



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería en Informática



TFG del Grado en Ingeniería Informática

**Extracción y procesamiento de  
datos de Amazon para su  
utilización en un estudio de  
marketing.**



Presentado por Daniel Arnaiz Gutierrez  
en Universidad de Burgos — 2 de julio de 2019  
Tutores: Dr. José Francisco Díez Pastor  
Dr. César Ignacio García Osorio

---

# Índice general

---

<b>Índice general</b>	<b>I</b>
<b>Índice de figuras</b>	<b>III</b>
<b>Índice de tablas</b>	<b>V</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	7
<b>Apéndice B Especificación de Requisitos</b>	<b>11</b>
B.1. Introducción . . . . .	11
B.2. Objetivos generales . . . . .	11
B.3. Catalogo de requisitos . . . . .	12
B.4. Especificación de requisitos . . . . .	13
<b>Apéndice C Especificación de diseño</b>	<b>20</b>
C.1. Introducción . . . . .	20
C.2. Diseño de datos . . . . .	20
C.3. Diseño procedimental . . . . .	24
C.4. Diseño arquitectónico . . . . .	26
<b>Apéndice D Documentación técnica de programación</b>	<b>28</b>
D.1. Introducción . . . . .	28
D.2. Estructura de directorios . . . . .	28
D.3. Manual del programador . . . . .	30
D.4. Compilación, instalación y ejecución del proyecto . . . . .	34
<b>Apéndice E Documentación de usuario</b>	<b>35</b>

*ÍNDICE GENERAL*

II

E.1. Introducción . . . . .	35
E.2. Requisitos de usuarios . . . . .	35
E.3. Instalación . . . . .	36
E.4. Manual del usuario . . . . .	38
<b>Bibliografía</b>	<b>45</b>

---

# Índice de figuras

---

A.1. Gráfico <i>burndown</i> del <i>sprint</i> 1 . . . . .	2
A.2. Gráfico <i>burndown</i> del <i>sprint</i> 2 . . . . .	3
A.3. Gráfico <i>burndown</i> del <i>sprint</i> 3 . . . . .	4
A.4. Gráfico <i>burndown</i> del <i>sprint</i> 4 . . . . .	5
A.5. Gráfico <i>burndown</i> del <i>sprint</i> 5 . . . . .	6
A.6. Gráfico <i>burndown</i> del <i>sprint</i> 6 . . . . .	7
B.1. Diagrama general de los casos de uso . . . . .	14
B.2. Diagrama desglosado de los casos de uso del usuario . . . . .	14
C.1. Estructura del JSON con los enlaces a los productos . . . . .	21
C.2. Estructura del JSON con los campos de cada producto . . . . .	21
C.3. Estructura del JSON con los comentarios de cada producto . . . . .	22
C.4. Diagrama de la base de datos utilizada. . . . .	23
C.5. Proceso de <i>web scraping</i> del proyecto . . . . .	24
C.6. Arquitectura que sigue Scrapy y sus diferentes componentes . . . . .	27
D.1. Tipo de <i>dataset</i> a crear en Dataturks . . . . .	31
D.2. Creación de un <i>dataset</i> en Dataturks . . . . .	32
D.3. Proceso de etiquetado manual en Dataturks . . . . .	33
E.1. Descarga de Python . . . . .	36
E.2. Descarga de Anaconda . . . . .	37
E.3. Descarga del contenido del repositorio . . . . .	38
E.4. Página que contiene los productos a extraer . . . . .	39
E.5. URLs introducidos en la <i>spider</i> encargada de extraer los enlaces de cada producto . . . . .	40
E.6. <i>urls_spider</i> ejecutada y en funcionamiento . . . . .	40
E.7. JSON que contiene los enlaces extraídos. . . . .	41
E.8. Campos de un producto extraído durante la ejecución de la <i>spider</i>	41

E.9. Comentarios de un producto extraídos durante la ejecución de la <i>spider</i> . . . . .	42
E.10. Subir <i>notebook</i> a Google Colab . . . . .	43
E.11. <i>Notebook</i> de Google Colab abierto en el navegador web . . . . .	43
E.12. Botón de ejecución de Google Colab . . . . .	44

---

# Índice de tablas

---

A.1. Costes de personal . . . . .	8
A.2. Costes de hardware . . . . .	9
A.3. Costes totales del proyecto . . . . .	9
A.4. Licencias utilizadas . . . . .	10
B.1. Caso de uso 1: Extraer información de los productos. . . . .	15
B.2. Caso de uso 2: Almacenamiento en base de datos. . . . .	16
B.3. Caso de uso 3: Generar archivo JSON. . . . .	16
B.4. Caso de uso 4: Entrenar un clasificador de imágenes. . . . .	17
B.5. Caso de uso 5: Clasificar automáticamente las imágenes. . . . .	18
B.6. Caso de uso 6: Generar documento Excel. . . . .	19

## *Apéndice A*

---

# **Plan de Proyecto Software**

---

### **A.1. Introducción**

La planificación de un proyecto es quizás la fase mas importante de éste. Una buena planificación consigue que el desarrollo del proyecto siga su curso con normalidad y con el mínimo número de imprevistos. A la hora de planificar el proyecto, son varias las partes a tener en cuenta: tiempo, trabajo, y dinero.

A continuación se tratarán los detalles del plan de proyecto llevado a cabo en dos secciones:

**Planificación temporal:** En este apartado se analizará y explicará cómo se ha planificado el tiempo durante el desarrollo del proyecto teniendo en cuenta el trabajo necesario para cada parte.

**Estudio de viabilidad:** En este apartado se estudiará la viabilidad del proyecto en distintos campos:

**Viabilidad económica:** Estimación de los costes y beneficios del proyecto.

**Viabilidad legal:** Análisis sobre los conceptos legales del proyecto, esto engloba las licencias y políticas utilizadas.

### **A.2. Planificación temporal**

Antes de empezar con el análisis sobre la planificación temporal es necesario añadir que se ha seguido una metodología ágil para la realización del proyecto, más en concreto se ha seguido el método *Scrum*. Es por esto que el desarrollo del proyecto se ha dividido en diferentes *sprints*, con reuniones

entre todos los integrantes del proyecto para tratar los cambios y las siguientes tareas a realizar.

A continuación se mostrarán las principales características de cada *sprint* junto con un gráfico *burndown* generado con la ayuda de la extensión ZenHub y GitHub.

### **Sprint 1 (18/11/2018 - 24/3/2019)**

El primer *sprint* ha resultado ser el más largo. Esto se debe a que el desarrollo del proyecto comenzó mas tarde de lo que en un principio se había planeado. Como se puede ver en el gráfico *burndown* a continuación, en el último tramo del *sprint* es cuando de verdad las tareas propuestas comienzan a darse por completadas.

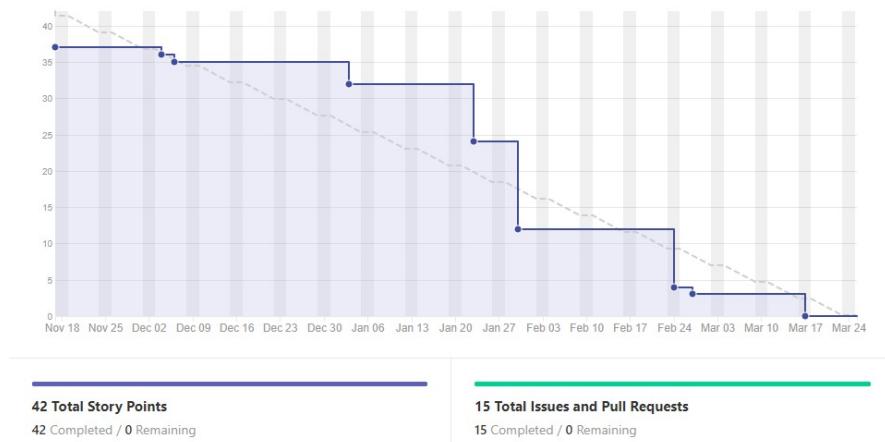


Figura A.1: Gráfico *burndown* del *sprint* 1

Este primer *sprint* se centra principalmente en la puesta en marcha de las principales herramientas a utilizar, además de la toma de algunas decisiones como pueden ser la elección de qué *web scraper* y etiquetador de imágenes utilizar.

Debido a la extensa duración de este *sprint*, al finalizar ya encontramos algunas tareas más avanzadas completadas:

- Primeras versiones de *spiders* funcionales.
- Etiquetado manual de un número reducido de imágenes.
- Extracción de los primeros productos con sus respectivos campos.

### **Sprint 2 (17/3/2018 - 12/4/2019)**

Este *sprint* cuenta con una duración mucho menor al anterior. Es aquí donde comienza la familiarización con L<sup>A</sup>T<sub>E</sub>X y las primeras mejoras al *web scraper*.

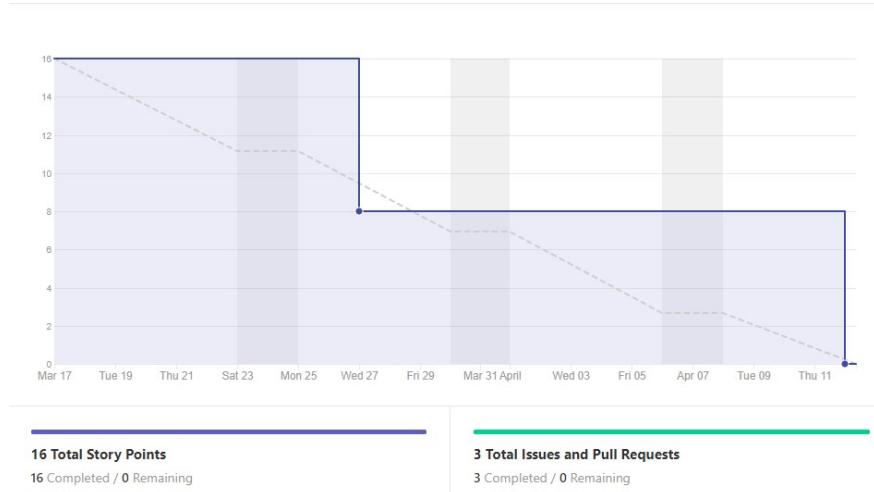


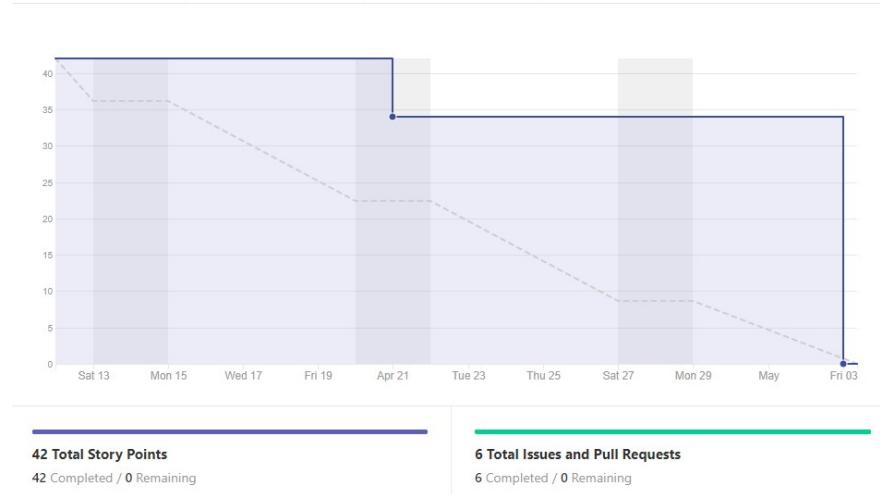
Figura A.2: Gráfico burndown del *sprint* 2

Las tareas realizadas en este *sprint* han sido las siguientes:

- Prueba con nuevas etiquetas en el clasificador manual de imágenes.
- Comenzar con la documentación del proyecto.
- Modificación de los campos a extraer en el *web scraper*.

### **Sprint 3**

Este *sprint* se centra en añadir nuevas funcionalidades al proyecto, como pueden ser el almacenamiento en la base datos o la extracción de comentarios.

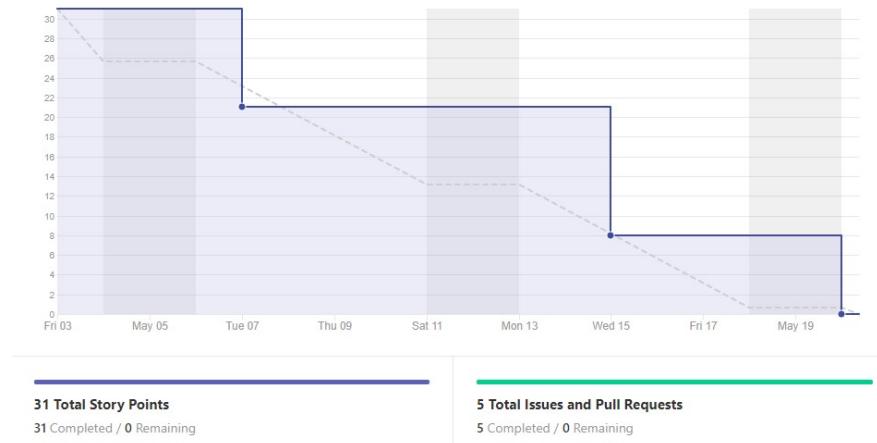
Figura A.3: Gráfico *burndown* del *sprint 3*

Las tareas realizadas en este *sprint* han sido las siguientes:

- Extracción de comentarios asociados a cada producto.
- Comenzar a hacer pruebas con el almacenamiento en una base de datos.
- Estudiar como generar un documento Excel a partir de los datos extraídos.
- Buscar información sobre como aplicar regresión lineal sobre la información extraída.

#### *Sprint 4 (3/5/2018 - 20/5/2019)*

Este *sprint* ha estado orientado sobre todo a la investigación de posibles modificaciones y mejoras.

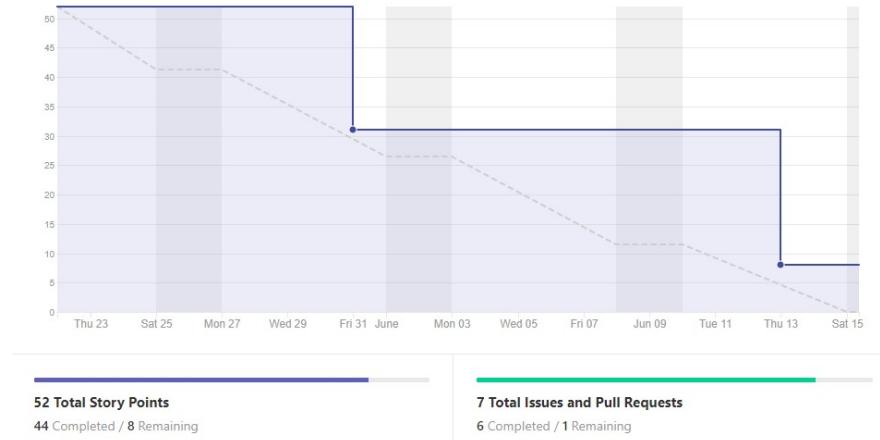
Figura A.4: Gráfico *burndown* del *sprint 4*

Las tareas realizadas en este *sprint* han sido las siguientes:

- Investigación sobre OpenCV y sus librerías de detección facial.
- Primeras iteraciones del clasificador automático de imágenes.
- Probar el funcionamiento del detector facial sobre nuestro propio *dataset*.
- Añadir los enlaces de productos para mujeres en el *web scraper*.
- Añadir un nuevo campo con el sexo al que va dirigido un artículo a la hora de extraer los productos.

### **Sprint 5 (22/5/2018 - 15/6/2019)**

*Sprint* algo más extenso que se centra principalmente en la primera versión funcional del clasificador automático y la resolución de algunos problemas encontrados.

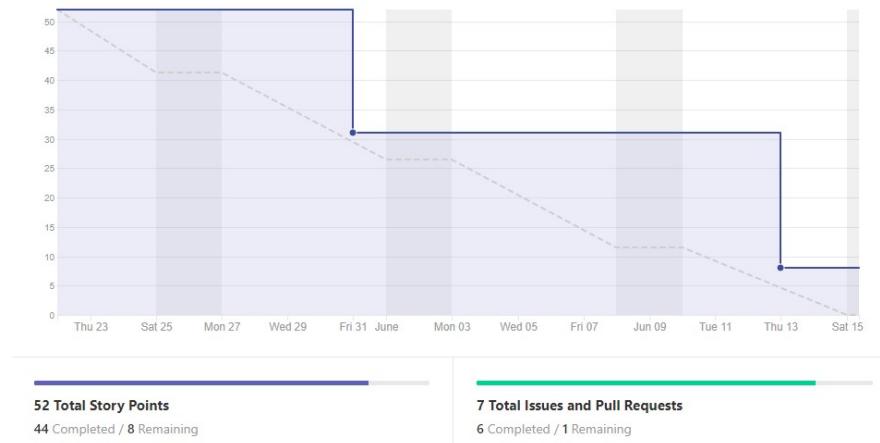
Figura A.5: Gráfico burndown del *sprint 5*

Las tareas realizadas en este *sprint* han sido las siguientes:

- Extracción de 1000 productos para la creación del *dataset* final.
- Primera versión funcional del clasificador automático Modelo/No modelo.
- Primeras pruebas con la detección de caras parcialmente visibles.
- Investigar sobre como generar un documento Excel a partir de la base de datos.

### *Sprint 6 (16/6/2018 - 1/7/2019)*

*Sprint* final del proyecto. Es aquí donde se finaliza el proyecto y se hace casi la totalidad de la documentación.

Figura A.6: Gráfico burndown del *sprint 6*

Las tareas realizadas en este *sprint* han sido las siguientes:

- Separación del campo que recoge el rango de precios de un producto en dos: precio mínimo y precio máximo.
- Revisión y actualización de los nombres de las tablas y los campos de la base de datos.
- Finalización de la documentación del proyecto y sus respectivos anexos.
- Finalización de los *notebooks* de entrenamiento y clasificación de imágenes.

### A.3. Estudio de viabilidad

#### Viabilidad económica

En este apartado se analizará los costes y beneficios estimados de este proyecto en un entorno empresarial.

#### Costes

#### Costes de personal

La totalidad de este proyecto ha sido desarrollada por una única persona a tiempo parcial durante un periodo aproximado de 5 meses. Aplicando el

salario mínimo y las cuotas de la Seguridad social<sup>1</sup>, se puede estimar un coste de personal de la siguiente forma:

Concepto	Coste (€)
Salario neto	1000 €
Retención IRPF (19 %)	360,53 €
Seguridad social (28,30 %)	537,00 €
Salario total (mensual)	1897,53 €
Total 5 meses	9.487,65 €

Tabla A.1: Costes de personal

Además de esto, se ha contado con la colaboración de dos profesores contratados durante 5 meses para la realización del proyecto. Si a cada profesor se le asigna 0,5 créditos, el coste aproximado sería el siguiente:

- Sueldo base de un ayudante doctor: 1815,61€, haciendo un coste anual por crédito de 907,5 €. El coste de este ayudante doctor sería el siguiente:

$$907,5 \text{ €} * 0,5 \text{ créditos} = 453,75 \text{ €}$$

- Sueldo base de un doctor titular: 2325,24 €, haciendo un coste anual por crédito de 1.162,62 €. El coste de este doctor titular sería el siguiente:

$$1.162,62 \text{ €} * 0,5 \text{ créditos} = 581,31 \text{ €}$$

El coste total de ambos tutores asciende a 1.035,06 €.

### Costes de material

En cuanto a los costes materiales y a nivel de programas informáticos, el único gasto es el del ordenador utilizado para la realización del proyecto. Todas las librerías utilizadas son gratuitas y no es necesario la contratación de un servidor para el funcionamiento del proyecto.

Por lo tanto, asociando un valor aproximado de 1300€ al ordenador y estimando una amortización de 5 años, el coste total de materiales para estos 5 meses es el siguiente:

$$\frac{1.300 \text{ €}}{5 \text{ años} * 12 \text{ meses}} = 21,67 \text{ €}$$

---

<sup>1</sup><http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537>

Concepto	Coste	Amortización
Ordenador	1.300 €	21,67 €
<b>Total 5 meses</b>	<b>108,34 €</b>	

Tabla A.2: Costes de hardware

**Coste total**

A continuación se muestra el coste total del proyecto:

Concepto	Coste
Coste desarrollador	9.487,65 €
Coste tutores	1.035,06 €
Material	108,34 €
<b>Total</b>	<b>10.631,05 €</b>

Tabla A.3: Costes totales del proyecto

**Beneficios**

En cuanto a los beneficios del proyecto, no se ha desarrollado teniendo como meta la generación de ingresos. Se podría comercializar cobrando una tarifa mensual a los clientes u ofreciendo un servicio personalizado a cada cliente que dependería del número de productos a extraer.

### Viabilidad legal

Para el estudio de la viabilidad del producto, se van a analizar las librerías usadas en el proyecto, anotando las licencias de las que hacen uso. A continuación se listan:

Dependencia	Versión	Licencia
Scrapy	1.5.1	BSD
Pillow	6.0.0	PIL <sup>2</sup>
requests	2.21.0	MIT
js2xml	0.3.1	MIT
Keras	2.3.1	MIT
Numpy	1.16.4	BSD
OpenCV	4.1.0	BSD
JSON	-	FREE
SQLite	3.28.0	FREE
Pandas	0.24.2	BSD

Tabla A.4: Licencias utilizadas

---

<sup>2</sup><http://www.pythontesting.net/doc/pil-license.html>

## *Apéndice B*

---

# Especificación de Requisitos

---

## B.1. Introducción

Este apartado recoge los requisitos y objetivos del proyecto. Se detallarán los objetivos generales y tanto los requisitos funcionales como los no funcionales.

## B.2. Objetivos generales

El principal objetivo de este proyecto es la creación de varias herramientas capaces de extraer y procesar información sobre un familia de productos concreta de Amazon, en este caso camisetas, para su posterior uso en estudios de mercado.

A su vez este proceso está dividido en diferentes fases que podemos identificar como objetivos secundarios:

- Extraer información de los distintos productos. Esta información incluye campos como el nombre de la marca, el rango de precios, número de valoraciones o imágenes del producto.
- Implementar un clasificador automático de imágenes.
- Almacenar la información descargada y las imágenes clasificadas en múltiples tablas con sus respectivos campos dentro de una base de datos.
- Visualizar la información descargada y procesada en un documento Excel.

## B.3. Catalogo de requisitos

### Requisitos funcionales

A continuación se muestran los requisitos funcionales que han sido implementados en las herramientas creadas para el proyecto:

- **RF 1 Extraer información de los productos.** La herramienta debe ser capaz de extraer los diferentes campos de los productos deseados.
  - **RF 1.1 Extraer los campos principales.** La aplicación debe ser capaz de extraer los campos principales de cada producto.
  - **RF 1.2 Extraer las imágenes.** La aplicación debe de ser capaz de extraer los enlaces de las imágenes asociadas a cada producto.
  - **RF 1.3 Extraer los comentarios.** La aplicación debe ser capaz de extraer los comentarios que los clientes han dejado a cada artículo.
- **RF 2 Almacenamiento en base de datos.** La herramienta debe ser capaz de almacenar la información extraída en una base de datos.
- **RF 3 Generar archivo JSON.** La aplicación debe ser capaz de generar un archivo JSON que contenga de forma estructurada todos los campos extraídos de cada producto.
  - **RF 3.1 Generar un archivo JSON con los campos principales.** La herramienta debe ser capaz de generar un archivo JSON que contenga los campos principales de cada artículo y los enlaces a sus imágenes.
  - **RF 3.2 Generar un archivo JSON con los comentarios.** La herramienta debe ser capaz de generar un archivo JSON que contenga los comentarios que los clientes han dejado a cada artículo.
- **RF 4 Entrenar un clasificador de imágenes.** La herramienta debe poder entrenar un clasificador de imágenes.
  - **RF 4.1 Conjunto de entrenamiento.** La herramienta ha de ser capaz de crear un conjunto de datos de entrenamiento para entrenar los clasificadores.
  - **RF 4.2 Entrenar clasificador modelos.** La herramienta debe de ser capaz de entrenar un clasificador de imágenes que detecte si en una imagen aparece un modelo o no.
  - **RF 4.3 Entrenar clasificador caras.** La herramienta debe ser capaz de entrenar un clasificador de imágenes que detecte si en una imagen la cara de un modelo es visible o no.

- **RF 5 Clasificar automáticamente imágenes.** La herramienta ha de ser capaz de clasificar imágenes de forma automática.
  - **RF 5.1 Clasificador modelos.** La herramienta debe ser capaz de clasificar imágenes en función de si en una imagen aparece un modelo o no.
  - **RF 5.2 Clasificador caras.** La herramienta debe ser capaz de clasificar imágenes en función de si en una imagen la cara de un modelo es visible o no.
- **RF 6 Generar Excel.** La herramienta ha de ser capaz de generar un documento Excel que contenga toda la información sobre los productos extraídos, a partir de la base de datos existente.

### Requisitos no funcionales

- **RNF 1 Proceso automático.** Todo el proceso, desde la extracción de datos a la generación del documento final, ha de ser lo mas automatizado posible.
- **RNF 2 Facilidad instalación.** La herramienta debe ser fácil de instalar y ponerse en marcha.
- **RNF 2 Software libre.** La herramienta ha de utilizar software libre.

## B.4. Especificación de requisitos

### Diagramas de casos de uso

Los siguientes diagramas muestran los casos de uso de los requisitos definidos previamente. Han sido creados con la herramienta Createley<sup>1</sup>.

---

<sup>1</sup><https://app.createley.com>

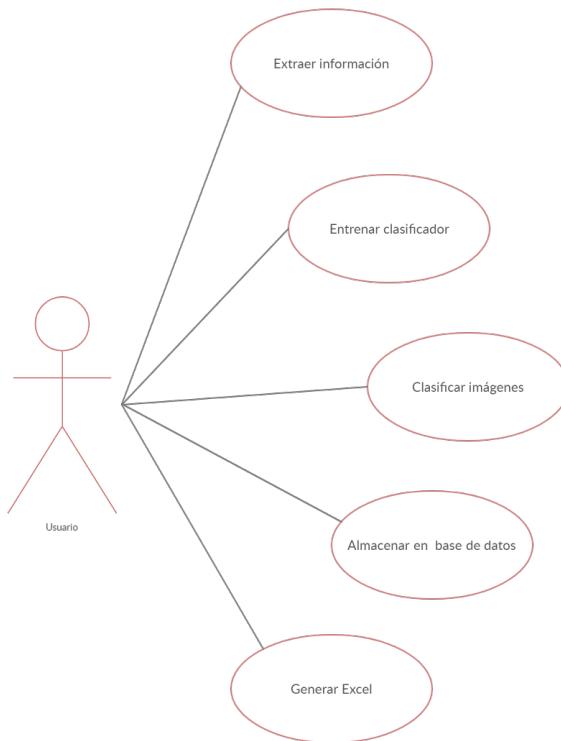


Figura B.1: Diagrama general de los casos de uso

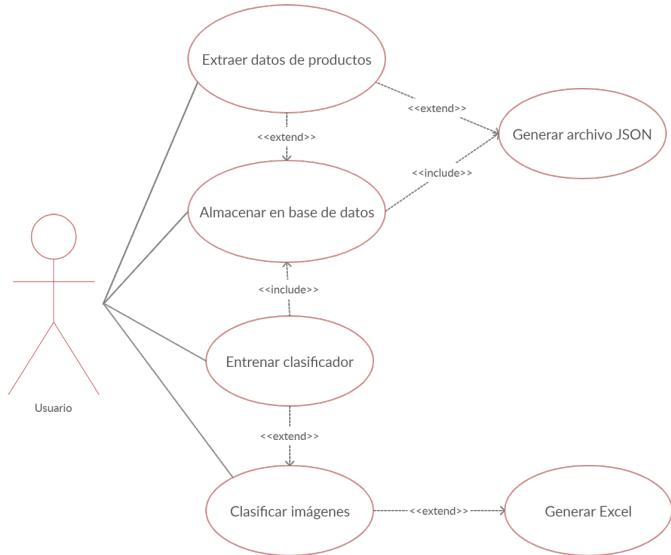


Figura B.2: Diagrama desglosado de los casos de uso del usuario

---

Caso de uso 1: Extraer información de los productos.											
Descripción	Permite al usuario extraer la información los productos deseados.										
Requisitos	RF 1 RF 1.1 RF 1.2 RF 1.3										
Precondiciones	Es necesaria conexión a Internet y los enlaces de los productos a extraer.										
Secuencia normal	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>El usuario indica el fichero que contiene los enlaces de los productos.</td></tr> <tr> <td>2</td><td>Se inicia el proceso de <i>web scraping</i>.</td></tr> <tr> <td>3</td><td>Se extrae de forma secuencial todos los campos de todos los productos seleccionados.</td></tr> <tr> <td>4</td><td>Se guarda en la base de datos la información extraída.</td></tr> </tbody> </table>	Paso	Acción	1	El usuario indica el fichero que contiene los enlaces de los productos.	2	Se inicia el proceso de <i>web scraping</i> .	3	Se extrae de forma secuencial todos los campos de todos los productos seleccionados.	4	Se guarda en la base de datos la información extraída.
Paso	Acción										
1	El usuario indica el fichero que contiene los enlaces de los productos.										
2	Se inicia el proceso de <i>web scraping</i> .										
3	Se extrae de forma secuencial todos los campos de todos los productos seleccionados.										
4	Se guarda en la base de datos la información extraída.										
Postcondiciones	Se ha extraído y almacenado la información de los productos deseados.										
Excepciones	No se ha podido acceder a «Amazon.com».										
Importancia	Alta										
Urgencia	Alta										

---

Tabla B.1: Caso de uso 1: Extraer información de los productos.

---

Caso de uso 2: Almacenamiento en base de datos.

Descripción	Permite al usuario almacenar la información extraída en una base de datos.	
Requisitos	RF 1 RF 2	
Precondiciones	Se debe haber ejecutado el <i>web scraper</i> .	
Secuencia normal	Paso	Acción
	1	Se crea una conexión con la base de datos.
	2	Se crean las diferentes tablas y sus campos.
	3	Se almacenan los campos de los productos descargados.
	4	Se cierra la conexión con la base de datos
Postcondiciones	Todos los campos quedan almacenados correctamente.	
Excepciones	Excepción SQL o sintaxis incorrecto.	
Importancia	Alta	
Urgencia	Media	

---

Tabla B.2: Caso de uso 2: Almacenamiento en base de datos.

---

Caso de uso 3: Generar archivo JSON.

Descripción	Permite al usuario extraer la información los productos deseados.	
Requisitos	RF 1 RF 2	
Precondiciones	Es necesario disponer de una terminal y conocer la ruta donde se desea guardar el archivo JSON.	
Secuencia normal	Paso	Acción
	1	El usuario indica la ruta del archivo que desea crear.
	2	Se inicia el proceso de <i>web scraping</i> .
Postcondiciones	Se genera correctamente el fichero JSON que contiene la información descargada.	
Excepciones	Error de sintaxis.	
Importancia	Media	
Urgencia	Media	

---

Tabla B.3: Caso de uso 3: Generar archivo JSON.

---

Caso de uso 4: Entrenar un clasificador de imágenes.											
Descripción	Permite al usuario extraer la información los productos deseados.										
Requisitos	RF 1 RF 1.1 RF 1.2 RF 1.3										
Precondiciones	Es necesario disponer del conjunto de datos de entrenamiento.										
Secuencia normal	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>Se descargan las imágenes etiquetadas manualmente.</td></tr> <tr> <td>2</td><td>Se divide el conjunto en 2 subconjuntos: entrenamiento y test.</td></tr> <tr> <td>3</td><td>Se comienza a entrenar el clasificador a partir de estos conjuntos.</td></tr> <tr> <td>4</td><td>Se descarga el clasificador ya entrenado para su posterior uso.</td></tr> </tbody> </table>	Paso	Acción	1	Se descargan las imágenes etiquetadas manualmente.	2	Se divide el conjunto en 2 subconjuntos: entrenamiento y test.	3	Se comienza a entrenar el clasificador a partir de estos conjuntos.	4	Se descarga el clasificador ya entrenado para su posterior uso.
Paso	Acción										
1	Se descargan las imágenes etiquetadas manualmente.										
2	Se divide el conjunto en 2 subconjuntos: entrenamiento y test.										
3	Se comienza a entrenar el clasificador a partir de estos conjuntos.										
4	Se descarga el clasificador ya entrenado para su posterior uso.										
Postcondiciones	Se entrenado correctamente el clasificador y es capaz de etiquetar las imágenes con una precisión aceptable.										
Excepciones	Error de sintaxis.										
Importancia	Alta										
Urgencia	Alta										

---

Tabla B.4: Caso de uso 4: Entrenar un clasificador de imágenes.

---

Caso de uso 5: Clasificar automáticamente las imágenes.

---

Descripción	Permite al usuario extraer la información los productos deseados.								
Requisitos	RF 1 RF 1.1 RF 1.2 RF 1.3 RF 3.1 RF 4 RF 4.1 RF 4.2 RF 4.3								
Precondiciones	Es necesario disponer de un clasificador previamente entrenado.								
Secuencia normal	<table border="1"> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El usuario indica la ruta de los productos de los que desea clasificar sus imágenes.</td> </tr> <tr> <td>2</td> <td>El clasificador itera sobre estas imágenes anotando una etiqueta en cada una en función de su clasificación.</td> </tr> <tr> <td>4</td> <td>Se guardan en la base de datos las anotaciones generadas asociadas a cada producto.</td> </tr> </tbody> </table>	Paso	Acción	1	El usuario indica la ruta de los productos de los que desea clasificar sus imágenes.	2	El clasificador itera sobre estas imágenes anotando una etiqueta en cada una en función de su clasificación.	4	Se guardan en la base de datos las anotaciones generadas asociadas a cada producto.
Paso	Acción								
1	El usuario indica la ruta de los productos de los que desea clasificar sus imágenes.								
2	El clasificador itera sobre estas imágenes anotando una etiqueta en cada una en función de su clasificación.								
4	Se guardan en la base de datos las anotaciones generadas asociadas a cada producto.								
Postcondiciones	Las imágenes han sido clasificadas y las anotaciones almacenadas.								
Excepciones	Error de sintaxis. Error al cargar una imagen. Error al clasificar. Excepción SQL								
Importancia	Alta								
Urgencia	Alta								

---

Tabla B.5: Caso de uso 5: Clasificar automáticamente las imágenes.

---

Caso de uso 6: Generar documento Excel.

---

Descripción	Permite al usuario generar un documento Excel con toda la información descargada y procesada.								
Requisitos	RF 1 RF 1.1 RF 1.2 RF 1.3 RF 3.1 RF 4 RF 4.1 RF 4.2 RF 4.3								
Precondiciones	Es necesario disponer de la base de datos completa junto con las predicciones del clasificador de imágenes.								
Secuencia normal	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>Creación de <i>dataframes</i> con la ayuda de la librería Pandas a partir de la base de datos.</td></tr> <tr> <td>2</td><td>Generación del documento Excel a partir de los <i>dataframes</i> creados anteriormente.</td></tr> <tr> <td>4</td><td>Descarga del documento Excel que contiene toda la información.</td></tr> </tbody> </table>	Paso	Acción	1	Creación de <i>dataframes</i> con la ayuda de la librería Pandas a partir de la base de datos.	2	Generación del documento Excel a partir de los <i>dataframes</i> creados anteriormente.	4	Descarga del documento Excel que contiene toda la información.
Paso	Acción								
1	Creación de <i>dataframes</i> con la ayuda de la librería Pandas a partir de la base de datos.								
2	Generación del documento Excel a partir de los <i>dataframes</i> creados anteriormente.								
4	Descarga del documento Excel que contiene toda la información.								
Postcondiciones	Se ha generado un documento excel con la información de cada producto y la predicción que ha devuelto el clasificador de imágenes.								
Excepciones	Error de sintaxis. Error al conectar con la base de datos.								
Importancia	Alta								
Urgencia	Alta								

---

Tabla B.6: Caso de uso 6: Generar documento Excel.

## *Apéndice C*

---

# Especificación de diseño

---

## C.1. Introducción

Este apartado se encarga de recoger los diferentes diseños que han sido llevados a cabo para la realización del proyecto y cumplir de forma satisfactoria los requisitos y objetivos anteriormente tratados.

## C.2. Diseño de datos

Esta sección detalla cómo se han almacenado los datos, tanto en la base de datos SQLite, como en los archivos JSON que genera Scrapy.

### Estructura JSON

Cada vez que *scrapeamos* un producto, éste será añadido a un archivo JSON que genera Scrapy. En total, se generan 3 archivos JSON por cada uso de la herramienta:

- **Enlaces a cada producto:** El primer JSON recoge todos los enlaces que apuntan a los productos de los cuales se desea extraer la información. La estructura es muy sencilla ya que únicamente guarda un enlace por cada producto:



Figura C.1: Estructura del JSON con los enlaces a los productos

Como se puede ver, cada página de «Amazon.com» contiene 48 productos. En este JSON se almacenan los enlaces a estos 48 artículos.

- **Campos principales de los productos:** Este JSON almacena los campos que se extraen de cada producto de la siguiente forma:



Figura C.2: Estructura del JSON con los campos de cada producto

Como se puede apreciar, la estructura se compone de los siguientes campos:

**ASIN:** Código identificativo de cada producto.

**Sexo:** Sexo al que va dirigido cada artículo.

**Rango de precios:** El precio de cada producto puede variar en función del color o la talla. Este campo recoge el precio máximo y mínimo de cada producto.

**Puntuación:** Media de las puntuaciones que los clientes han votado a cada producto. Va del 0 al 5.

**Número de valoraciones:** Número de clientes que han proporcionado una valoración del producto.

**Marca:** El nombre de la marca de cada artículo.

**Descripción:** Texto que el vendedor ha utilizado para describir y dar detalles de cada producto.

**Enlaces a las imágenes:** Enlaces de las imágenes asociadas a cada artículo.

- **Comentarios de los productos:** Por último, este JSON se encarga de recoger los diferentes comentarios que los clientes han dejado a cada artículo. Se compone de dos únicos campos: una lista con los comentarios y el identificador del producto:

```
array ► 0 ►
  □ ▼ array [574]
  □ □ ▼ 0 {2}
  □ □   asin : B077ZMKWVM
  □ □   ► comments [10]
  □ □ ▼ 1 {2}
  □ □   asin : B0000ANHT7
  □ □   ► comments [10]
  □ □ ▼ 2 {2}
  □ □   asin : B07BBB7Y6Z7
  □ □   ► comments [10]
  □ □ ▼ 3 {2}
  □ □   asin : B016EAQ42W
  □ □   ► comments [10]
  □ □ ▼ 4 {2}
  □ □   asin : B07KCH442C
  □ □   ► comments [10]
```

Figura C.3: Estructura del JSON con los comentarios de cada producto

### Estructura de la base de datos SQLite

Este proyecto también hace uso de una base de datos para almacenar toda la información una vez han sido procesadas y clasificadas las imágenes. Para ello se hace uso de una base de datos cuyas tablas y campos son los siguientes:

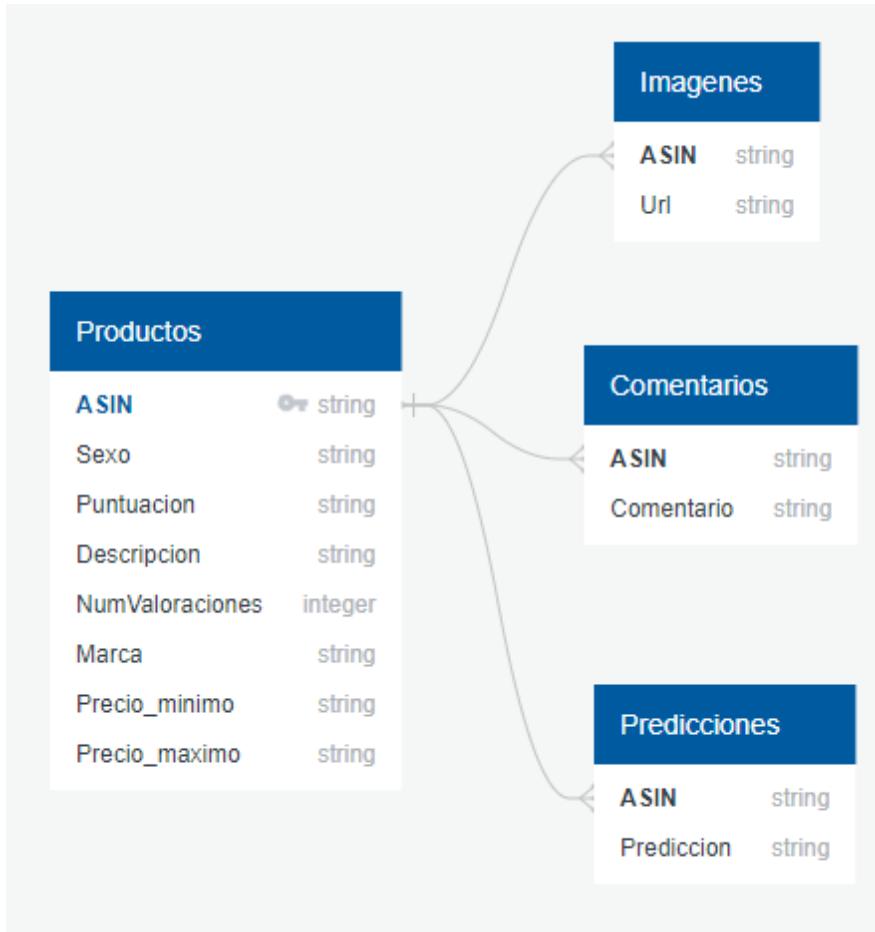


Figura C.4: Diagrama de la base de datos utilizada.

**Tabla productos** Recoge los principales campos de cada producto. Cada entrada pertenece a un artículo diferente.

**Tabla Imágenes** Almacena los enlaces a las imágenes de cada producto. Puede haber varias imágenes para un único producto.

**Tabla Comentarios** Almacena los comentarios asociados a cada artículo. Puede haber varios comentarios de cada producto.

**Tabla Predicciones** Almacena las predicciones que los clasificadores han generado para la imagen principal de cada artículo. Cada entrada es de un producto diferente.

### C.3. Diseño procedimental

En esta sección se muestran los diferentes procedimientos para la realización de los procesos mas importantes del proyecto:

#### *Web scraping*

A la hora de la extracción de información de los productos, el proceso a seguir es el siguiente:

1. Extraer los enlaces a dichos productos. Para ello se hace uso del *spider* llamado «urls\_spider» en Scrapy.
2. Con el archivo JSON que se ha generado en el anterior paso, se pasará a la extracción de información de cada producto por medio del *spider* llamado «products\_spider».
3. Por último, si se desean extraer los comentarios de los productos, se hará uso del *spider* llamado «comments\_spider» para ello.

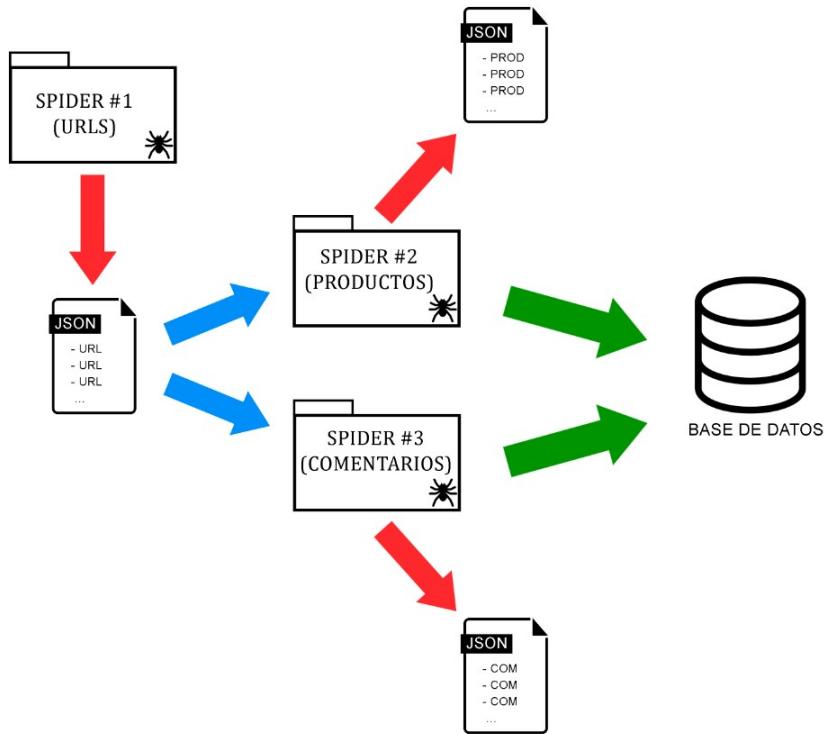


Figura C.5: Proceso de *web scraping* del proyecto

## Entrenamiento de los clasificadores

Para entrenar los clasificadores es necesario haber creado un conjunto de entrenamiento y haber etiquetado de forma manual las imágenes que conforman dicho conjunto. Una vez se dispone de ese conjunto de entrenamiento los pasos para entrenar el clasificador son los siguientes:

1. Descargar el *dataset* que contiene las imágenes previamente etiquetadas.
2. De forma secuencial, se convertirá cada imagen a formato *array* y se almacenará junto a su etiqueta.
3. Crear conjuntos de test y entrenamiento para aplicar la técnica de validación cruzada.
4. Inicializar el modelo del clasificador.
5. Entrenar el clasificador.
6. Almacenar el clasificador ya entrenado para su posterior uso.

## Clasificación de imágenes y almacenamiento de resultados

El proceso de clasificación de imágenes y su posterior almacenamiento es el siguiente:

1. Importar y cargar los clasificadores previamente entrenados.
2. Importar el archivo JSON que contiene los productos a clasificar.
3. Descargar cada imagen con el ASIN del producto como nombre.
4. Clasificar cada imagen siguiendo el siguiente algoritmo:

```

if Imagen cargada correctamente then
    Redimensionar imagen.
    Convertir imagen a formato array.
    Aplicar clasificador que detecta modelo.
    if El clasificador ha encontrado un modelo en la imagen then
        Aplicar clasificador que detecta cara.
        if El clasificador ha encontrado la cara del modelo then
            | Devolver predicción: Modelo con cara.
        end
        Devolver predicción: Modelo sin cara.
    end
    Devolver predicción: Sin modelo.
end

```

**Algoritmo 1:** Algoritmo de clasificación de imágenes

5. Importar la base de datos
6. Crear la tabla «Predicciones» en la base de datos
7. Almacenar en la base de datos cada predicción generada por el clasificador quedando asociada al artículo al que pertenece la imagen.

## C.4. Diseño arquitectónico

En cuanto al diseño arquitectónico del proyecto, es necesario señalar que la primera parte del proyecto, todo lo relacionado con la extracción de datos y *web scraping*, funciona bajo la estructura se Scrapy. El resto se encuentra dividido en dos *notebooks* de Google Colab, el primero para el entrenamiento de los clasificadores, y el segundo destinado a la clasificación de imágenes y su posterior almacenamiento en la base de datos.

El siguiente diagrama muestra una visión general de la arquitectura de Scrapy junto con los componentes que lo conforman y el flujo de datos que se lleva a cabo en el sistema [1]:

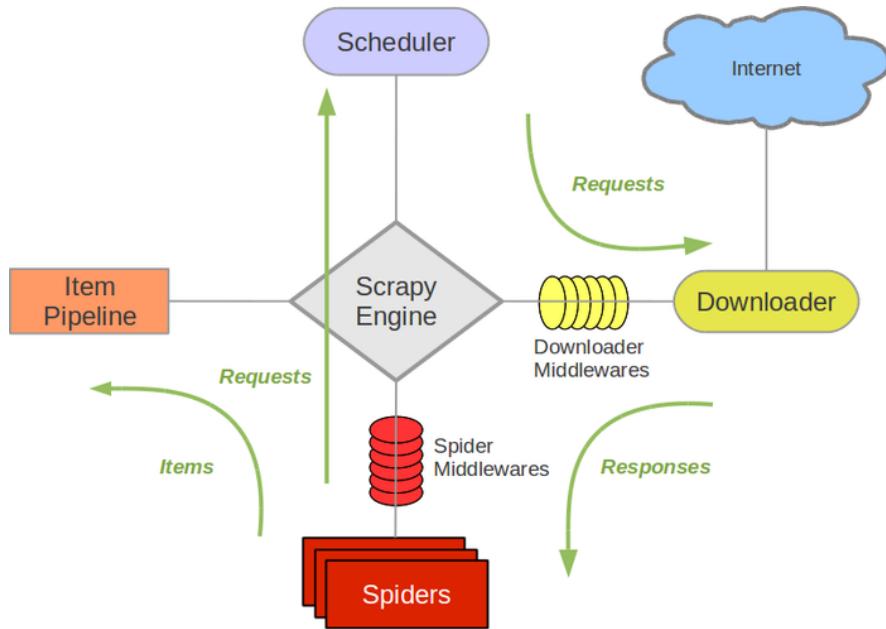


Figura C.6: Arquitectura que sigue Scrapy y sus diferentes componentes

### Componentes de scrapy

A continuación se describen los principales componentes de la arquitectura de un proyecto de Scrapy:

- **Items:** Los diferentes elementos o campos a extraer.
- **Spiders:** Clases escritas por el usuario que contienen los diferentes procedimientos para parsear y extraer los *items* de uno o varios enlaces. En este caso hemos hecho uso de 3 *spiders*: Una para obtener los enlaces a cada artículo, otra para extraer los comentarios, y la última para el resto de campos.
- **Pipelines:** Apartado responsable de procesar los *items* una vez han sido extraídos por las *spiders*. En este proyecto se han usado para almacenar los campos extraídos en una base de datos.

## *Apéndice D*

---

# Documentación técnica de programación

---

## D.1. Introducción

Esta sección se encarga de detallar los conceptos relacionados con la documentación técnica que sigue el proyecto. Se tratará de explicar la estructura de directorios que sigue el proyecto, así como los pasos a seguir para ser instalado y ejecutado.

## D.2. Estructura de directorios

A continuación se muestra la estructura de directorios en la que se distribuye el proyecto:

- **Scrapy/** Directorio principal de Scrapy. Se compone de los siguientes archivos y subdirectorios:
  - **amazon/spiders/** Directorio donde se encuentran las diferentes *spiders*.
    - **urls\_spiders.py** *Spider* utilizada para extraer los enlaces de los productos.
    - **products\_spiders.py** *Spider* utilizada para extraer los principales campos de cada producto.
    - **comments\_spiders.py** *Spider* utilizada para extraer los comentarios asociados a cada producto.
  - **amazon/items.py** Fichero que recoge los diferentes campos que extrae Scrapy.

- **amazon/middlewares.py** Fichero que recoge la configuración de los *middlewares* se Scrapy si los hubiera.
- **amazon/pipelines.py** Fichero que se encarga del almacenamiento en la base de datos de los campos extraídos.
- **amazon/settings.py** Fichero de configuración del proyecto de Scrapy.
- **Colab notebooks/** Directorio donde se encuentran los *notebooks* de Google Colab utilizados en el proyecto:
  - **Clasificador\_Modelo\_Sin\_modelo.ipynb** *Notebook* encargado del entrenamiento del clasificador de imágenes que detecta si hay un modelo en la imagen o no.
  - **Clasificador\_cara\_sin\_cara.ipynb** *Notebook* encargado del entrenamiento del clasificador de imágenes que detecta si la cara de un modelo es visible o no.
  - **Clasificador\_final\_+\_excel\_+\_.db.ipynb** *Notebook* encargado de la clasificación de imágenes y su posterior almacenamiento en la base de datos y documento Excel.
- **Databases/** Carpeta donde se genera la abse de datos una vez finalizado el proceso de *web scraping*.
- **Datasets/** Directorio que recoge los *datasets* utilizados para el entrenamiento de los clasificadores:
  - **dataset-modelo.zip** *Dataset* que contiene las imágenes ya etiquetadas utilizadas para el entrenamiento del clasificador de imágenes que detecta modelos.
  - **dataset-cara.zip** *Dataset* que contiene las imágenes ya etiquetadas utilizadas para el entrenamiento del clasificador de imágenes que detecta caras.
- **Documentacion/** Este directorio contiene todos los archivos relacionados con la documentación y la memoria del proyecto, Tanto los ficheros fuente de LATEX, como los PDFs ya finalizados.
- **Keras\_models/** Directorio donde encontramos los clasificadores de Keras ya entrenados para su posterior utilización:
  - **clasificador-modelo.h5** Clasificador de imágenes encargado de detectar modelos ya entrenado y listo para su utilización.
  - **clasificador-cara.h5** Clasificador de imágenes encargado de detectar caras ya entrenado y listo para su utilización.

- **Outputs/** Directorio donde se guardan todos los resultados del proyecto: Archivos JSON, documentos Excel y bases de datos ya completados.
- **Scripts/** Directorio que incluye algunos *scripts* utilizados para algunas tareas del proyecto:
  - **JSONmerger.py** *Script* que combina los campos de dos archivos JSON utilizando el campo «url» como elemento identificador.
  - **dataextractor.py** *Script* que descarga imágenes dada su URL y las renombra en función de cómo han sido etiquetadas. Utilizado para la creación de los *datasets*.
  - **dataturks\_fixer.py** *Script* usado para arreglar el JSON generado por Dataturks para su correcto funcionamiento.
  - **urlextractor.py** *Script* que extrae los enlaces de las imágenes de cada producto y lo añade a una lista para posteriormente etiquetarlo en Dataturks.
- **Videos/** Directorio que recoge los vídeos que muestran el proceso de ejecución del proyecto.

### D.3. Manual del programador

Apartado que detalla algunos de los procesos llevados a cabo en el proyecto necesarios para que pueda seguir desarrollándose.

#### Manual de Scrapy

Para la creación del proyecto de Scrapy es necesaria la instalación de la librería. Esto se puede hacer de varias formas, tal y como describe la documentación oficial<sup>1</sup>:

- **Utilizando Conda:** Se instala la librería con el siguiente comando:

```
conda install -c conda-forge scrapy
```

- **Utilizando PIP:** Se instala la librería con el siguiente comando:

```
pip install Scrapy
```

---

<sup>1</sup><http://doc.scrapy.org/en/latest/intro/install.html>

Una vez instalado el paquete de Scrapy, la creación de un nuevo proyecto es sencilla:

```
scrapy startproject [nombre_proyecto]
```

Este comando se encarga de crear la estructura de carpetas necesaria para el correcto funcionamiento de Scrapy. A continuación se detalla el proceso a seguir para añadir *spiders* al proyecto:

```
cd [nombre_proyecto]
scrapy genspider [nombre_spider] [direccion_web]
```

En cuanto a la ejecución de un *spider*, se hace con el siguiente comando:

```
scrapy crawl [nombre_spider] -o [nombre_output]
```

### Manual de Dataturks

A continuación se detallará el proceso a seguir para el etiquetado manual de imágenes con la ayuda de la herramienta Dataturks<sup>2</sup>.

1. Iniciar sesión por medio de GitHub o creando una cuenta a parte.
2. Crear un nuevo *dataset* y elegir el tipo adecuado, en este caso será el llamado *Image Classification*.

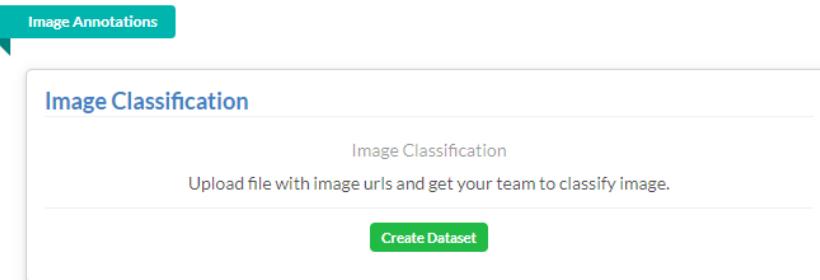


Figura D.1: Tipo de *dataset* a crear en Dataturks

3. Configurar el funcionamiento del *dataset*

---

<sup>2</sup><https://dataturks.com/>

## Create Dataset

Dataset Name

List of Entities/Categories

Tagging Instruction

Etiquetar "Cara" si la cara del modelo está visible.  
Etiquetar "Sin cara" si la cara del modelo no está visible.

Submit

The screenshot displays a user interface for creating a dataset. At the top, the title 'Create Dataset' is centered. Below it, there are three main input fields: 'Dataset Name' containing 'Dataset caras', 'List of Entities/Categories' containing 'Cara, Sin cara', and 'Tagging Instruction' which contains the text: 'Etiquetar "Cara" si la cara del modelo está visible.' followed by 'Etiquetar "Sin cara" si la cara del modelo no está visible.'. At the bottom right of the form area, there is a grey 'Submit' button.

Figura D.2: Creación de un *dataset* en Dataturks

4. Importar una lista con los enlaces a las imágenes que se desean etiquetar. Deberán estar en un documento *txt*.
5. Una vez subidas las imágenes, se puede empezar a etiquetar seleccionando la etiqueta adecuada en cada caso.

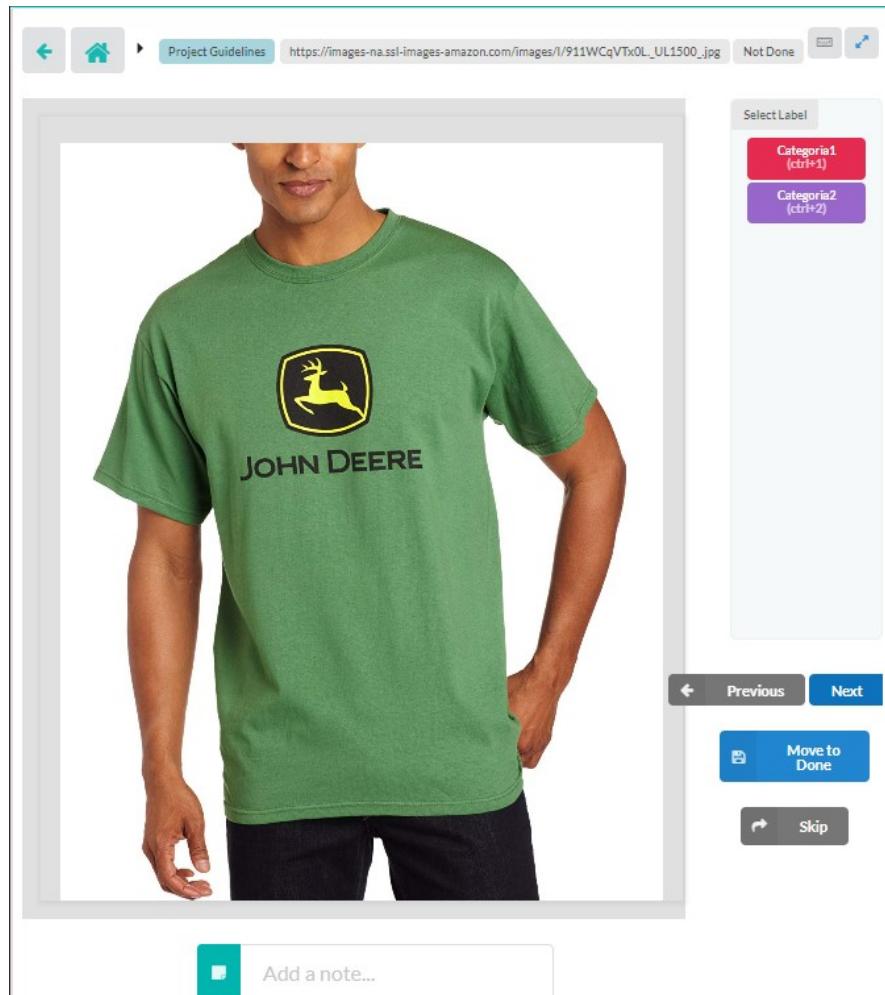


Figura D.3: Proceso de etiquetado manual en Dataturks

6. Por último, para exportar los resultados simplemente se ha de descargar el archivo generado pulsando en el botón *Download*.

### Entrenamiento de los clasificadores

El proceso de entrenamiento de los clasificadores es muy sencillo ya que está muy automatizado. Los *notebooks* se encargan de descargar y descomprimir el *dataset* previamente subido al repositorio del proyecto para así entrenar el clasificador. Una vez entrenado, se muestra la precisión alcanzada y un resumen de las capas utilizadas por el modelo de Keras.

Por último se descarga el clasificador ya entrenado para su posterior uso.

## D.4. Compilación, instalación y ejecución del proyecto

Lo más importante a la hora de instalar y ejecutar este proyecto es la parte de Scrapy. Puesto que en el repositorio ya se encuentra el proyecto de Scrapy al completo, lo primero será descargar el contenido del repositorio. Para ello se utiliza el siguiente comando:

```
git clone https://github.com/daniarnaizg/TFG-Amazon-Scraper.git
```

Con esto, quedará descargado el repositorio. Lo siguiente será instalar el paquete de Scrapy si no se tiene instalado (Ver manual de Scrapy en la sección anterior).

También es necesaria la instalación de varios paquetes, se pueden instalar por medio de PIP:

```
pip install js2xml
pip install pypiwin32
pip install Scrapy-UserAgents
```

A partir de aquí simplemente se puede lanzar una *spider*, tal y como se ha mostrado antes. Es necesario indicar el fichero al que se desea exportar el contenido descargado en formato JSON:

```
scrapy crawl urls_spider -o urls_output.json
scrapy crawl products_spider -o products_output.json
scrapy crawl comments_spider -o comments_output.json
```

Es importante tener en cuenta el nombre del fichero JSON ya que por defecto está configurado para leer los campos de los ficheros con estos nombres. Se puede utilizar otro nombre siempre y cuando se cambie el nombre del archivo de entrada en el código de la *spider*.

En cuanto a la ejecución de los *notebooks*, esto se hará importando los archivos *ipynb* en el entorno de Google Colab a través del siguiente enlace:  
<https://colab.research.google.com/>

## *Apéndice E*

---

# Documentación de usuario

---

## E.1. Introducción

En esta sección se tratará de explicar todo el proceso de utilización de las herramientas que conforman el proyecto. Se tratará de especificar los requisitos técnicos necesarios para el buen funcionamiento, el proceso de instalación, y por último un manual de uso de las herramientas.

## E.2. Requisitos de usuarios

A continuación se listan los requisitos técnicos necesarios para el correcto funcionamiento de las herramientas:

- **Sistema operativo:** Necesario para la realización del proyecto. Se ha utilizado Windows 10, aunque en principio todas las librerías son compatibles con Linux.
- **Conexión a internet:** Es necesario para realizar la extracción de datos y para ejecutar los *notebooks* en Google Colab.
- **Navegador Web:** Necesario para acceder a Google Colab. Se ha utilizado Google Chrome aunque cualquier navegador es compatible.
- **Python 3.7:** Necesario para la ejecución del *web scraper*.
- **Librerías:** Serán necesarias las librerías de las que hace uso el proyecto de Scrapy. Estas librerías son:
  - Scrapy
  - js2xml
  - pypiwin32

- Scrapy-UserAgents
- **Proyecto:** Se deberá descargar el contenido del repositorio<sup>1</sup> donde se encuentran los archivos fuente del proyecto.

### E.3. Instalación

En este apartado se detalla el proceso de instalación necesario para el correcto funcionamiento del proyecto:

- **Python:** Existen diversas formas de instalar Python en Windows:
  - Método manual: Este método es sencillo y rápido.
    1. Descargar el instalador desde la web oficial de Python:  
<https://www.python.org/downloads/>  
Es importante que sea la versión 3.7.



Figura E.1: Descarga de Python

2. Ejecutar el instalador y seguir los pasos que muestra.
- Anaconda: Método más complejo pero con algunos paquetes preinstalados.
  1. Descargar el instalador desde la web oficial de Anaconda:  
<https://www.anaconda.com/distribution/#download-section>  
Es importante que sea la versión 3.7.

---

<sup>1</sup><https://github.com/daniarnaizg/TFG-Amazon-Scraper>

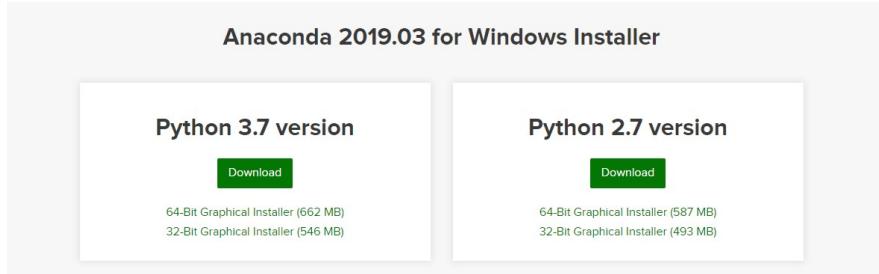


Figura E.2: Descarga de Anaconda

2. Ejecutar el instalador y seguir los pasos que muestra.

- **Librerías:** Una vez instalado Python, será necesario instalar las librerías de las que hace uso el proyecto de Scrapy. Para ello es necesario la instalación del gestor de paquetes PIP<sup>2</sup>. Si se ha instalado Python con el método Anaconda, este gestor ya estará instalado. En caso contrario se deberá instalar con los siguientes comandos:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py
```

Una vez PIP se encuentra operativo, se deberán instalar las librerías necesarias con los siguientes comandos:

```
pip install Scrapy
pip install js2xml
pip install pypiwin32
pip install Scrapy-UserAgents
```

- **Descarga del proyecto:** Para descargar el proyecto existen dos formas:
  - Descargar un fichero ZIP con el contenido del repositorio desde el siguiente enlace:  
<https://github.com/daniarnaizg/TFG-Amazon-Scraper>

---

<sup>2</sup><https://www.pypa.io/en/latest/>

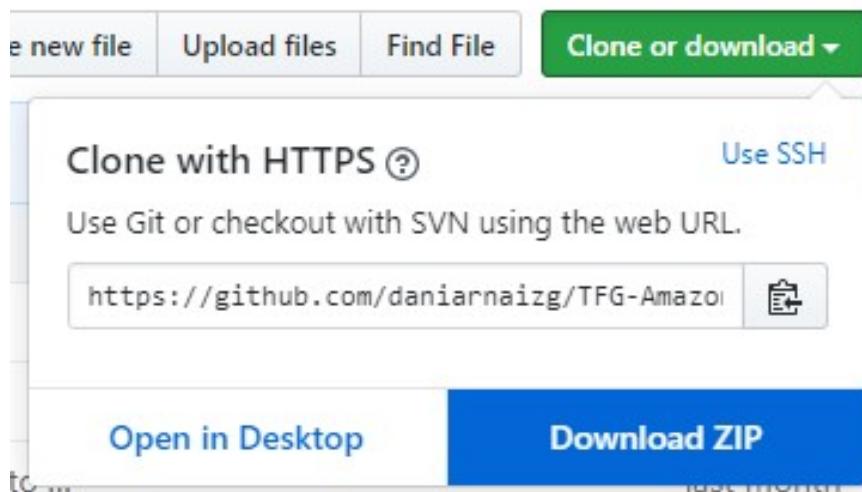


Figura E.3: Descarga del contenido del repositorio

- Clonar el repositorio completo a través de Git con el siguiente comando:

```
git clone https://github.com/daniarnaizg/TFG-Amazon-Scraper.git
```

## E.4. Manual del usuario

Se dividirá este manual en varias secciones:

### Proceso de *web scraping*

En primer lugar será necesario tener el enlace de la página que contiene los productos a extraer, para ello basta con hacer una búsqueda en los departamentos de Amazon.com y una vez se tenga la categoría de los productos deseados, se anota la URL.

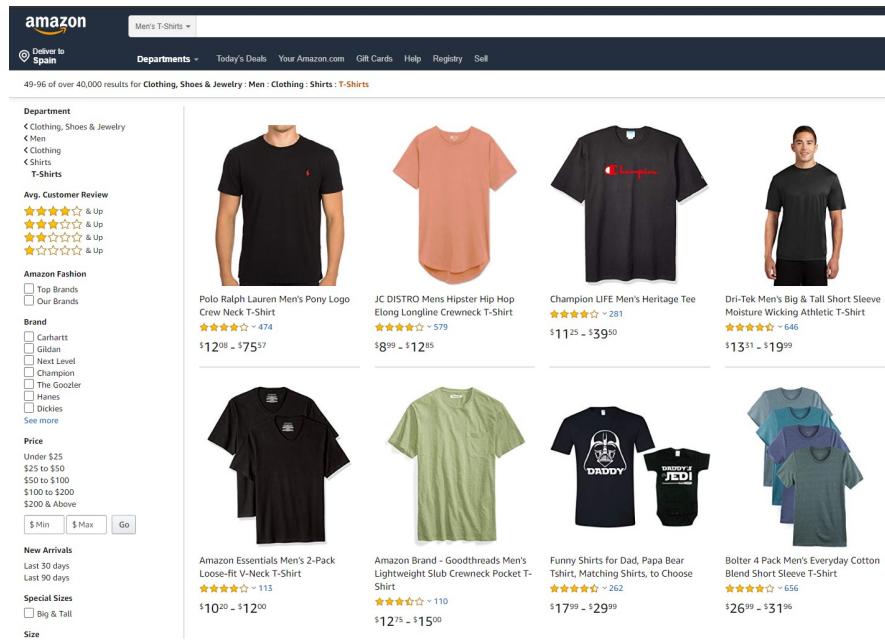


Figura E.4: Página que contiene los productos a extraer

En este caso se necesitará una URL para los productos para hombre, y otra para los productos para mujer. Un ejemplo podría ser el siguiente:

[https://www.amazon.com/s?rh=n%3A7141123011%2Cn%3A7147445011%2Cn%3A12035955011%2Cn%3A9103696011%2Cn%3A9056985011%2Cn%3A9056986011&page=2&qid=1562069588&ref=lp\\_9056986011\\_pg\\_2](https://www.amazon.com/s?rh=n%3A7141123011%2Cn%3A7147445011%2Cn%3A12035955011%2Cn%3A9103696011%2Cn%3A9056985011%2Cn%3A9056986011&page=2&qid=1562069588&ref=lp_9056986011_pg_2)

El siguiente paso será modificar el código de la *spider* llamada «urls\_spider» para que utilice los enlaces que acabamos de anotar. Hay que cambiar el código directamente ya que las *spiders* de Scrapy no admiten ejecuciones parametrizadas a parte de la exportación de los resultados. El número de páginas que se desean extraer se basa en el valor de la variable llamada «n\_pages».

```

classUrlsSpider(scrapy.Spider):
    name = 'urls_spider'
    allowed_domains = ['amazon.com']
    n_pages = 1

    men_urls = [
        'https://www.amazon.com/s?rh=n%3A7141123011%2Cn%3A7147445011%2Cn%3A12035955011%2Cn%3A910369'

    women_urls = []
        'https://www.amazon.com/s?rh=n%3A7141123011%2Cn%3A7147445011%2Cn%3A12035955011%2Cn%3A910369'

    start_urls = men_urls + women_urls

```

Figura E.5: URLs introducidos en la *spider* encargada de extraer los enlaces de cada producto

El siguiente paso será la ejecución de esta *spider*, para ello es necesario situarse en la raíz del proyecto de Scrapy y ejecutar el siguiente comando:

```
scrapy crawl urls_spider -o urls_output.json
```

Es necesario que el archivo JSON que se pasa por parámetro tenga ese nombre ya que las siguientes *spiders* toman este archivo como punto de partida.

```

PS D:\TFG\TFG-Amazon-Scraper\_scrapy\amazon> scrapy crawl urls_spider -o urls_output.json
2019-07-02 17:35:43 [scrapy.utils.log] INFO: Scrapy 1.5.1 started (bot: amazon)
2019-07-02 17:35:43 [scrapy.utils.log] INFO: Versions: lxml 4.2.5.0, libxml2 2.9.5, cssselect 1.1.0, pyDispatcher 2.2.5, w3lib 1.18.0.0 (OpenSSL 1.1.0j 20 Nov 2018), cryptography 2.4.2, Platform Windows-10-10.0.18362-SP0
2019-07-02 17:35:43 [scrapy.crawler] INFO: Overridden settings: {'BOT_NAME': 'amazon', 'CONCURRENT_REQUESTS': 16, 'ROBOTSTXT_OBEY': True, 'SPIDER_MODULES': ['amazon.spiders']}
2019-07-02 17:35:43 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.feedexport.FeedExporter',
 'scrapy.extensions.logstats.LogStats']
2019-07-02 17:35:43 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware',
 'scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.defaul

```

Figura E.6: *urls\_spider* ejecutada y en funcionamiento

Al finalizar este proceso, aparecerá un resumen con el número de elementos extraídos, el tiempo que ha necesitado, y demás información. En estos momentos ya se ha generado el primer JSON del proceso. Este JSON contiene los enlaces de los productos que contenía la URL.

```
TFG-Amazon-Scraper › _scrapy › amazon › urls_output.json › ...
2
3   "page_urls": [
4     "https://www.amazon.com/dp/B07GP35CDX/M",
5     "https://www.amazon.com/dp/B00ITCDOSG/M",
6     "https://www.amazon.com/dp/B00Z7SJRRM/M",
7     "https://www.amazon.com/dp/B000C4UATA/M",
8     "https://www.amazon.com/dp/B01K6PBF0C/M",
9     "https://www.amazon.com/dp/B0101S7K6C/M",
10    "https://www.amazon.com/dp/B01K6PBIRM/M",
11    "https://www.amazon.com/dp/B07DQ7PZBK/M",
12    " Ctrl + click to follow link /dp/B07DQF2SYV/M",
13    "https://www.amazon.com/dp/B07S5PTGSG/M".
```

Figura E.7: JSON que contiene los enlaces extraídos.

Es el momento de pasar al siguiente punto en el proceso de *web scraping*. En este segundo punto, se extraerán los principales campos de cada producto. El proceso a partir de aquí no requiere de nada más que la ejecución de comandos en la terminal.

La primera *spider* que se va a ejecutar será la llamada «products\_spider». Es la encargada de iterar sobre todos los enlaces extraídos en el paso anterior, y extraer los campos deseados por medio de expresiones regulares y XPath. También se encarga de extraer los enlaces de las imágenes de cada producto.

Como en la anterior *spider*, se ejecuta con el siguiente comando:

```
scrapy crawl products_spider -o products_output.json
```

```
2019-07-02 17:51:31 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.amazon.com/dp/B07GP35CDX/M> (referer: None)
-----
Product stored in Database
-----
2019-07-02 17:51:32 [scrapy.core.scraping] DEBUG: Scraped from <200 https://www.amazon.com/dp/B07GP35CDX/M>
{'asin': 'B07GP35CDX',
 'brand': 'OtterBox',
 'description': 'With Symmetry Series for iPhone 8 Plus and iPhone 7 Plus, '
 'thin equals tough. Featuring an ultra-slim profile and '
 'OtterBox Certified Drop Protection, Symmetry Series stays '
 'true to your phone\'s sleek design - and we stay true to our '
 'promise of protection. Plus, with the wild array of colors, '
 'graphics and styles, your phone is always dressed to '
 'impress.\n'
 '\t\t\t\t\t\n'
 '\t\t\t\t',
 'image_urls': ['https://images-na.ssl-images-amazon.com/images/I/614w8-k-0ZL..\_SL1000\_.jpg',
 'https://images-na.ssl-images-amazon.com/images/I/71HDFcjdjYL..\_SL1000\_.jpg',
 'https://images-na.ssl-images-amazon.com/images/I/61dt70bdENL..\_SL1000\_.jpg',
 'https://images-na.ssl-images-amazon.com/images/I/81%2BsHswtx0L..\_SL1500\_.jpg',
 'https://images-na.ssl-images-amazon.com/images/I/81n7mgeaNL..\_SL1500\_.jpg',
 'https://images-na.ssl-images-amazon.com/images/I/91uKB-UiILL..\_SL1500\_.jpg'],
 'max_price': '54.95',
 'min_price': '54.95',
 'rating': '4.1',
 'reviews': 4186,
 'sex': 'Male'}
```

Figura E.8: Campos de un producto extraído durante la ejecución de la *spider*

Cada producto que se extrae queda automáticamente almacenado en la base de datos. No es necesario manipular nada para lograrlo. También se genera otro archivo JSON que se utilizará en la siguiente *spider*.

El siguiente paso se basa en extraer los comentarios de cada producto, para ello partimos del fichero creado anteriormente «products\_output.json» y ejecutamos la *spider* llamada «comments\_spider» con el siguiente comando:

```
scrapy crawl products_spider -o products_output.json
```

```
2019-07-02 18:01:25 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.amazon.com/product-reviews/B00ITCDO5G/ref=cm_cr_arp_d_paging_btm_next_2?pageNumber=1>
Comment stored in Database
-----
2019-07-02 18:01:26 [scrapy.core.scrapers] DEBUG: Scraped from <200 https://www.amazon.com/product-reviews/B00ITCDO5G/ref=cm_cr_arp_d_paging_btm_next_2?pageNumber=1>
{'asin': 'B00ITCDO5G',
 'comments': ['I am a woman that usually takes an 8 to an 8.5 sized shoe. I ' +
    'read mixed reviews, some saying these fit as expected while ' +
    'others said they were a little small. I looked up the Nike ' +
    'Men's shoe size chart and measured my feet. I saw that a 6.5 " ' +
    'would fit perfectly. Based on reviews and my experience with ' +
    'Nike products (typically are made half a size smaller than my ' +
    'normal size) I decided to order a size 7. That way, if the ' +
    'sandals was a little small for my "expected" size, a size 7 ' +
    'should fit just right. I ordered them and they fit perfectly. The ' +
    'size 7 was a perfect fit!The straps have nice padding in them ' +
    'and the bottoms of the sandals themselves are very comfortable. ' +
    'You can tell they are made nicely and should last quite a few ' +
    'summers!On bear feet I have a good half an inch of room left at ' +
    'the tops with maybe a quarter of an inch of space on the backs ' +
    'of my feet. The sandals also fit well with regular socks and ' +
    'thin socks. I highly recommend these for the price. They are ' +
    '# GOOD value. These will last you a while. I would highly ' +
    'suggest buying your size off of your history with Nike products ' +
    'as I do find their shoes are made a little smaller than other ' +
    'companies. Also remember, if you are female, order 1.5 to 2 ' +
    'inches smaller than your regular shoe size. I usually wear a ' +
    'size 8.5 and since my size 8 Nikes fit well but are snug, I ' +
    'decided on a size 7. Hope this helps!',
```

Figura E.9: Comentarios de un producto extraídos durante la ejecución de la *spider*

Al finalizar la ejecución, ya tendremos toda la información extraída de los productos deseados almacenada en la base de datos y los archivos JSON completos. Es el momento de guardar estos archivos para utilizarlos más adelante.

Las rutas de estos archivos son las siguientes:

- **/TFG-Amazon-Scraper/\_scrapy/amazon/urls\_output.json**
- **/TFG-Amazon-Scraper/\_scrapy/amazon/products\_output.json**
- **/TFG-Amazon-Scraper/\_scrapy/amazon/comments\_output.json**
- **/TFG-Amazon-Scraper/databases/database.db**

Hay que recordar borrar el contenido de estos ficheros si se desea volver a ejecutar el proyecto de Scrapy.

Todo este proceso se puede encontrar en formato vídeo, en el archivo llamado *1 - Proceso de Web Scraping - Scrapy.mp4* que se encuentra en el directorio «Videos» dentro del repositorio del proyecto.

## Ejecución de *notebooks*

Tanto el proceso de entrenamiento de los clasificadores, como el de clasificación de imágenes se encuentran en *notebooks* pensados para ser ejecutados en el entorno remoto de Google Colab.

Para ejecutar estos *notebooks* basta con importarlos en un nuevo cuaderno de Google Colab a través del siguiente enlace: <https://colab.research.google.com/>

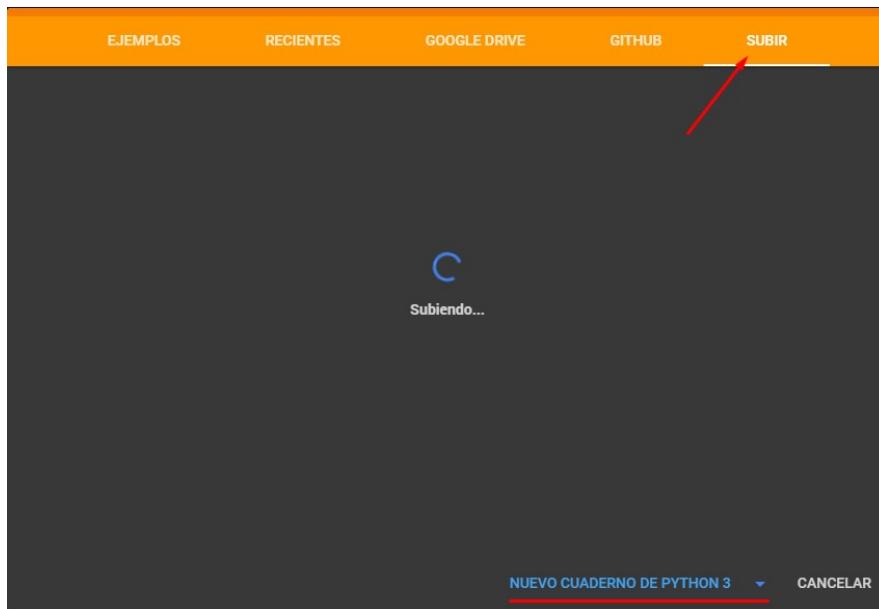


Figura E.10: Subir *notebook* a Google Colab

Figura E.11: *Notebook* de Google Colab abierto en el navegador web

Tanto los cuadernos de entrenamiento de los clasificadores, como el cuaderno general de clasificación, almacenamiento y visualización de los productos extraídos están pensados para ser ejecutados sin necesidad de cambiar el código.

Para ello únicamente es necesario pulsar en el botón de ejecución de cada celda o pulsar la combinación de teclas *Shift + Enter*.



The screenshot shows a Google Colab interface. At the top, there's a section titled "Importación de los clasificadores ya entrenados" with a note about loading models from GitHub. Below this is a terminal window showing the command-line output of running a cell. A red arrow points to the play button icon in the toolbar above the terminal, indicating where to click to execute the code.

```
1 # Descargamos los clasificadores ya entrenados.
2 !wget https://github.com/danialrnatig/TFG-Amazon-Scraper/raw/master/keras_models/clasificador-modelos.h5
3 !wget https://github.com/danialrnatig/TFG-Amazon-Scraper/raw/master/keras_models/clasificador-caras.h5

--2019-06-26 15:05:40-- https://github.com/danialrnatig/TFG-Amazon-Scraper/raw/master/keras_models/clasificador-modelos.h5
Resolving github.com (github.com)... 192.30.253.112
Connecting to github.com (github.com)|192.30.253.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/danialrnatig/TFG-Amazon-Scraper/master/keras_models/clasificador-modelos.h5 [following]
--2019-06-26 15:05:40-- https://raw.githubusercontent.com/danialrnatig/TFG-Amazon-Scraper/master/keras_models/clasificador-modelos.h5
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 151.101.64.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 33287738 (32M) [application/octet-stream]
Saving to: 'clasificador-modelos.h5'
```

Figura E.12: Botón de ejecución de Google Colab

---

## Bibliografía

---

- [1] Scrapy Docs. Scrapy - architecture overview, 2018.  
<https://docs.scrapy.org/en/latest/topics/architecture.html>.