Corsair 16GB	16	DDR4-2400	2400	1	Corsair	47.99
	ADATA Premier DDR5-4800	16	DDR5-4800	4800	1	ADATA	49.99

Nuestro programa realiza una búsqueda entre las memorias RAM que tengan la capacidad deseada y selecciona en base al menor precio. En este caso, devuelve la RAM “G.Skill Aegis DDR4 DDR4-2133 16GB”.

Los siguientes componentes a comprobar son los de almacenamiento (HDD y SSD). El proceso que sigue la query es buscar tanto un HDD como un SSD, ambos con capacidad superior o igual a la requerida por el videojuego.

Nuestro código ofrece como SSD “Kingston SSDNow A400 Series 120 GB” y HDD “Toshiba X300 HDWE160EZSTA 6TB 128MB”.

Los últimos componentes a comprobar son: ventilador (cooler), placa base (motherboard), batería (power supply) y caja (case). Estos elementos no están implícitos en los requerimientos, sino que debemos buscar los de menor precio que aseguren la compatibilidad y garanticen el funcionamiento del ordenador. A continuación se muestra el resultado completo de la query.

project* query* SQL File 4*

```

1 • SET @program = 'Cyberpunk';
2 • SET @gpu_base = (SELECT performance FROM gpu WHERE serie LIKE CONCAT('%', (SELECT rgpu FROM program WHERE name = @program), '%')) LIMIT
3 • SET @cpu_base = (SELECT performances FROM cpu WHERE name LIKE CONCAT('%', (SELECT rcpu FROM program WHERE name = @program), '%')) ORDER
4 • SET @ram = (SELECT rram FROM program WHERE name = @program);
5 -- SSD en GB
6 • SET @ssd = (SELECT rstorage FROM program WHERE name = @program);
7 -- HDD en TB
8 • SET @hd = (SELECT rstorage FROM program WHERE name = @program);

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows: |

CPU	Cooler	Motherboard	RAM	GPU	Power_Supply
AMD Ryzen 3 3200G	Arctic Alpine 64 GT - 80mm	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	Arctic Alpine 64 Pro Rev.2 - 92mm	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	DEEPCOOL GAMMAXX 200T 2 Heat pipes 120mm	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	Arctic Alpine 64 GT - 80mm	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	Arctic Alpine 64 plus - 92mm	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	Deepcool GAMMAXX 300 120mm Hydro	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	Arctic Alpine 64 GT - 80mm	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	Antec A30 - 92mm	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	Arctic Alpine 64 Pro Rev.2 - 92mm	ASRock A320M-DGS	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3
AMD Ryzen 3 3200G	Arctic Alpine 64 GT - 80mm	Gigabyte AX370-Gaming K3	G.Skill Aegis DDR4 DDR4-2133 16GB	Gigabyte Radeon RX 6600 Eagle	EVGA SuperNOVA G3

project* query* SQL File 4*

```

1 • SET @program = 'Cyberpunk';
2 • SET @gpu_base = (SELECT performance FROM gpu WHERE serie LIKE CONCAT('%', (SELECT rgpu FROM program WHERE name = @program), '%')) LIMIT
3 • SET @cpu_base = (SELECT performances FROM cpu WHERE name LIKE CONCAT('%', (SELECT rcpu FROM program WHERE name = @program), '%')) ORDER
4 • SET @ram = (SELECT rram FROM program WHERE name = @program);
5 -- SSD en GB
6 • SET @ssd = (SELECT rstorage FROM program WHERE name = @program);
7 -- HDD en TB
8 • SET @hd = (SELECT rstorage FROM program WHERE name = @program);

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows: |

Power_Supply	Case_	SSD	HDD	Price
EVGA SuperNOVA G3	Antec P110 Luce Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	599.0400085449219
EVGA SuperNOVA G3	Antec P110 Luce Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	603.0200090408325
EVGA SuperNOVA G3	Antec P110 Luce Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	604.0400085449219
EVGA SuperNOVA G3	be quiet! Silent Base 600 Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	604.3400077819824
EVGA SuperNOVA G3	Antec P110 Luce Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	605.700008392334
EVGA SuperNOVA G3	Antec P110 Luce Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	605.9800090789795
EVGA SuperNOVA G3	Raijintek Arcadia Midi-Tower - white	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	607.7700080871582
EVGA SuperNOVA G3	Antec P110 Luce Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	608.0400085449219
EVGA SuperNOVA G3	be quiet! Silent Base 600 Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	608.3200082778931
EVGA SuperNOVA G3	Antec P110 Luce Midi-Tower - black	Kingston SSDNow A400 Series 120 GB	Toshiba X300 HDWE160EZSTA 6TB 128MB	609.0400085449219

El precio de la configuración completa es de 599\$.

Para comprobar la compatibilidad total entre los componentes hemos utilizado una página web llamada [partpicker](#), que permite crear un pc eligiendo los componentes e informa en caso de existir alguna incompatibilidad.

Compatibility: Warning! These parts have potential issues or incompatibilities. See [details](#) below.

Estimated Wattage: 166W

Component	Selection	Base	Promo	Shipping	Tax	Price	Where	
CPU	AMD Ryzen 3 3200G 3.6 GHz Quad-Core Processor	\$91.00	—	✓Prime	—	⚙ \$91.00	amazon.com	Buy X
CPU Cooler	ARCTIC Alpine 64 GT 25.6 CFM CPU Cooler	—	—	—	—	⚙ No Prices Available	—	Buy X
Motherboard	ASRock A320M-DGS Micro ATX AM4 Motherboard	—	—	—	—	⚙ No Prices Available	—	Buy X
Memory	G.Skill Aegis 16 GB (1 x 16 GB) DDR4-2133 CL15 Memory	\$29.99	—	FREE	—	⚙ \$29.99	BH	Buy X
	+ Add Additional Memory							
Storage	Toshiba X300 6 TB 3.5" 7200 RPM Internal Hard Drive	\$132.99	—	FREE	—	⚙ \$132.99	BH	Buy X
	+ Add Additional Storage							
Video Card	+ Choose A Video Card							
Case	Antec P110 Luce ATX Mid Tower Case	—	—	—	—	⚙ No Prices Available	—	Buy X
Power Supply	EVGA SuperNOVA 650 G3 650 W 80+ Gold Certified Fully Modular ATX Power Supply	\$176.93	—	—	—	⚙ \$176.93	amazon.com	Buy X
Operating System	+ Choose An Operating System							
Monitor	+ Choose A Monitor							
Expansion Cards / Networking	Sound Cards, Wired Network Adapters, Wireless Network Adapters							
Peripherals	Headphones, Keyboards, Mice, Speakers, Webcams							
Accessories / Other	Case Accessories, Case Fans, Fan Controllers, Thermal Compound, External Storage, Optical Drives, UPS Systems							
Total: \$430.91								Buy From Amazon

Se puede apreciar que aparece un posible error entre componentes. El log es el siguiente:

Warning: The ASRock A320M-DGS Micro ATX AM4 Motherboard supports the AMD Ryzen 3 3200G 3.6 GHz Quad-Core Processor with BIOS version P5.90. If the motherboard is using an older BIOS version, upgrading the BIOS will be necessary to support the CPU.

Tal y como muestra la imagen, la página sólo informa que en caso de usar una versión antigua de la BIOS será necesario actualizarla para que no haya ningún problema con la CPU.

Por tanto, podemos concluir que la configuración proporcionada por nuestra query es totalmente compatible y funcional.

Como ejemplo adicional, hemos buscado ordenadores prediseñados que cumplan el requisito de la GPU. Los productos están en venta en la página de [pcComponentes](#).



-1%
Envío gratis

Aures Gaming Minotaur A55
Rx66 AMD Ryzen 5
5500/16GB/512GB SSD/RX 6600

692,89€ 703,89€

Recíbelo entre el lunes 12 y el
martes 13 de junio

Vendido por **POWER ZONE SHOP**

Comparar



-2%
Envío gratis

Aures Gaming Minotaur I131F
RX66 Intel Core i3-
13100F/16GB/512GB SSD/RX...

725,89€ 747,89€

Recíbelo entre el lunes 12 y el
martes 13 de junio

Vendido por **POWER ZONE SHOP**

Comparar



-1%
Envío gratis

Aures Gaming Minotaur A55 RX
66XT AMD Ryzen 5
5500/16GB/512GB SSD/RX...

780,89€ 791,88€

Recíbelo entre el lunes 12 y el
martes 13 de junio

Vendido por **POWER ZONE SHOP**

Comparar



Aures Gaming Basilisk A56 RX 66
AMD Ryzen 5 5600/16GB/1TB
SSD/RX 6600

813,89€

Recíbelo entre el lunes 12 y el
martes 13 de junio

Vendido por **POWER ZONE SHOP**

Comparar

Después de ordenar por precio, el ordenador más barato de la página es aproximadamente cien euros más caro que nuestra configuración.

8. CONCLUSIONES

Al plantear inicialmente nuestro proyecto, establecimos como objetivo principal la creación de una herramienta que pudiera generar una configuración de ordenador basada en los requisitos de un programa específico.

Para lograr esto, iniciamos recopilando una amplia cantidad de datos sobre diferentes elementos de hardware y los requerimientos de diversos programas a través de un proceso de web scraping.

Durante el desarrollo de nuestro proyecto, nos enfrentamos a un desafío importante, asegurarnos de que los componentes seleccionados fueran capaces de satisfacer las necesidades específicas del programa y, al mismo tiempo, garantizar la compatibilidad entre ellos.

Para abordar este desafío, realizamos un análisis exhaustivo de las relaciones entre los componentes, identificando las características clave que debíamos considerar al seleccionar la configuración. Este paso fue fundamental para establecer la estructura de los componentes en nuestra base de datos.

La última fase del proyecto se centró en desarrollar una consulta que pudiera proporcionar una configuración de ordenador con el mejor precio posible, al mismo tiempo que asegurara un rendimiento óptimo.

Durante esta fase, definimos varias métricas para comparar los componentes entre sí, y de esta manera, garantizar una elección de elementos adecuada para el programa en cuestión.

Al finalizar el proyecto, nos complace destacar que, tras realizar diversas comprobaciones a través de diferentes páginas web, las configuraciones que proporciona nuestro código son completamente compatibles y funcionales.

Estamos satisfechos con los resultados obtenidos y el conocimiento que hemos adquirido sobre las bases de datos durante la elaboración de este proyecto.

9. ANEXO

9.1. ARCHIVO DE SCRAPING

```
from bs4 import BeautifulSoup
import urllib3
from urllib.request import urlopen
import smtplib
from flask import Flask, request, Response
import re
import time
from textblob import TextBlob

def make_soup(url):
    html = urlopen(url)
    return BeautifulSoup(html, 'html.parser')
```

CPUS

```
# exportar los datos de los procesadores a un csv

import csv
import requests
from bs4 import BeautifulSoup

def make_soup(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'html.parser') return soup

url = "https://www.pc-kombo.com/us/components/cpus" soup =
make_soup(url)

procesadores = []
for i in soup.find("ol", {"id": "hardware"}).find_all('li'): if(i != 'None' and i != -1):
    nombre = i.find("h5", {"class": "name"}) subtitle = i.find("div", {"class":
    'subtitle'}) turbo = ""
    clock = ""
    threads = ""
    cores = ""
    price = ""
    socket = None # Inicializar la variable socket # buscar el valor del socket
    socket_html = subtitle.find("span", {"class": 'socket'}) if socket_html is not None:
        socket = socket_html.text.strip()
    for atributo in subtitle.find_all('span'): if atributo.text.startswith('Turbo'):
        turbo = atributo.text.split(' ')[1] elif atributo.text.startswith('Clock'):
            clock = atributo.text.split(' ')[1] elif atributo.text.endswith('Threads'):
                threads = atributo.text.split(' ')[0]
            elif atributo.text.endswith('Cores'):
```

```

cores = atributo.text.split(' ')[0]
# buscar el precio en el HTML
precio_html = i.find("span", {'class': 'price'}) if precio_html is not None:
# obtener el valor del precio y asignarlo a la variable price
price = precio_html.text.strip()[4:]

if price != "":
url_cpu = (i.find('a', href=True)['href']).split(" ")[0] response = requests.get(url_cpu)

soup = BeautifulSoup(response.text, 'html.parser')
# Obtener el productor
producer = soup.find("dd",{'itemprop':'brand'}) if producer is not None:
producer = producer.text

# Obtener el performance level
perf_level = soup.find('p', {'itemprop': 'aggregateRating'})
if perf_level is not None:
perf_level = perf_level.text
perf_level_pattern = re.compile(r'(\d+\.\d+\d+)') perf_level_match =
perf_level_pattern.search(perf_level)

if perf_level_match is not None:
perf_level = perf_level_match.group(1) perf_level = perf_level[:-3]

procesador = {
'nombre': nombre.get_text(strip=True), 'socket': socket,
'turbo': turbo,
'clock': clock,
'threads': threads,
'cores': cores,
'perf_level' : perf_level,
'producer': producer,
'precio': price
}

if turbo != " and clock != ":

procesadores.append(procesador)
# Escribir los datos en un archivo CSV
with open('procesadores.csv', mode='w', newline='', encoding='utf-8') as outfile:
writer = csv.writer(outfile)
# Escribir la cabecera del archivo CSV
writer.writerow(['Nombre', 'Socket', 'Turbo', 'Clock', 'Threads', 'Cores','Performance Level
/10','Producer', 'Precio']) # Escribir cada fila en el archivo CSV
for procesador in procesadores:
writer.writerow([procesador['nombre'], procesador['socket'], procesador['turbo'],
procesador['clock'], procesador['threads'], procesador['cores'],
procesador['perf_level'],procesador['producer'],

```

```
procesador['precio'])])
```

GPUS

```
# exporta los datos necesarios de las gpus
import csv
import requests
import re
from bs4 import BeautifulSoup

def make_soup(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup

url = "https://www.pc-kombo.com/us/components/gpus"
soup = make_soup(url)

gpus = []
for i in soup.find("ol", {"id": "hardware"}).find_all('li'):
    if(i != 'None' and i != -1):
        nombre = i.find("h5", {"class": "name"})
        subtitle = i.find("div", {"class": "subtitle"}).find_all('span')
        serie = ""
        vram = ""
        potencia = ""
        serie = i.find("span", {"class": "series"}).text
        vram = i.find("span", {"class": "vram"}).text[-2]
        for atributo in subtitle:
            if atributo.text.endswith('W'):
                potencia = atributo.text[-1]

# buscar el precio en el HTML
precio_html = i.find("span", {"class": "price"}) if precio_html is not None:
# obtener el valor del precio y asignarlo a la variable price
price = precio_html.text.strip()[4:]

if price != "":
    url_ram = (i.find('a', href=True)['href']).split(" ")[0]
    response = requests.get(url_ram)

    soup = BeautifulSoup(response.text, 'html.parser')
    # Obtener el productor
    producer = soup.find("dd", {"itemprop": "brand"}) if producer is not None:
        producer = producer.text

#obtener el length
length = soup.find('dt', text='Length')
if length is not None:
    length = length.find_next_sibling('dd').text[-2]
```

```

#obtener los slots
slots = soup.find('dt', text='Slots')
if slots is not None:
    slots = slots.find_next_sibling('dd').text
#obtener los 8 pin
eight_pin = soup.find('dt', text='8-pin connectors') if eight_pin is not None:
eight_pin = eight_pin.find_next_sibling('dd').text else:
eight_pin = 0

#obtener los 6 pin
six_pin = soup.find('dt', text='6-pin connectors') if six_pin is not None:
six_pin = six_pin.find_next_sibling('dd').text
#obtener el boost clock
boost_clock = soup.find('dt', text='Boost Clock') if boost_clock is not None:
boost_clock =
boost_clock.find_next_sibling('dd').text[:-3]

#obtener memory clock
memory_clock = soup.find('dt', text='Memory Clock') if memory_clock is not None:
memory_clock =
memory_clock.find_next_sibling('dd').text[:-3]

# Obtener el performance level
perf_level = soup.find('meta', {'itemprop': 'ratingValue'})
if perf_level is not None:
    perf_level = perf_level.text
    perf_level_pattern = re.compile(r'(\d+\.\d+/\d+)') perf_level_match =
    perf_level_pattern.search(perf_level)

    if perf_level_match is not None:
        perf_level = perf_level_match.group(1) perf_level = perf_level[:-3]

gpu = {
    'nombre': nombre.get_text(strip=True),
    'serie': serie,
    'vram': vram,
    'potencia': potencia,
    'length': length,
    'slots': slots,
    'eight_pin': eight_pin,
    'six_pin': six_pin,
    'boost_clock': boost_clock,
    'memory_clock': memory_clock,
    'producer': producer,
    'perf_level': perf_level,
    'precio': price
}
gpus.append(gpu)

```

```

# Escribir los datos en un archivo CSV
with open('gpus.csv', mode='w', newline='', encoding='utf-8') as outfile:
    writer = csv.writer(outfile)
# Escribir la cabecera del archivo CSV
writer.writerow(['Name', 'Serie', 'VRAM', 'TDP (W)', 'Length (mm)', 'slots', '8 Pin', '6 Pin',
    'Boost Clock', 'Memory Clock', 'Producer', 'Performance Level /10', 'Price'])
# Escribir cada fila en el archivo CSV
for gpu in gpus:
    writer.writerow([gpu['nombre'], gpu['serie'], gpu['vram'], gpu['potencia'], gpu['length'],
        gpu['slots'], gpu['eight_pin'], gpu['six_pin'],
        gpu['boost_clock'], gpu['memory_clock'], gpu['producer'], gpu['perf_level'], gpu['precio']])

```

MOTHERBOARD

#exporta los datos necesarios de las motherboards

```

import csv
import requests
from bs4 import BeautifulSoup

def make_soup(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup

def scrape_motherboards():
    url = "https://www.pc-kombo.com/us/components/motherboards"  soup =
make_soup(url)

    motherboards = []
    for i in soup.find("ol", {'id': 'hardware'}).find_all('li'): if(i != 'None' and i != -1):
        nombre = i.find("h5", {'class': 'name'}) size = i.find("span", {'class': 'size'}).text socket =
i.find("span", {'class': 'socket'}).text chipset = i.find("span", {'class': 'chipset'}).text
        ramslots = i.find("span", {'class': 'ramslots'}).text # buscar el precio en el HTML
        precio_html = i.find("span", {'class': 'price'}) if precio_html is not None:
            # obtener el valor del precio y asignarlo a la variable precio
            precio = precio_html.text.strip()[4:]
        if precio != "":
            url_motherboard = (i.find('a', href=True) ['href']).split(" ")[0]
            response = requests.get(url_motherboard)  soup = BeautifulSoup(response.text,
                'html.parser')
            """ # Obtener el form factor
            form_factor = soup.find('dt', text='Form Factor') if form_factor is not None:
                form_factor =
                form_factor.find_next_sibling('dd').text"""
            # Obtener el tipo de memoria y su capacidad memory_type = soup.find('dt',
            text='Memory Type') if memory_type is not None:

```

```

memory_type =
memory_type.find_next_sibling('dd').text

memory_capacity = soup.find('dt', text='Memory Capacity')
if memory_capacity is not None:
memory_capacity =
memory_capacity.find_next_sibling('dd').text

# Obtener el productor
producer = soup.find("dd",{' itemprop': 'brand'}).text

motherboard = {
'nombre': nombre.get_text(strip=True), 'size': size,
'socket': socket,
'chipset': chipset,
'ramslots': ramslots,
'memory_type': memory_type,
'memory_capacity': memory_capacity,
'producer': producer,
'precio': precio
}

motherboards.append(motherboard)
# Escribir los datos en un archivo CSV
with open('motherboards.csv', mode='w', newline='', encoding='utf 8') as outfile:
writer = csv.writer(outfile)
# Escribir la cabecera del archivo CSV
writer.writerow(['Nombre', 'Size', 'Socket', 'Chipset', 'Ramslots', 'Memory_Type',
'Memory_Capacity', 'Producer', 'Precio']) # Escribir cada fila en el archivo CSV
for motherboard in motherboards:

writer.writerow([motherboard['nombre'], motherboard['size'], motherboard['socket'],
motherboard['chipset'], motherboard['ramslots'], motherboard['memory_type'],
motherboard['memory_capacity'], motherboard['producer'], motherboard['precio']])

scrape_motherboards()

```

RAM

#exporta los datos necesarios de las ram

```

import csv
import requests
from bs4 import BeautifulSoup

def make_soup(url):
res = requests.get(url)

```

```

res.raise_for_status()
soup = BeautifulSoup(res.text, 'html.parser')
return soup

def scrape_rams():
url = "https://www.pc-kombo.com/us/components/rams" soup =
make_soup(url)

rams = []
for i in soup.find("ol", {"id": 'hardware'}).find_all('li'): if(i != 'None' and i != -1):
nombre = i.find("h5", {"class": 'name'})
size = i.find("span", {"class": 'size'}).text[:-2] types = i.find("span", {"class": 'type'}).text

# buscar el precio en el HTML
precio_html = i.find("span", {"class": 'price'}) if precio_html is not None:
# obtener el valor del precio y asignarlo a la variable precio
precio = precio_html.text.strip()[4:]

if precio != "":
url_ram = (i.find('a', href=True)['href']).split(" ") [0]
response = requests.get(url_ram)

soup = BeautifulSoup(response.text, 'html.parser')

#obtener el clock
clock = soup.find('dt', text='Clock') if clock is not None:
clock = clock.find_next_sibling('dd').text
#obtener los sticks
sticks = soup.find('dt', text='Sticks') if sticks is not None:
sticks = sticks.find_next_sibling('dd').text
# Obtener el productor
producer = soup.find("dd", {"itemprop":'brand'}).text

ram = {
'nombre': nombre.get_text(strip=True), 'size': size,
'type': types,
'clock': clock,
'sticks': sticks,
'producer': producer,
'precio': precio
}
rams.append(ram)

# Escribir los datos en un archivo CSV
with open('rams.csv', mode='w', newline='', encoding='utf-8') as outfile:
writer = csv.writer(outfile)
# Escribir la cabecera del archivo CSV

```

```

writer.writerow(['Name', 'Size', 'Type', 'Clock', 'Sticks', 'Producer', 'Price'])
# Escribir cada fila en el archivo CSV
for ram in rams:

writer.writerow([ram['nombre'],ram['size'],ram['type'],ram['clock'],ram['sticks'],
,ram['producer'],ram['precio'])]

scrape_rams()

```

HDD

#exporta los datos necesarios de las HDD

```

import csv
import requests
from bs4 import BeautifulSoup

def make_soup(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup

def scrape_hdds():
    url = "https://www.pc-kombo.com/us/components/hdds"  soup =
make_soup(url)

    hdds = []
    for i in soup.find("ol", {'id': 'hardware'}).find_all('li'): if(i != 'None' and i != -1):
        nombre = i.find("h5", {'class': 'name'}) size = i.find("span", {'class': 'size'}).text[:-2]
        rpm = i.find("span", {'class': 'rpm'}).text

        # buscar el precio en el HTML
        precio_html = i.find("span", {'class': 'price'}) if precio_html is not None:
            # obtener el valor del precio y asignarlo a la variable precio
            precio = precio_html.text.strip()[4:]
            if precio != "":
                url_hdd = (i.find('a', href=True)['href']).split(" ") [0]
                response = requests.get(url_hdd)
                soup = BeautifulSoup(response.text, 'html.parser')

        # Obtener el productor
        producer = soup.find("dd",{'itemprop':'brand'}).text

    hdd = {

```

```

'nombre': nombre.get_text(strip=True),
'size' :size,
        'rpm' : rpm,
        'producer': producer,
        'precio': precio
}

hdds.append(hdd)

# Escribir los datos en un archivo CSV
with open('hdds.csv', mode='w', newline='', encoding='utf-8') as outfile:
writer = csv.writer(outfile)
# Escribir la cabecera del archivo CSV
writer.writerow(['Name', 'Size', 'Rpm', 'Producer', 'Price']) # Escribir cada fila en el archivo CSV
for hdd in hdds:
writer.writerow([hdd['nombre'],hdd['size'],hdd['rpm'] ,hdd['producer'],hdd['precio']])
scrape_hdds()

```

SSD

```

#exporta los datos necesarios de las SSD

import csv
import requests
from bs4 import BeautifulSoup

def make_soup(url):
res = requests.get(url)
res.raise_for_status()
soup = BeautifulSoup(res.text, 'html.parser')
return soup

def scrape_ssds():
url = "https://www.pc-kombo.com/us/components/ssds" soup =
make_soup(url)

ssds = []
for i in soup.find("ol", {"id": 'hardware'}).find_all('li'): if(i != 'None' and i != -1):
nombre = i.find("h5", {'class': 'name'}) size = i.find("span", {'class': 'size'}).text[:-2]

# buscar el precio en el HTML
precio_html = i.find("span", {'class': 'price'}) if precio_html is not None:
# obtener el valor del precio y asignarlo a la variable precio
precio = precio_html.text.strip()[4:]
if precio != "":

```

```

url_hdd = (i.find('a', href=True)['href']).split(" ") [0]
response = requests.get(url_hdd)
soup = BeautifulSoup(response.text, 'html.parser')
#obtener el form factor
form_factor = soup.find('dt', text='Form Factor') if form_factor is not None:
form_factor =
form_factor.find_next_sibling('dd').text

#obtener el protocolo
protocolo = soup.find('dt', text='Protocol') if protocolo is not None:
    protocolo = protocolo.find_next_sibling('dd').text

# Obtener el productor
producer = soup.find("dd",{'itemprop':'brand'}).text

ssd = {
'nombre': nombre.get_text(strip=True), 'size':size,
'form_factor':form_factor,
'protocolo':protocolo,
'producer': producer,
'precio': precio
}
ssds.append(ssd)

# Escribir los datos en un archivo CSV
with open('ssds.csv', mode='w', newline='', encoding='utf-8') as outfile:
writer = csv.writer(outfile)
# Escribir la cabecera del archivo CSV
writer.writerow(['Name', 'Size', 'Form Factor','Protocol', 'Producer', 'Price'])
# Escribir cada fila en el archivo CSV
for ssd in ssds:
    writer.writerow([ssd['nombre'],ssd['size'],ssd['form_factor'],ssd['proto
tocolo'],
,ssd['producer'],ssd['precio']])
scrape_ssds()

```

POWER SUPPLY

#exporta los datos necesarios de las Power Supplies

```

import csv
import requests

```

```

from bs4 import BeautifulSoup

def make_soup(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup

def scrape_powerSupply():
    url = "https://www.pc-kombo.com/us/components/psus"  soup =
make_soup(url)

    pss = []
    for i in soup.find("ol", {"id": "hardware"}).find_all('li'): if(i != 'None' and i != -1):
        nombre = i.find("h5", {"class": "name"})
        size = i.find("span", {"class": "size"}).text[:-2]  watts = i.find("span", {"class":
'watt'}).text[:-1]

        # buscar el precio en el HTML
        precio_html = i.find("span", {"class": "price"})  if precio_html is not None:
            # obtener el valor del precio y asignarlo a la variable precio
            precio = precio_html.text.strip()[4:]
            if precio != "":
                url_hdd = (i.find('a', href=True)['href']).split(" ") [0]
                response = requests.get(url_hdd)
                soup = BeautifulSoup(response.text, 'html.parser')
                #obtener el efficiency
                efficiency = soup.find('dt', text='Efficiency Rating')  if efficiency is not None:
                    efficiency =
                    efficiency.find_next_sibling('dd').text

                #obtener los 8 pin
                eight_pin = soup.find('dt', text='PCI-E cables 8-pin')  if eight_pin is not None:
                    eight_pin = eight_pin.find_next_sibling('dd').text  else:
                    eight_pin = 0

                #obtener los 6 pin
                six_pin = soup.find('dt', text='PCI-E cables 6-pin')  if six_pin is not None:
                    six_pin = six_pin.find_next_sibling('dd').text  else:
                    six_pin = 0

                # Obtener el productor
                producer = soup.find("dd", {"itemprop": "brand"} ).text

                ps = {
                    'nombre': nombre.get_text(strip=True), 'size': size,
                    'watts': watts,
                    'efficiency': efficiency,

```

```

        'eight_pin': eight_pin,
        'six_pin': six_pin,
        'producer': producer,
        'precio': precio
    }

pss.append(ps)

# Escribir los datos en un archivo CSV
with open('powerSupplies.csv', mode='w', newline='', encoding='utf-8') as
outfile:
    writer = csv.writer(outfile)
# Escribir la cabecera del archivo CSV
writer.writerow(['Name', 'Size', 'Watts', 'Efficiency Rating', '8 Pin', '6 Pin', 'Producer',
'Price'])
# Escribir cada fila en el archivo CSV
for ps in pss:

    writer.writerow([ps['nombre'], ps['size'], ps['watts'], ps['efficiency'],
    ps['eight_pin'], ps['six_pin'], ps['producer'], ps['precio']])
scrape_ssds()
#exporta los datos necesarios de los coolers

import csv
import requests
from bs4 import BeautifulSoup

def make_soup(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup

def scrape_coolers():
    url = "https://www.pc-kombo.com/us/components/cpucoolers"  soup =
make_soup(url)

    coolers = []
    for i in soup.find("ol", {'id': 'hardware'}).find_all('li'): if(i != 'None' and i != -1):
        nombre = i.find("h5", {'class': 'name'})

# buscar el precio en el HTML
    precio_html = i.find("span", {'class': 'price'}) if precio_html is not None:
# obtener el valor del precio y asignarlo a la variable precio
        precio = precio_html.text.strip()[4:]
        if precio != "":
            url_hdd = (i.find('a', href=True)['href']).split(" ") [0]
            response = requests.get(url_hdd)

```

```

soup = BeautifulSoup(response.text, 'html.parser')
#obtener los sockets
sockets = soup.find('dt', text='Supported Sockets') if sockets is not None:
    sockets =
        sockets.find_next_sibling('dd').text[11:]

#obtener el height
height = soup.find('dt', text='Height') if height is not None:
    height = height.find_next_sibling('dd').text[:-2]
#obtener el tdp
tdp = soup.find('dt', text='TDP')
if tdp is not None:
    tdp = tdp.find_next_sibling('dd').text[:-2]

# Obtener el productor
producer = soup.find("dd",{'itemprop':'brand'}).text

cooler = {
    'nombre': nombre.get_text(strip=True), 'sockets': sockets,
    'height':height,
    'tdp':tdp,
    'producer': producer,
    'precio': precio
}
coolers.append(cooler)

# Escribir los datos en un archivo CSV
with open('coolers.csv', mode='w', newline='', encoding='utf-8') as outfile:
    writer = csv.writer(outfile)
    # Escribir la cabecera del archivo CSV
    writer.writerow(['Name', 'Sockets', 'Height', 'TDP', 'Producer', 'Price'])
    # Escribir cada fila en el archivo CSV
    for cooler in coolers:

        writer.writerow([cooler['nombre'],cooler['sockets'],cooler['height'],cooler['tdp'],
                        cooler['producer'],cooler['precio']])
scrape_powerSupply()

```

CASES

```
#exporta los datos necesarios de las cases
```

```

import csv
import requests
from bs4 import BeautifulSoup

def make_soup(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup

def scrape_cases():
    url = "https://www.pc-kombo.com/us/components/cases"  soup =
make_soup(url)

    cases = []
    for i in soup.find("ol", {'id': 'hardware'}).find_all('li'): if(i != 'None' and i != -1):
        nombre = i.find("h5", {'class': 'name'})

        # buscar el precio en el HTML
        precio_html = i.find("span", {'class': 'price'}) if precio_html is not None:
            # obtener el valor del precio y asignarlo a la variable precio
            precio = precio_html.text.strip()[4:]
            if precio != "":
                url_hdd = (i.find('a', href=True)['href']).split(" ") [0]
                response = requests.get(url_hdd)
                soup = BeautifulSoup(response.text, 'html.parser')
                #obtener el width
                width = soup.find('dt', text='Width') if width is not None:
                    width = width.find_next_sibling('dd').text[:-2]
                #obtener el Depth
                depth = soup.find('dt', text='Depth') if depth is not None:
                    depth = depth.find_next_sibling('dd').text[:-2]
                #obtener el height
                height = soup.find('dt', text='Height') if height is not None:
                    height = height.find_next_sibling('dd').text[:-2]
                #obtener la máxima length de gpu prmitida  max_gpu_length = soup.find('dt', text='Supported
GPU length')
                if max_gpu_length is not None:
                    max_gpu_length =
                    max_gpu_length.find_next_sibling('dd').text[:-2]

        # Obtener el productor
        producer = soup.find("dd",{'itemprop':'brand'}).text

    case = {
        'nombre': nombre.get_text(strip=True), 'width':width,
        'depth':height,

```

```

        'height':height,
        'max_gpu_length': max_gpu_length,
        'producer': producer,
        'precio': precio
    }

cases.append(case)

# Escribir los datos en un archivo CSV
with open('cases.csv', mode='w', newline='', encoding='utf-8') as outfile:
writer = csv.writer(outfile)
# Escribir la cabecera del archivo CSV
writer.writerow(['Name', 'Width mm', 'Depth mm','Height mm',' Maximum GPU
Length','Producer', 'Price'])
# Escribir cada fila en el archivo CSV
for case in cases:

writer.writerow([case['nombre'],case['width'],case['depth'],case['heig ht'],
case['max_gpu_length'],case['producer'],case['precio']])
scrape_cases()

```

JUEGOS STEAM

```

#exporta los datos necesarios de las cases

import csv
import requests
from bs4 import BeautifulSoup

def make_soup(url):
cookies = {'birthtime': '568022401'}
res = requests.get(url, cookies=cookies)
res.raise_for_status()
soup = BeautifulSoup(res.text, 'html.parser')
return soup

# Juegos más
def scrape_games():
#not_games_set =
{"https://store.steampowered.com/app/1675200/Steam_Deck/?
snr=1_7_7_7000_150_1"}
url = "https://store.steampowered.com/search/?filter=topsellers" soup = make_soup(url)

a = soup.find('div', {'id':
'search_result_container'}).find('div', {'id': 'search_resultsRows'})
```

```

for i in a.find_all('a', href=True):
if not 'Steam_Deck' in str(i):

print(i['href'])
title = str(i['href']).split('/')[-2].replace('_', ' ') print("-----Mínimos-----")

print(title + ",", end="")

soupa = make_soup(i['href'])

req = soupa.find('div', {'class':
'sysreq_contents'}).find('ul').find_all('li')

for i in req:
val = i.text.split(" ")
if val[0] in ['Processor:', 'Graphics:', 'Memory:', 'Storage:']:
for i in val[1:]:
if i == 'or' or i == "/" or i == "|": break
else:
print(i, end = "")
print(", ", end="")

print("\n")
recomended = soupa.find('div', {'class': 'sysreq_contents'}).find('div',
{'class':'game_area_sys_req_rightCol'})
if recommended:
print("-----Recomendados-----") for i in recommended.find_all('li'):
print(i.text)
print("\n\n")

```

9.2. ARCHIVO DE QUERY MYSQL

```
SET @program = 'Cyberpunk';
SET @gpu_base = (SELECT performance FROM gpu WHERE serie LIKE CONCAT('%', (SELECT rgpu FROM program WHERE name = @program), '%') LIMIT 1);
SET @cpu_base = (SELECT performances FROM cpu WHERE name LIKE CONCAT('%', (SELECT rcpu FROM program WHERE name = @program), '%') ORDER BY performances LIMIT 1);
SET @ram = (SELECT rram FROM program WHERE name = @program);
-- SSD en GB
SET @ssd = (SELECT rstorage FROM program WHERE name = @program);
-- HDD en TB
SET @hdd = (SELECT rstorage FROM program WHERE name = @program);

SELECT q3.Cpu AS CPU, q3.Cooler, q3.Motherboard, q3.Ram AS RAM, q2.Gpu AS GPU,
q2.Power AS Power_Supply,
q1.CASE_ AS Case_, q4.Ssd AS SSD, q5.Hdd AS HDD,
q1.Price + q2.Price + q3.Price + q4.Price + q5.Price AS Price

FROM (SELECT DISTINCT C.name AS CASE_, C.width AS width, C.height AS height,
C.depth AS depth, C.price AS Price FROM mipc_test.case AS C
JOIN(SELECT DISTINCT width, depth, height, MIN(price) AS price FROM mipc_test.case
GROUP BY width, depth, height) q1
WHERE C.height = q1.height AND C.width = q1.width AND C.depth = q1.depth AND C.price = q1.price) q1

JOIN(SELECT DISTINCT GPU.name AS Gpu, POWER.name AS Power, GPU.length,
POWER.width,
POWER.height, POWER.depth, GPU.price + POWER.price AS Price
FROM gpu GPU
JOIN power POWER ON POWER.`6pin` >= GPU.`6pin` AND POWER.`8pin` >= GPU.`8pin`
JOIN(SELECT GPU.length, POWER.width, POWER.height, POWER.depth, MIN(GPU.price + POWER.price) AS Price
FROM gpu GPU
JOIN power POWER ON POWER.`6pin` >= GPU.`6pin` AND POWER.`8pin` >= GPU.`8pin`
WHERE GPU.performance >= @gpu_base
GROUP BY GPU.length, POWER.width, POWER.height, POWER.depth) q1
ON GPU.length = q1.length AND POWER.width = q1.width AND POWER.height = q1.height
AND
POWER.depth = q1.depth AND GPU.price + POWER.price = q1.Price AND GPU.performance
>= @gpu_base) q2
ON q2.length < q1.depth AND q2.width < q1.width AND q2.height < q1.height AND q2.depth <
q1.depth

JOIN(SELECT q1.Cpu, q1.Cooler, q2.Motherboard, q2.Ram, q1.COOLER_Height,
q2.MB_Height,
q2.MB_Width, q1.Price + q2.Price AS Price
FROM
(SELECT DISTINCT CPU.name AS Cpu, CPU.socket AS Socket, COOLER.name AS Cooler,
COOLER.height AS COOLER_Height, CPU.price + COOLER.price AS Price
```

```

FROM CPU cpu
JOIN cooler COOLER ON COOLER.sockets LIKE CONCAT('%',CPU.socket,'%')
JOIN
(SELECT CPU.socket,COOLER.height,MIN(CPU.price + COOLER.price) AS price
FROM cpu CPU
JOIN cooler COOLER ON COOLER.sockets LIKE CONCAT('%',CPU.socket,'%')
WHERE CPU.performances >= @cpu_base
GROUP BY CPU.socket,COOLER.height) q1
WHERE CPU.socket = q1.socket AND COOLER.height = q1.height AND CPU.price +
COOLER.price = q1.price AND CPU.performances >= @cpu_base) q1

JOIN (SELECT DISTINCT MB.name AS Motherboard, MB.height AS MB_Height,
MB.width AS MB_Width, MB.socket AS Socket, RAM.name AS Ram, (MB.price + RAM.price)
AS Price
FROM motherboard MB
JOIN ram RAM ON RAM.size >= @ram AND RAM.type LIKE
CONCAT('%',MB.memorytype,'%') AND MB.ramslots >= RAM.sticks
JOIN (SELECT MB.socket,MB.width,MB.height, MIN(MB.price + RAM.price) AS price
FROM motherboard MB
JOIN ram RAM ON RAM.size >= @ram AND RAM.type LIKE CONCAT('%',MB.memorytype,'%')
AND MB.ramslots >= RAM.sticks
GROUP BY MB.socket,MB.width,MB.height) q1
WHERE MB.socket = q1.socket AND MB.width = q1.width AND MB.height = q1.height
AND (MB.price + RAM.price) = q1.price AND RAM.size >= @ram) q2 ON q1.Socket =
q2.Socket) q3
ON q3.MB_Height < q1.height AND q3.MB_Width < q1.depth AND COOLER_Height <
q1.height

JOIN (SELECT name AS Ssd, price AS Price
FROM ssd
WHERE size >= @ssd
ORDER BY price
LIMIT 1) q4

JOIN(SELECT name AS Hdd, price AS Price
FROM hdd
WHERE size*1000 >= @hdd
ORDER BY price
LIMIT 1) q5

ORDER BY Price
LIMIT 1000

```