UNIVERSIDAD POLITECNICA DE CARTAGENA

TRABAJO FINAL DE GRADO

DESARROLLO DE UN ROBOT PYTHON PARA LA AUTOMATIZACION INTELIGENTE DE TAREAS WEB RUTINARIAS

Estefanía Lozano Cano

Mayo 2023

Información relevante del proyecto

Autora	Estefanía Lozano Cano
E-mail de la autora	fanylozano@gmail.com
Director	Francesc Burrull i Mestres
E-mail del director	francesc.burrull@upct.es
Título del TFG	Desarrollo de un robot Python para la automatización inteligente de tareas web rutinarias
Resumen	El proyecto trata sobre el desarrollo de un robot programado en Python que automatiza tareas rutinarias con inteligencia. Por ejemplo, será capaz de conectarse a una base de datos y ejecutar consultas SQL y en función de los resultados realizar acciones en web como rellenar formularios, descargas de documentos, aceptar o denegar pulsando en botones de pop-up, del mismo modo que haría una persona. El TFG va a ir enfocado de forma didáctica de forma que un alumno de la UPCT con solo tener su correo nombre_alumno@edu.upct.es a través de este proyecto se crea automáticamente toda la infraestructura necesaria/deseada en Azure, dotando al alumno de unas herramientas adecuadas para el aprendizaje o manejo avanzado de datos. Dado que en la actualidad y en los próximos años los datos es el eje principal de cualquier empresa para el seguimiento y crecimiento de esta.
Titulación	Grado en Ingeniería Telemática
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de presentación	Mayo 2023

Agradecimientos

A mi **marido**, mi mejor amigo y compañero en todas las etapas y proyectos de mi vida, que lo es todo y que, sin él, nada de esto habría sido posible. Por estar siempre ahí para darme ese apoyo que he necesitado en cualquier situación o momento de mi vida.

A mi familia, por ayudarme durante todo mi periodo de estudiante, apoyarme siempre y confiar en que yo podía con todo de manera incondicional.

A todos mis amig@s y compañer@s con los que he podido compartir toda esta experiencia y con los que tengo un grato recuerdo…

y en especial a un amigo que he podido conocer después de la carrera, habiendo estudiado en la misma facultad y que por coincidencias del destino hemos acabo en la misma ciudad incluso compartiendo alguna empresa y que es como un hermano.

A Francesc, por su apoyo, comprensión y las facilidades en llevar a cabo este proyecto.

Índice

Tabla de Ilustraciones

Tabla de enlaces

Enlace de descarga de la página de Python [1]

Versión de Python que vamos a descargar para nuestro Windows 64 bits [2]

Versión del Driver que controla Chrome [3]

Versión del driver controlador del Chrome que descargamos para Windows [4]

Enlace de Google [5]

Enlace de descarga del fichero de los drivers de Microsoft para SQL [6]

Enlace de descarga directa de Microsoft OBDC for SQL [7]

Enlace para la instalación de los drivers de Microsoft para SQL Server [8]

Enlace de Azure [9]

Enlace del campus de la UPCT [10]

Ventana de login del alumno de la UPCT [11]

Enlace a un tutorial de XPATH de Selenium para PYTHON [12]

Enlace a la web de Azure para estudiante [13]

Introducción

En la actualidad, las plataformas Cloud se ha convertido en un recurso fundamental para el desarrollo y despliegue de aplicaciones y servicios. En particular, Azure es una de las plataformas de nube líderes en el mercado, que ofrece una amplia gama de servicios y herramientas para los desarrolladores y los equipos de operaciones.

En este contexto, la automatización de la infraestructura en la nube se ha convertido en una práctica clave para la agilización de los procesos, la reducción de errores, reducción de costes en el despliegue de recursos y lo más importante la flexibilidad a la hora de escalar los recursos, es decir, si se necesita crecer en número de máquinas disponibles, CPUs, RAM, ROM, etc, con pulsar un simple botón toda la infraestructura montada se auto escala en cuestión de unos pocos de minutos. Esta flexibilidad es clave hoy en día para las empresas, dada la rapidez, sencillez y espacio físico que se ahorran las empresas.

En este proyecto se realiza la creación de un automatismo con Python que generará toda la infraestructura Cloud necesaria para tener una subscripción en la plataforma que nos permitirá ir creando servicios asociados a esta suscripción, un servidor de base de datos, un servicio de base de datos SQL Server añadiéndole una base de datos de ejemplo que proporciona Microsoft.

En resumen, este proyecto busca contribuir al campo de la automatización de infraestructuras en plataformas Cloud, al tiempo que proporciona una solución práctica y eficaz para el aprendizaje profundo del lenguaje Python y algunas librerías clave para la automatización de tareas rutinarias que habitualmente realizan personas, lo que permite un ahorro de recursos para empresas.

Objetivo del proyecto

El objetivo principal de este proyecto es dar a conocer el potencial de ciertas librerías de Python para la automatización de tareas (ver Anexo III para aprender a utilizar Python). Explotándolo de manera adecuada, ayuda a las empresas a ahorrar tiempo de personas que realmente se puedan dedicar a otras cosas que las máquinas no puedan hacer y poder enfocar el tiempo disponible en otras tareas más valiosas.

Este proyecto se enfocará en proporcionar a los alumnos una plataforma ya configurada en pocos minutos. Porque la primera vez que se intenta trabajar con una plataforma de estas características es complicado saber los pasos que hay que dar o configurar para tener las herramientas que se desean y se podrían tardar varios días en conseguirlo. De esta forma ya queda todo automatizado en pocos minutos, si lo que se desea es trabajar con servicios de Cloud de base de datos. Además, se pretende explorar las capacidades y limitaciones de Azure como plataforma de nube, y analizar el impacto de la automatización en la gestión de la infraestructura en la nube.

Otro de los objetivos, es familiarizar a los estudiantes con la plataforma de nube de Azure y enseñarles los conceptos básicos de la infraestructura en la nube, y su importancia en el mundo empresarial actual.

Instalaciones previas necesarias

En primer lugar, debemos comprobar con qué tipo de procesador y sistema operativo vamos a trabajar, para poder realizar dicha comprobación desde el panel de control se verifica esa información en nuestro ordenador, como se puede visualizar en la siguiente imagen en nuestro caso se trata de un procesador de 64 bit con Windows 10 como sistema operativo.

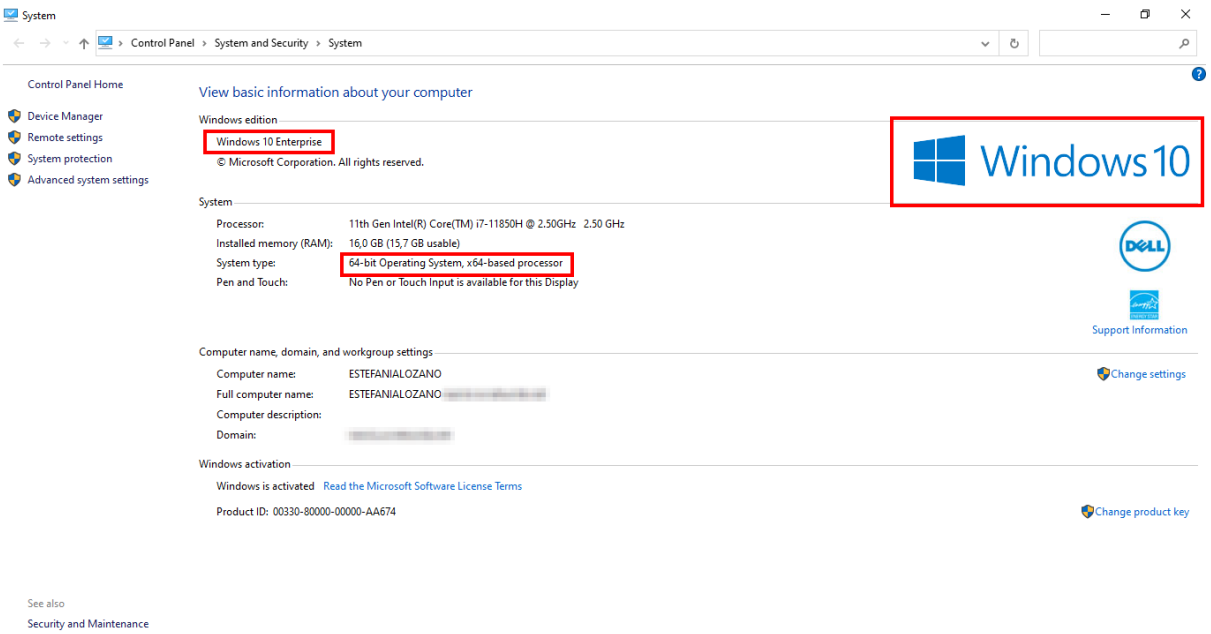

Ilustración 1: Imagen del sistema de nuestro Panel de Control.

No es necesario un ordenador de última generación para poder llevar a cabo este proyecto, dado que Python se adapta a ordenadores básicos sin grandes prestaciones. Pero si es necesario conocer nuestro ordenador donde instalaremos el Python para saber que versión instalarnos.

Tras dicha comprobación se comienza con todas las descargas de los programas necesarios para realizar este proyecto.

Enlace de descarga de Python:

- https://www.Python.org/downloads/ (1)

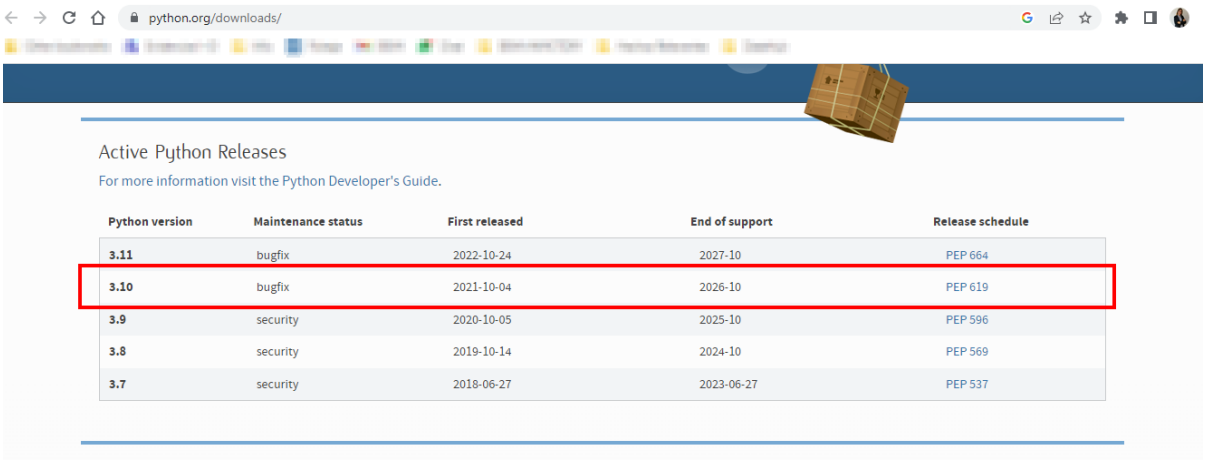

Ilustración 2: Imagen de la versión de Python que vamos a descargar/utilizar.

Enlace de la versión de Python que vamos a descargar para nuestro Windows 64 bits:

- https://www.Python.org/downloads/release/Python-3109/ (2)

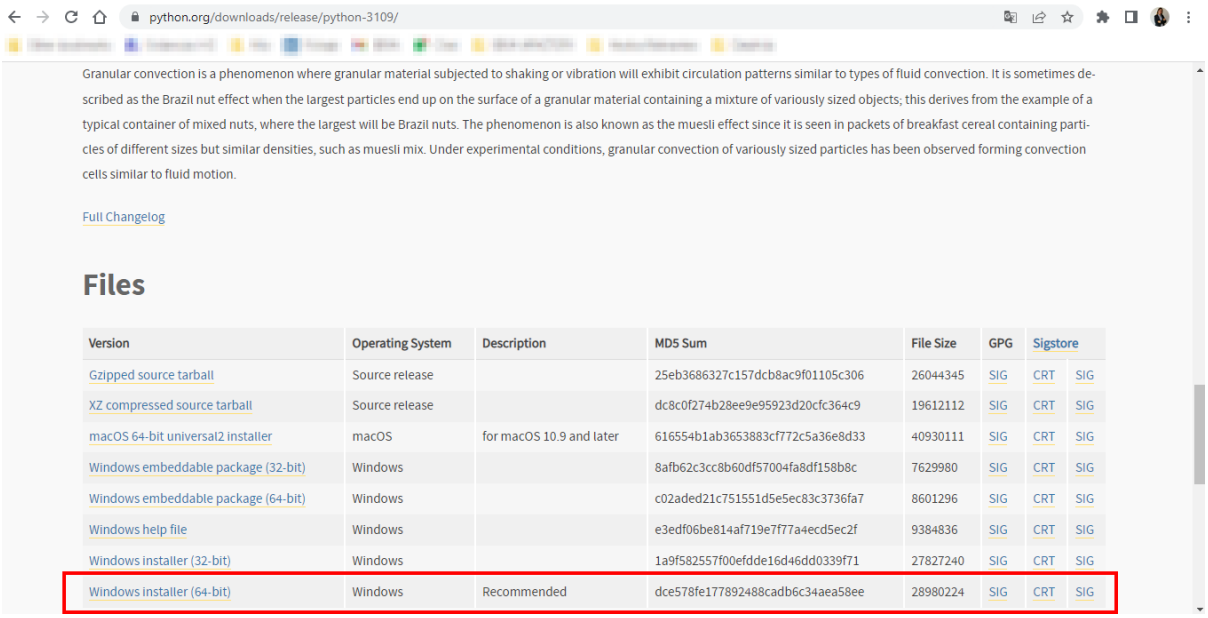

Ilustración 3: Versión de Python para Windows (64 bits) que vamos a descargar.

Dependiendo de cada ordenador se deberá tener en cuenta si tiene un procesador de 32 o **64** bits y si su sistema operativo es **Windows**, macOS u otro.

En este caso se procede a descargar la versión **3.10.9** de Python. Una vez descargado, se comienza a instalar el programa en su versión requerida.

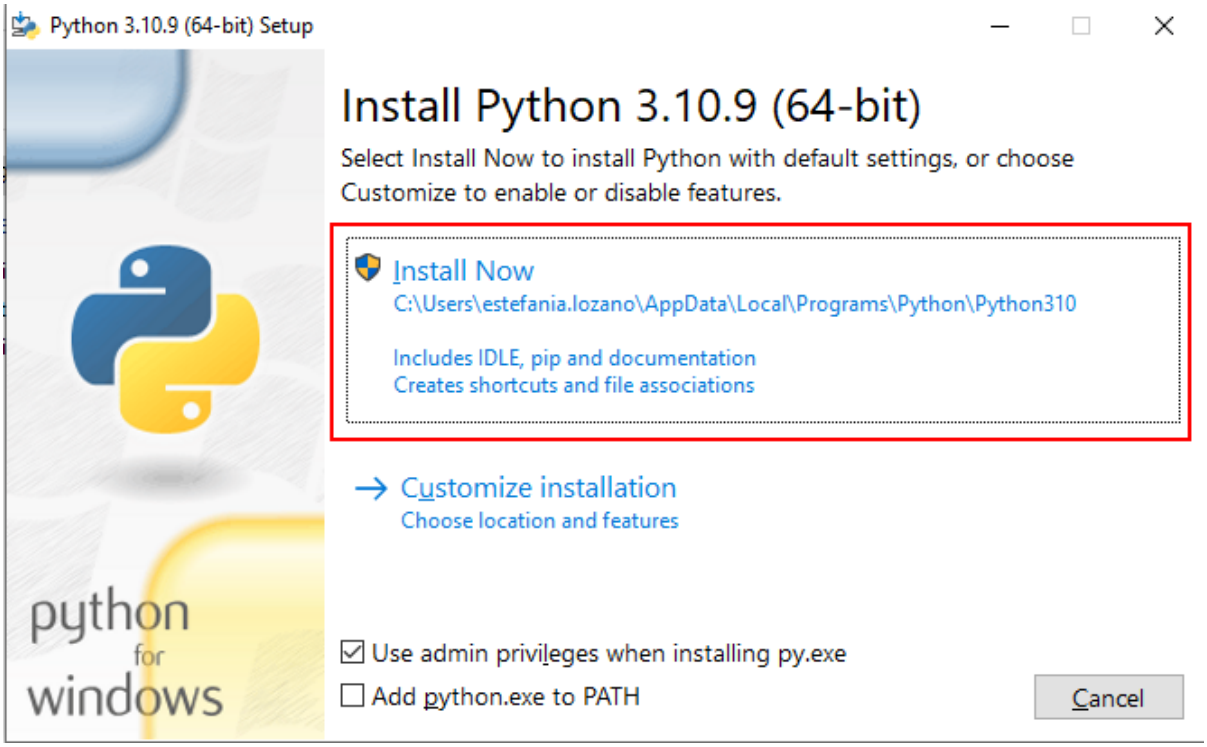

Ilustración 4: Instalación de Python 3.10.9.

Una vez instalado, se abre el programa que en nuestro caso se encuentra en la siguiente ruta:

- C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310

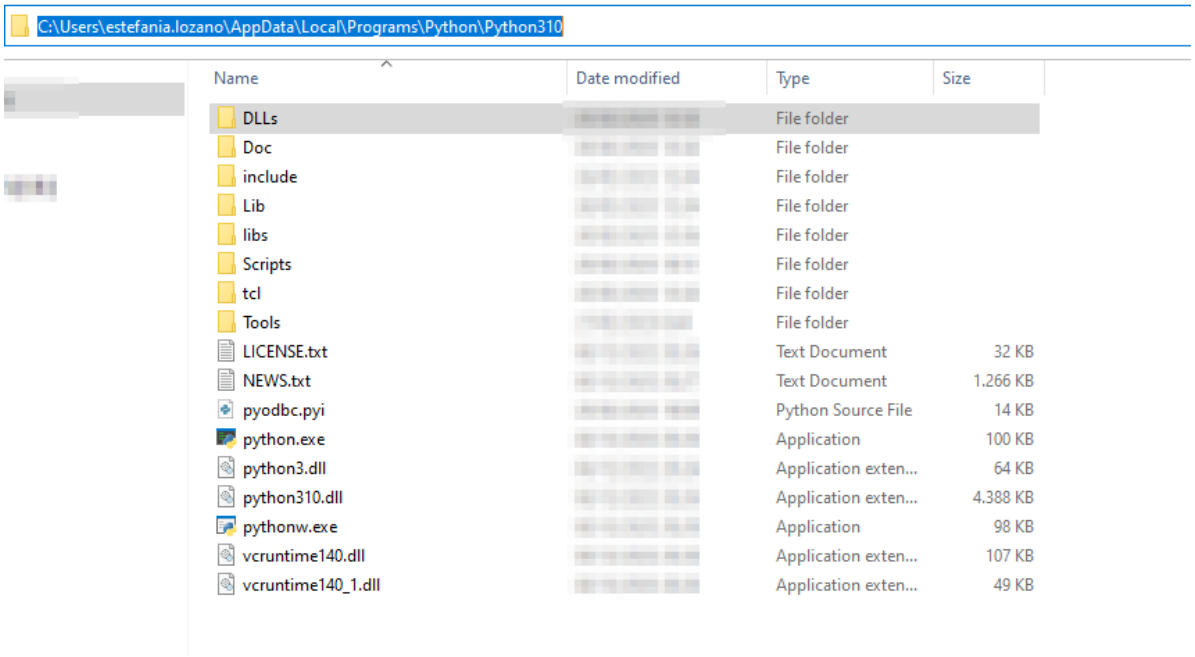

Ilustración 5: Ruta local del pc donde hemos instalado Python.

Abrimos una consola de comandos poniendo en el buscador del pc:

- **cmd**

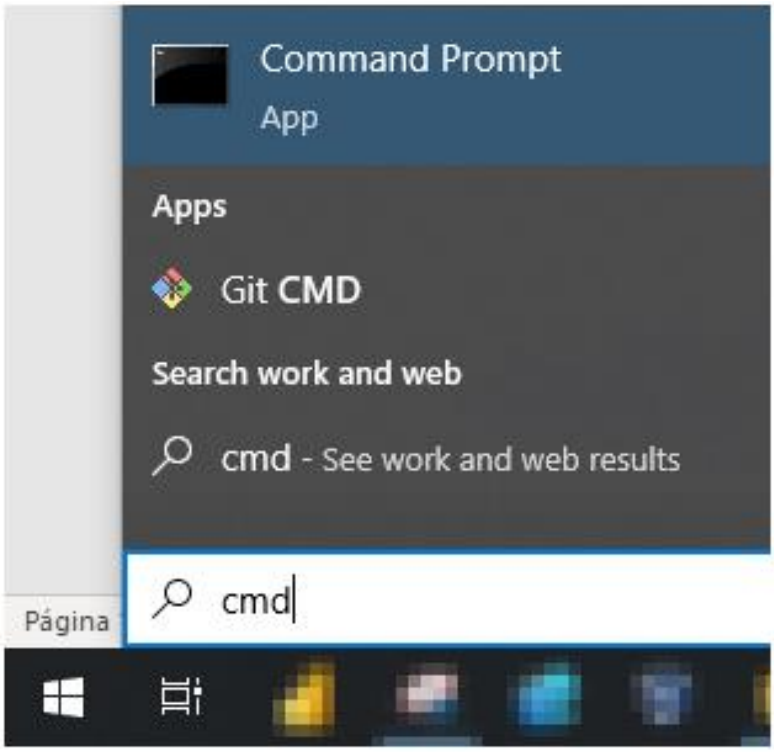

Ilustración 6: Ventana de comandos.

Y una vez dentro, nos cambiamos al directorio donde se encuentran los archivos del programa instalados, con el siguiente comando:

- **cd C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310**

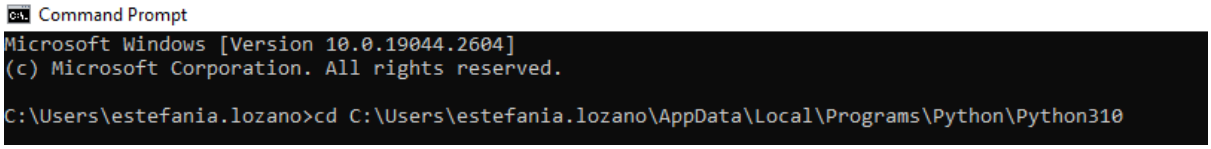

Ilustración 7: Comando para situarnos en la ruta donde se va a encontrar el código a ejecutar.

*Con el comando **dir** nos muestra el contenido completo del directorio.*

- **dir**

```
C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310> dir
 Volume in drive C has no label.
 Volume Serial Number is 9A00-5465

 Directory of C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310

[redacted]    <DIR>          .
[redacted]    <DIR>          ..
[redacted]    <DIR>          DLLs
[redacted]    <DIR>          Doc
[redacted]    <DIR>          include
[redacted]    <DIR>          Lib
[redacted]    <DIR>          libs
[redacted]           32.762 LICENSE.txt
[redacted]        1.295.384 NEWS.txt
[redacted]           13.620 pyodbc.pyi
[redacted]          101.760 python.exe
[redacted]           64.896 python3.dll
[redacted]        4.492.664 python310.dll
[redacted]          100.216 pythonw.exe
[redacted]    <DIR>          Scripts
[redacted]    <DIR>          tcl
[redacted]    <DIR>          Tools
[redacted]          109.392 vcruntime140.dll
[redacted]           49.488 vcruntime140_1.dll
               9 File(s)      6.260.182 bytes
              10 Dir(s)  328.534.642.688 bytes free

C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310>
```

Ilustración 8: Contenido completo del directorio tras la instalación de Python310, donde vamos a implementar el código a ejecutar.

Una vez instalado Python lo primero que vamos a hacer es instalar un driver para poder controlar el explorador web Chrome, como en este caso hemos instalado la versión de Python 3.10.9 necesitamos descargar el driver acorde a la instalación de Python que tenemos.

Enlace de la versión del Driver que controla Chrome:

- https://chromedriver.chromium.org/downloads [(3)]

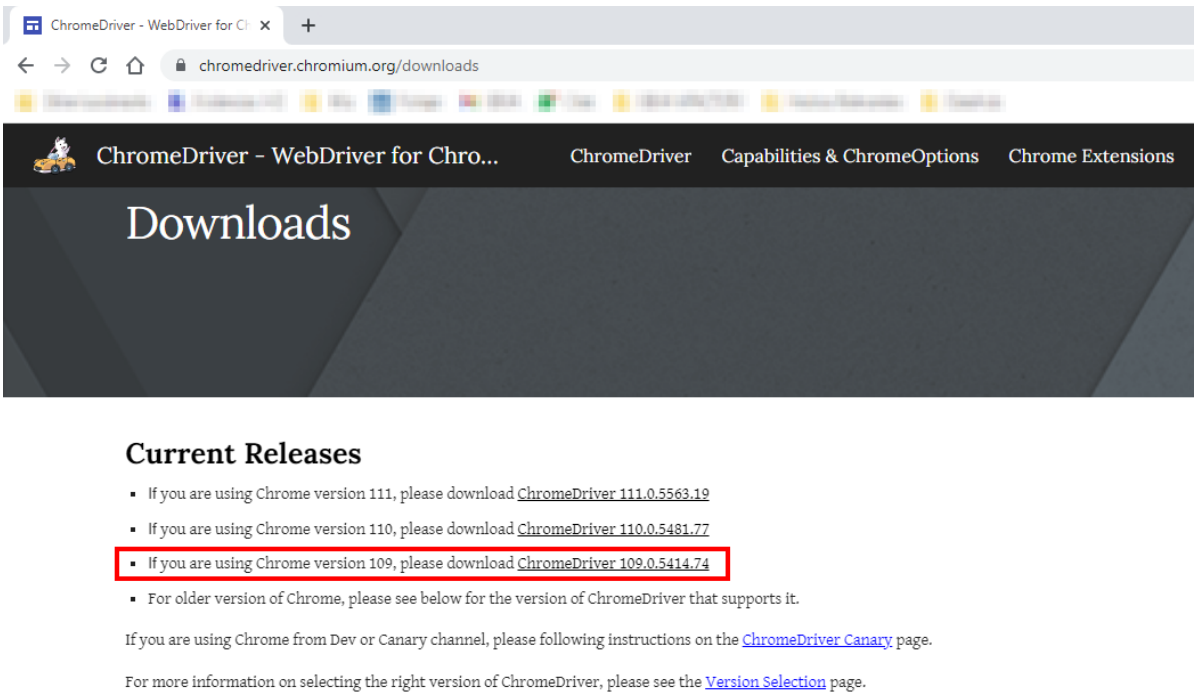

Ilustración 9: Versión del driver que necesitamos instalar según nuestro Python instalado.

Enlace de la versión del driver controlador del Chrome que descargamos para Windows:

- https://chromedriver.storage.Googleapis.com/index.html?path=109.0.5414.74/ (4)

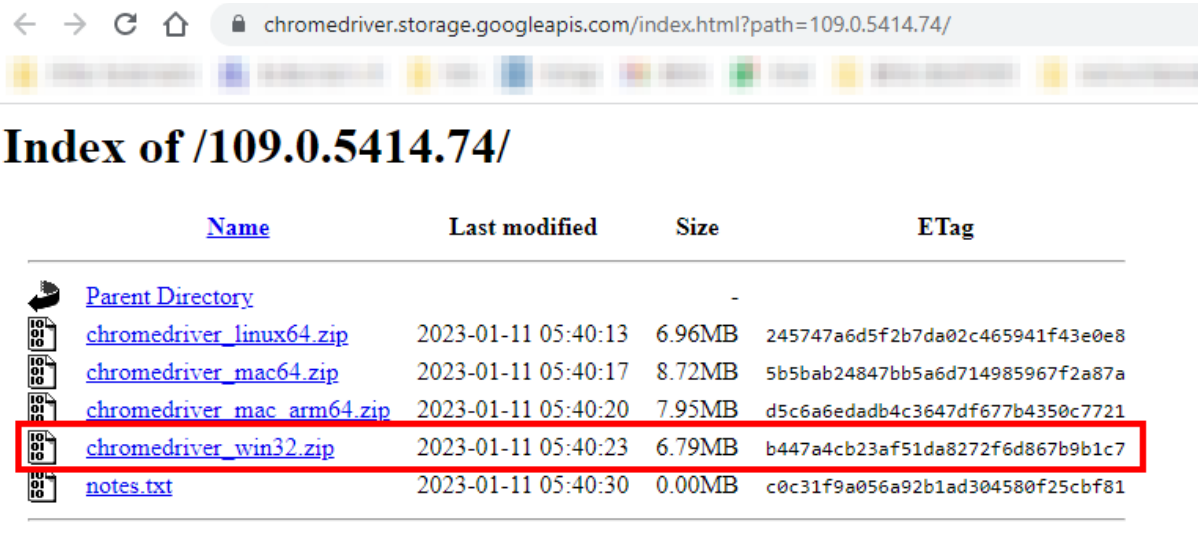

Ilustración 10: Versión del driver controlador del Chrome que descargamos para Windows.

Se descarga en la siguiente ruta:

- C:\Users\estefania.lozano\Downloads\chromedriver_win32

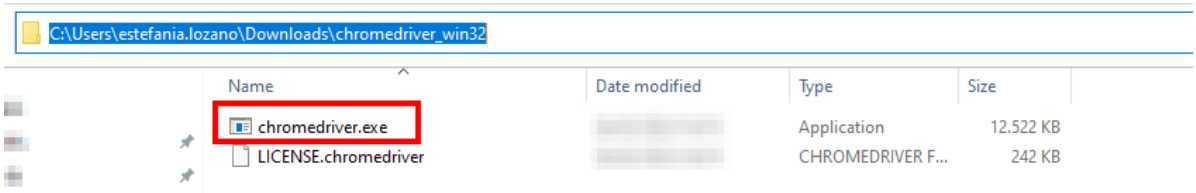

Ilustración 11: Carpeta de descargas, donde podemos recoger el driver descargado para controlar Chrome.

Una vez obtenido el driver hay que llevarlo a la carpeta donde tenemos el Python:

- **C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310**

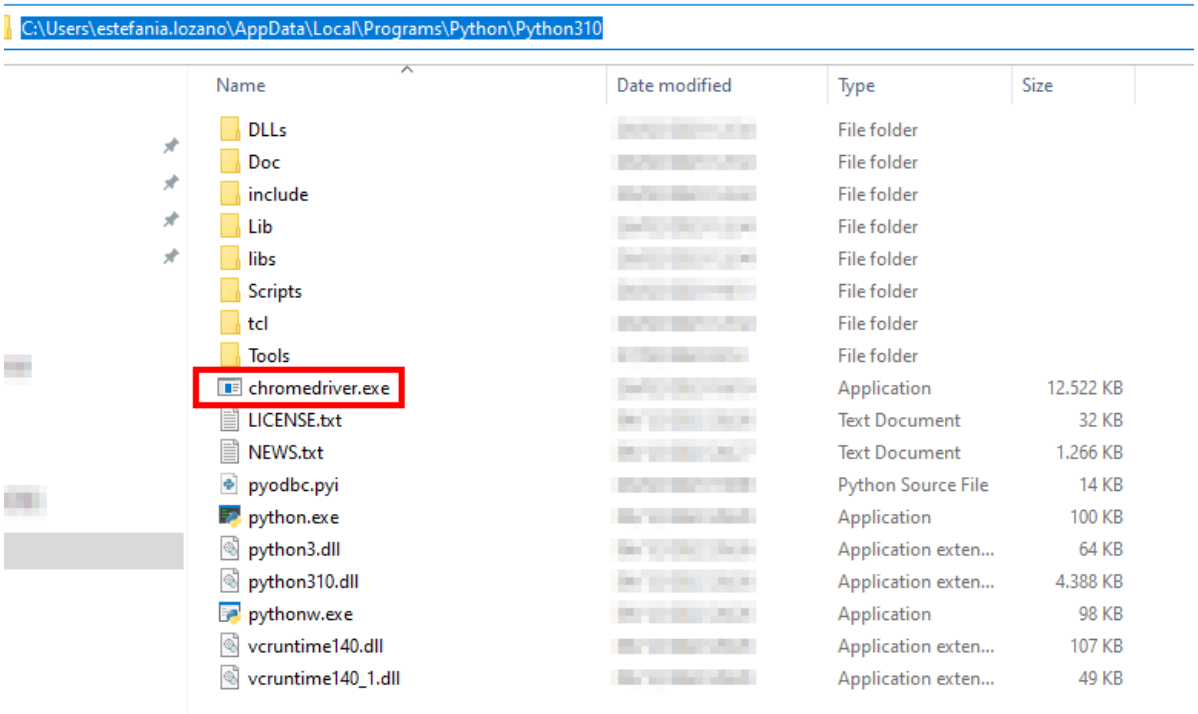

Ilustración 12: Trasladado el driver necesario para controlar Chrome a la carpeta que instalamos de Python.

Ahora como primer paso, vamos a instalar la primera librería en Python que necesitamos, en concreto la librería “Selenium”.

¿Qué es Selenium?

Selenium es una herramienta de automatización que permite simular las interacciones de un usuario con una aplicación web. Es una biblioteca que permite a los desarrolladores escribir scripts en Python para automatizar pruebas funcionales y de integración de aplicaciones web. Selenium utiliza un controlador de navegador (como Chrome Driver, Firefox Driver o Edge Driver) para interactuar con el navegador y simular las acciones de un usuario, como hacer clic en botones, completar formularios y navegar por diferentes páginas web. Esto hace posible la automatización de pruebas repetitivas y la validación de múltiples escenarios de uso.
Además, Selenium es compatible con una amplia variedad de lenguajes de programación y plataformas de sistema operativo, lo que lo hace una herramienta versátil y popular para la automatización de pruebas.

Para instalar la librería Selenium, desde la ventana de comandos nos posicionamos sobre la carpeta de Script y ejecutamos el siguiente comando:

- **pip.exe install -U Selenium**

```
C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts>pip.exe install -U selenium
```

Ilustración 13: Instalación de la librería Selenium.

```
Collecting Selenium
  Downloading Selenium-4.8.2-py3-none-any.whl (6.9 MB)
     ---------------------------------------- 6.9/6.9 MB 6.0 MB/s eta 0:00:00
Collecting trio-websocket~=0.9
  Downloading trio_websocket-0.9.2-py3-none-any.whl (16 kB)
Collecting certifi>=2021.10.8
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB)
     ---------------------------------------- 155.3/155.3 kB 9.1 MB/s eta 0:00:00
Collecting urllib3[socks]~=1.26
  Downloading urllib3-1.26.14-py2.py3-none-any.whl (140 kB)
     ---------------------------------------- 140.6/140.6 kB 8.7 MB/s eta 0:00:00
Collecting trio~=0.17
  Downloading trio-0.22.0-py3-none-any.whl (384 kB)
     ---------------------------------------- 384.9/384.9 kB 6.0 MB/s eta 0:00:00
Collecting idna
  Downloading idna-3.4-py3-none-any.whl (61 kB)
     ---------------------------------------- 61.5/61.5 kB ? eta 0:00:00
Collecting async-generator>=1.9
  Downloading async_generator-1.10-py3-none-any.whl (18 kB)
Collecting attrs>=19.2.0
  Downloading attrs-22.2.0-py3-none-any.whl (60 kB)
     ---------------------------------------- 60.0/60.0 kB ? eta 0:00:00
Collecting cffi>=1.14
  Downloading cffi-1.15.1-cp310-cp310-win_amd64.whl (179 kB)
     ---------------------------------------- 179.1/179.1 kB 11.3 MB/s eta 0:00:00
Collecting sortedcontainers
```

```
  Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl (29 kB)
Collecting sniffio
 Downloading sniffio-1.3.0-py3-none-any.whl (10 kB)
Collecting outcome
  Downloading outcome-1.2.0-py2.py3-none-any.whl (9.7 kB)
Collecting exceptiongroup>=1.0.0rc9
  Downloading exceptiongroup-1.1.0-py3-none-any.whl (14 kB)
Collecting wsproto>=0.14
  Downloading wsproto-1.2.0-py3-none-any.whl (24 kB)
Collecting PySocks!=1.5.7,<2.0,>=1.5.6
  Downloading PySocks-1.7.1-py3-none-any.whl (16 kB)
Collecting pycparser
  Downloading pycparser-2.21-py2.py3-none-any.whl (118 kB)
     ---------------------------------------- 118.7/118.7 kB 6.8 MB/s eta 0:00:00
Collecting h11<1,>=0.9.0
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ---------------------------------------- 58.3/58.3 kB ? eta 0:00:00
Installing collected packages: sortedcontainers, urllib3, sniffio, PySocks, pycparser, idna, h11, exceptiongroup, certifi,
attrs, async-generator, wsproto, outcome, cffi, trio, trio-websocket, Selenium
Successfully installed PySocks-1.7.1 async-generator-1.10 attrs-22.2.0 certifi-2022.12.7 cffi-1.15.1 exceptiongroup-1.1.0
h11-0.14.0 idna-3.4 outcome-1.2.0 pycparser-2.21 Selenium-4.8.2 sniffio-1.3.0 sortedcontainers-2.4.0 trio-0.22.0 trio-
websocket-0.9.2 urllib3-1.26.14 wsproto-1.2.0

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Python.exe -m pip
install --upgrade pip
```

Una vez instalado "Selenium" instalamos el driver para Chrome dentro de Python "webdriver" de manera similar a la librería anterior:

- **pip install webdrivermanager**

```
C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts>pip install webdrivermanager
```

Ilustración 14: Instalación de webdrivermanager de la librería Selenium.

```
Collecting webdrivermanager
  Downloading webdrivermanager-0.10.0.tar.gz (33 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting appdirs
  Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Collecting lxml
  Downloading lxml-4.9.2-cp310-cp310-win_amd64.whl (3.8 MB)
     ---------------------------------------- 3.8/3.8 MB 6.0 MB/s eta 0:00:00
Collecting tqdm
  Downloading tqdm-4.64.1-py2.py3-none-any.whl (78 kB)
     ---------------------------------------- 78.5/78.5 kB ? eta 0:00:00
Collecting BeautifulSoup4
  Downloading beautifulsoup4-4.11.2-py3-none-any.whl (129 kB)
     ---------------------------------------- 129.4/129.4 kB 7.4 MB/s eta 0:00:00
Collecting requests
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
     ---------------------------------------- 62.8/62.8 kB 3.5 MB/s eta 0:00:00
```

```
Collecting soupsieve>1.2
Downloading soupsieve-2.4-py3-none-any.whl (37 kB)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from requests-
>webdrivermanager) (2022.12.7)
Collecting charset-normalizer<4,>=2
  Downloading charset_normalizer-3.0.1-cp310-cp310-win_amd64.whl (96 kB)
     ---------------------------------------- 96.5/96.5 kB ? eta 0:00:00
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from requests-
>webdrivermanager) (1.26.14)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from requests-
>webdrivermanager) (3.4)
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Building wheels for collected packages: webdrivermanager
  Building wheel for webdrivermanager (pyproject.toml) ... done
  Created wheel for webdrivermanager: filename=webdrivermanager-0.10.0-py2.py3-none-any.whl size=19550
sha256=2dab66243f0abf887739e83f736991f09ef85cabc3f9545036b580a7dc2dd351
  Stored in directory:
c:\users\estefania.lozano\appdata\local\pip\cache\wheels\c9\6f\d6\7170bf4b2fcee03185c0ea011debd69348a979bad
a5dd06bc6
Successfully built webdrivermanager
Installing collected packages: charset-normalizer, appdirs, soupsieve, requests, lxml, colorama, tqdm, BeautifulSoup4,
webdrivermanager
  WARNING: The script normalizer.exe is installed in
'C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script tqdm.exe is installed in
'C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script webdrivermanager.exe is installed in
'C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed BeautifulSoup4-4.11.2 appdirs-1.4.4 charset-normalizer-3.0.1 colorama-0.4.6 lxml-4.9.2 requests-
2.28.2 soupsieve-2.4 tqdm-4.64.1 webdrivermanager-0.10.0

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Python.exe -m pip
install --upgrade pip
```

- **pip install webdriver_manager**

```
C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts>pip install webdriver_manager
```

Ilustración 15: Instalación de webdriver_manager de la librería Selenium.

```
Collecting webdriver_manager
  Downloading webdriver_manager-3.8.5-py2.py3-none-any.whl (27 kB)
Collecting Python-dotenv
  Downloading Python_dotenv-1.0.0-py3-none-any.whl  (19 kB)
Requirement already satisfied: requests in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from webdriver_manager)
(2.28.2)
Collecting packaging
  Downloading packaging-23.0-py3-none-any.whl (42 kB)
```

```
     ---------------------------------------- 42.7/42.7 kB 690.2 kB/s eta 0:00:00
Requirement already satisfied: tqdm in c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-
packages (from webdriver_manager) (4.64.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from requests-
>webdriver_manager) (1.26.14)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from requests-
>webdriver_manager) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from requests-
>webdriver_manager) (2022.12.7)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from requests-
>webdriver_manager) (3.0.1)
Requirement already satisfied: colorama in
c:\users\estefania.lozano\appdata\local\programs\Python\Python310\lib\site-packages (from tqdm-
>webdriver_manager) (0.4.6)
Installing collected packages: Python-dotenv, packaging, webdriver_manager
  WARNING: The script dotenv.exe is installed in
'C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed packaging-23.0 Python-dotenv-1.0.0 webdriver_manager-3.8.5

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Python.exe -m pip
install --upgrade pip
```

Con esto ya nos permitirá controlar el navegador web de Chrome mediante comandos Python desde nuestro script. Como primera prueba vamos a abrir el navegador y que nos lleve a la página web de www.Google.es [5] por ejemplo.

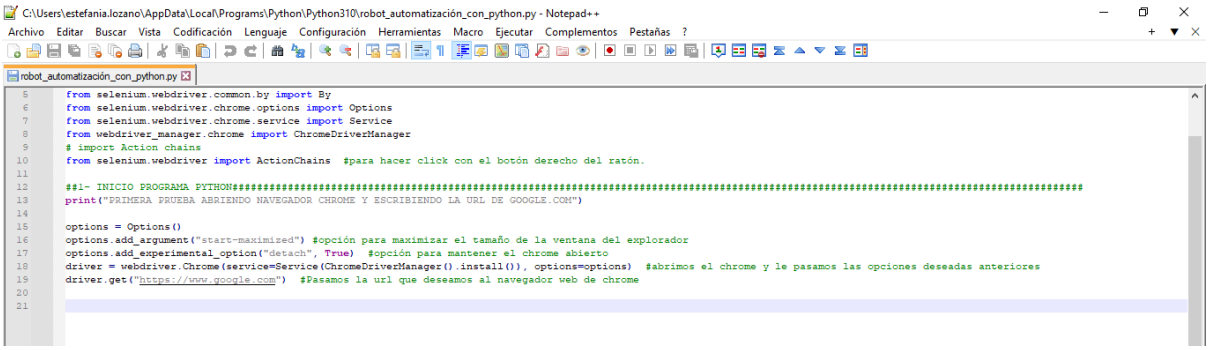

Ilustración 16: Prueba de abrir el navegador de Chrome y que navegue hasta la página Google.

Ejecutamos el script anterior con el siguiente comando (comando de la versión definitiva entregada, distinta a la de las pruebas) para comprobar que nos lleva a la ruta establecida.

- **python robot_automatización_con_python_Final.py**

```
C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310>.\python.exe robot_automatización_con_python.py
```

Ilustración 17: Ejecución de nuestro código para realizar la prueba de entrar en Google.

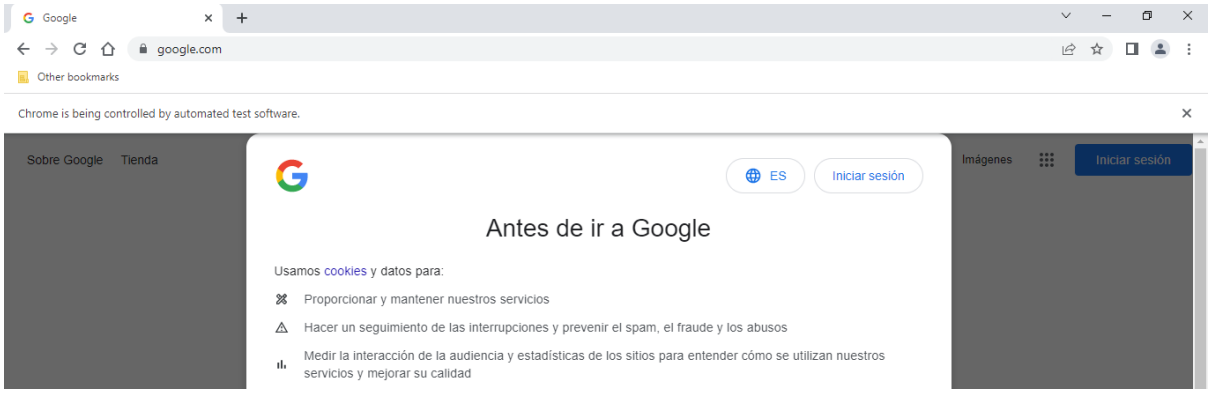

Ilustración 18: Demostración de que nuestro código abre una página del Chrome para ir a Google.

Verificamos que efectivamente funciona abrir el Chrome y pasarle la URL que deseamos abrir. Notad que en el propio navegador nos aparece un mensaje que nos indica que el navegador está siendo controlado por un software (en este caso, por Chromedriver.exe):

"*Chrome is being controlled by automated test software*".

Ahora vamos a conectar con nuestro servicio de base de datos SQL Server de Azure desde Python para poder ejecutar comandos SQL y utilizar la información que nos devuelva dentro de nuestro script o programa.

Para ello, lo primero es instalar y añadir las librerías necesarias en Python.

- **pip install pyodbc**

```
C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts>pip install pyodbc
```

Ilustración 19: Instalación de librerías de Python.

```
Collecting pyodbc
  Downloading pyodbc-4.0.35-cp310-cp310-win_amd64.whl (66 kB)
     ---------------------------------------- 66.0/66.0 kB 3.7 MB/s eta 0:00:00
Installing collected packages: pyodbc
Successfully installed pyodbc-4.0.35

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Python.exe -m pip install --upgrade pip
```

Ahora descargamos e instalamos el driver ODBC de Microsoft para servidores SQL Server.

Enlace de descarga del fichero de los drivers de Microsoft para SQL:

- https://learn.Microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-SQL-Server?view=SQL-Server-ver16#download-for-Windows [6]

Enlace de descarga directa de Microsoft OBDC for SQL:

- Download Microsoft ODBC Driver 17 for SQL Server (x64) [7]

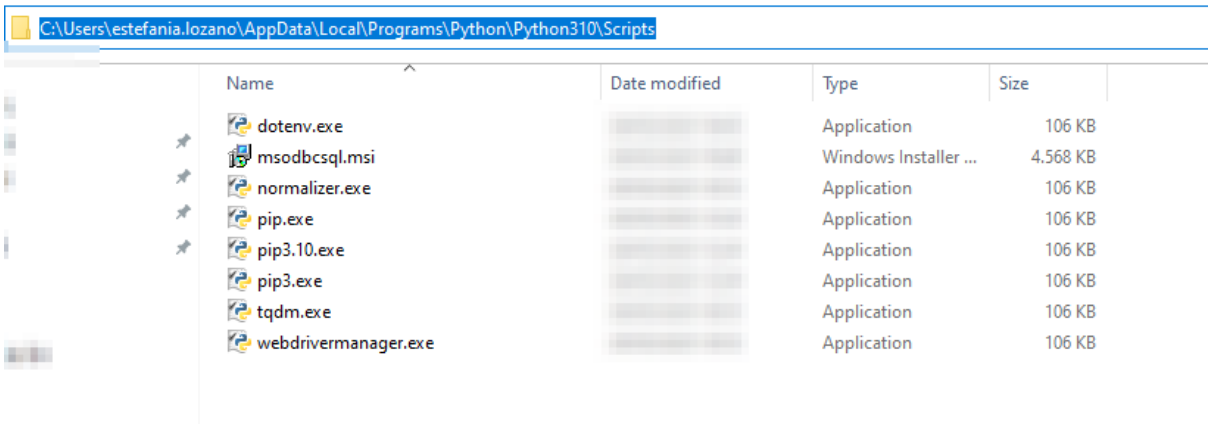

Ilustración 20: Descarga del fichero de los drivers de Microsoft para SQL.

En la consola ejecutamos el siguiente comando:

- **msiexec /i msodbcSQL.msi ADDLOCAL=ALL**

```
C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310\Scripts>msiexec /i msodbcsql.msi ADDLOCAL=ALL
```

Ilustración 21: Comando para instalar los drivers de Microsoft para SQL.

Enlace para la instalación de los drivers de Microsoft para SQL Server:

- https://learn.Microsoft.com/en-us/SQL/connect/odbc/Windows/system-requirements-installation-and-driver-files?view=SQL-Server-ver16#installing-Microsoft-odbc-driver-for-SQL-Server [8]

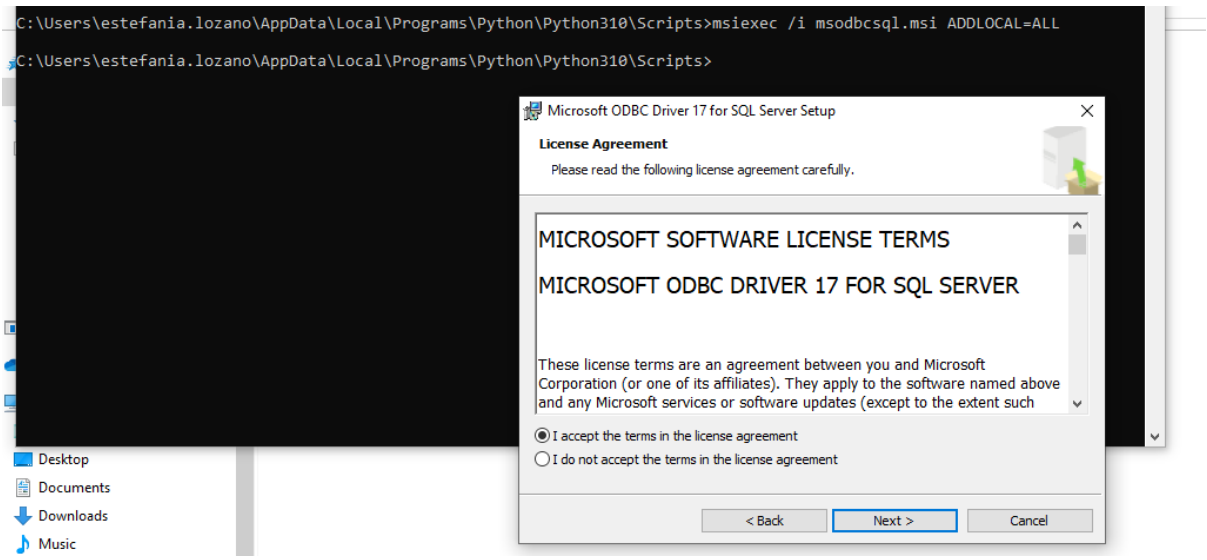

Ilustración 22: Instalación de los drivers de Microsoft para SQL.

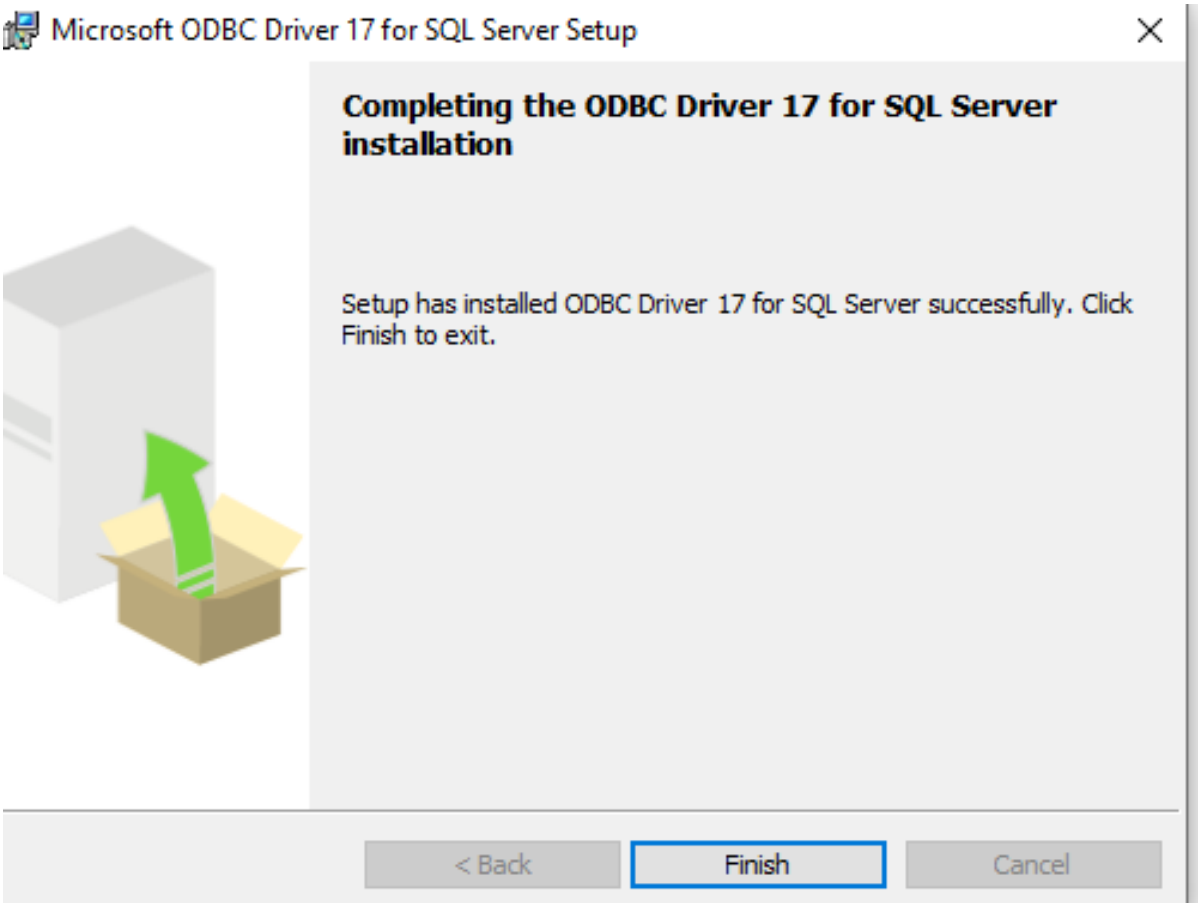

Ilustración 23: Finalización de la instalación de los drivers de Microsoft para SQL.

Una vez finalizada la instalación de los drivers, comprobamos mediante un script que podemos conectarnos a la base de datos sin problemas y hacer una consulta SQL sobre una tabla.

```
##0- IMPORTAMOS LAS LIBRERIAS QUE NECESITAMOS DE PYTHON##########################################################################################################################
from selenium import webdriver
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
# import Action chains
from selenium.webdriver import ActionChains  #para hacer click con el botón derecho del ratón.
import pyodbc #para conectar a base de datos, para utilizar esta libería es necesario instalar:  pip install pyodbc

##1- INICIO PROGRAMA PYTHON######################################################################################################################################################
print("Paso 1: PRIMERA PRUEBA ABRIENDO NAVEGADOR CHROME Y ESCRIBIENDO LA URL DE GOOGLE.COM")

options = Options()
options.add_argument("start-maximized") #opción para maximizar el tamaño de la ventana del explorador
options.add_experimental_option("detach", True)  #opción para mantener el chrome abierto
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)  #abrimos el chrome y le pasamos las opciones deseadas anteriores
driver.get("https://www.google.com")  #Pasamos la url que deseamos al navegador web de chrome

server = 'servidortfgteleco.database.windows.net'
database = 'bbdd_tfg'
username = 'usuadmin'
password = 'Fuente2020'
driverSQL= '{ODBC Driver 17 for SQL Server}'

connection=pyodbc.connect('DRIVER='+driverSQL+';SERVER=tcp:'+server+';PORT=1433;DATABASE='+database+';UID='+username+';PWD='+ password) # Connection string
cursor = connection.cursor()
print("Paso 2: Conexión con base de datos Azure PowerBI establecida")

cursor.execute("SELECT TOP (10) [ProductID] ,[Name],[ProductNumber],[Color],[StandardCost],[ListPrice],[Size],[Weight],[ProductCategoryID],[ProductModelID] FROM [SalesLT].[Product]"
) # Executing a query

print("Paso 3: recorremos el cursor para ir mostrando registro a registro el resultado del SQL")

for row in cursor: # Looping over returned rows and printing them

    print("Paso 4: Mostramos el resultado de la SQL: " + str(row[0]) + "   " + str(row[1]) + "   " + str(row[2]) + "   " + str(row[3]) + "   " + str(row[4]) + "   " + str(row[5]) +
      "   " + str(row[6]) + "   " + str(row[7]) + "   " + str(row[8]) + "   " + str(row[9]))
```

Ilustración 24: Parte del código donde vamos a ejecutar la prueba de conexión con una base de datos.

```
C:\Users\estefania.lozano\AppData\Local\Programs\Python\Python310>.\python.exe robot_automatización_con_python.py
Paso 1: PRIMERA PRUEBA ABRIENDO NAVEGADOR CHROME Y ESCRIBIENDO LA URL DE GOOGLE.COM

DevTools listening on ws://127.0.0.1:59713/devtools/browser/79ac4531-0772-4759-9adf-6602e003af99
Paso 2: Conexión con base de datos Azure PowerBI establecida
Paso 3: recorremos el cursor para ir mostrando registro a registro el resultado del SQL
Paso 4: Mostramos el resultado de la SQL: 680   HL Road Frame - Black, 58   FR-R92B-58   Black   1059.3100   1431.5000   58   1016.04   18   6
Paso 4: Mostramos el resultado de la SQL: 706   HL Road Frame - Red, 58   FR-R92R-58   Red   1059.3100   1431.5000   58   1016.04   18   6
Paso 4: Mostramos el resultado de la SQL: 707   Sport-100 Helmet, Red   HL-U509-R   Red   13.0863   34.9900   None   None   35   33
Paso 4: Mostramos el resultado de la SQL: 708   Sport-100 Helmet, Black   HL-U509   Black   13.0863   34.9900   None   None   35   33
Paso 4: Mostramos el resultado de la SQL: 709   Mountain Bike Socks, M   SO-B909-M   White   3.3963   9.5000   M   None   27   18
Paso 4: Mostramos el resultado de la SQL: 710   Mountain Bike Socks, L   SO-B909-L   White   3.3963   9.5000   L   None   27   18
Paso 4: Mostramos el resultado de la SQL: 711   Sport-100 Helmet, Blue   HL-U509-B   Blue   13.0863   34.9900   None   None   35   33
Paso 4: Mostramos el resultado de la SQL: 712   AWC Logo Cap   CA-1098   Multi   6.9223   8.9900   None   None   23   2
Paso 4: Mostramos el resultado de la SQL: 713   Long-Sleeve Logo Jersey, S   LJ-0192-S   Multi   38.4923   49.9900   S   None   25   11
Paso 4: Mostramos el resultado de la SQL: 714   Long-Sleeve Logo Jersey, M   LJ-0192-M   Multi   38.4923   49.9900   M   None   25   11
```

Ilustración 25: Demostración de la ejecución de la primera prueba entrando al navegador y conectándonos a una base de datos de Azure.

Desarrollo

Introducción a la plataforma de Azure.

Microsoft Azure (anteriormente Windows Azure y Azure Services Platform) es una plataforma de computación en la nube creado por Microsoft para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos. Proporciona software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS) y es compatible con muchos lenguajes, herramientas y marcos de programación diferentes, incluidos software y sistemas específicos de Microsoft y de terceros.

Dentro de la propia página web https://Azure.Microsoft.com/ [9] se pueden encontrar vídeos y documentación explicativa de como comenzar a usar o trabajar con esta plataforma.

Detalle de las utilidades de Azure que vamos a utilizar.

Suscripción:

- **¿Qué es?**

  Dentro de Azure, una suscripción es una cuenta de pago que le permite acceder y utilizar los servicios de la plataforma de nube de Azure. Con una suscripción de Azure, puede crear y administrar recursos en la nube, como máquinas virtuales, bases de datos, servicios de aplicaciones, redes, almacenamiento y mucho más.

- **¿Cómo configurarla?**

  Para configurar una suscripción de Azure, se deben seguir los siguientes pasos:

  Vaya a la página principal de Azure en su navegador web y haga clic en "**Comenzar gratis**". Utilizamos esta opción, dado que este proyecto es educativo, si fuera para uso profesional deberíamos utilizar la versión de pago creando la suscripción en la que Azure solicitará una tarjeta de crédito a través de la cual se realizará la facturación y los cobros, según el uso.

  Inicie sesión con su cuenta de Microsoft o cree una nueva si no la tiene.

  Seleccione la oferta de suscripción gratuita de Azure que mejor se adapte a sus necesidades.

Proporcione la información de facturación requerida para crear su cuenta, como su dirección de correo electrónico, número de teléfono y dirección de facturación.

Una vez que haya completado los pasos anteriores, su suscripción de Azure se activará y podrá comenzar a crear y administrar recursos en la nube.

Es importante tener en cuenta que la suscripción de Azure es una cuenta de pago y que se le cobrará en función de su uso de los recursos en la nube. Para evitar cargos no deseados, asegúrese de comprender los precios de los diferentes servicios de Azure y configure límites de gastos y alertas de facturación en su cuenta de Azure.

Grupo de recursos:

- **¿Qué es?**

  Dentro de Azure, un grupo de recursos es un contenedor lógico que agrupa los recursos relacionados de Azure, como máquinas virtuales, bases de datos, redes virtuales y otros servicios en la nube. El uso de un grupo de recursos permite a los usuarios administrar, implementar y monitorear los recursos de manera eficiente y coherente.

  Los grupos de recursos también tienen como finalidad ayudar a mantener organizados los recursos en una estructura jerárquica. Esto permite una mejor organización de los recursos y facilita la gestión y administración de los mismos.

- **¿Cómo crear un grupo de recursos en Azure?**

  Para crear un grupo de recursos en Azure, siga los siguientes pasos:

  Inicie sesión en su cuenta de Azure Portal.

  Seleccione "Grupos de recursos" en el menú de la izquierda.

  Haga clic en el botón "Agregar" para crear un nuevo grupo de recursos.

  Seleccione una suscripción existente de Azure para el grupo de recursos.

  Asigne un nombre descriptivo al grupo de recursos.

  Seleccione una región para el grupo de recursos.

  Haga clic en "Revisar + crear" para confirmar la creación del grupo de recursos.

Una vez que se haya creado un grupo de recursos, puede agregar recursos relacionados a él, como máquinas virtuales, bases de datos, redes virtuales y otros servicios en la nube. También puede administrar y monitorear todos los recursos del grupo de recursos de manera coherente y eficiente.

Importante un mismo nombre de grupo de recursos no se puede utilizar 2 veces, aunque ese grupo de recursos haya sido previamente creado y borrado, la plataforma no te deja volver a crear un grupo de recursos con un nombre utilizado previamente.

En resumen, un grupo de recursos en Azure es un contenedor lógico que agrupa los recursos relacionados en una estructura jerárquica. Su uso facilita la gestión, administración y monitoreo de los recursos de Azure, permitiendo una mejor organización y eficiencia en su uso.

Servidor de base de datos en Azure:

- **¿Qué es?**

  Un servidor de base de datos de Azure **es un servicio** en la nube que permite a los usuarios alojar y administrar sus bases de datos en Azure. Proporciona una solución escalable, segura y de alta disponibilidad para almacenar y administrar datos en la nube.

- **¿Cómo crear y configurar un servidor de base de datos de Azure?**

  Para crear y configurar un servidor de base de datos de Azure, se realizan los siguientes pasos:

  Inicie sesión en su cuenta de Azure Portal.

  Seleccione "Bases de datos SQL" en el menú de la izquierda.

  Haga clic en "Agregar" para crear un nuevo servidor de base de datos.

  Seleccione una suscripción existente de Azure para el servidor de base de datos.

  Asigne un nombre descriptivo al servidor de base de datos.

  Seleccione una región para el servidor de base de datos.

  Cree un grupo de recursos o seleccione uno existente para el servidor de base de datos.

  Seleccione un nivel de servicio para el servidor de base de datos, como Básico, Estándar o Premium.

Cree una cuenta de administrador y una contraseña segura para el servidor de base de datos.

Haga clic en "Revisar + crear" para confirmar la creación del servidor de base de datos.

Una vez que haya creado un servidor de base de datos de Azure, puede configurar y administrar sus bases de datos en el servidor utilizando herramientas como SQL Server Management Studio o Azure Portal.

Puede crear bases de datos en el servidor de base de datos, realizar copias de seguridad y restauraciones de bases de datos, monitorear y optimizar el rendimiento de las bases de datos, y mucho más.

En resumen, un servidor de base de datos de Azure es un servicio en la nube que proporciona una solución escalable, segura y de alta disponibilidad para almacenar y administrar bases de datos en Azure. Su configuración es relativamente sencilla, y una vez creado, permite la administración eficiente de las bases de datos en la nube.

Servicio de base de datos en Azure:

- **¿Qué es?**

  Azure SQL Database es un motor de base de datos de plataforma como servicio (PaaS) totalmente administrado que se encarga de la mayoría de las funciones de administración de bases de datos, como actualizar, aplicar revisiones, crear copias de seguridad y supervisar sin intervención del usuario. Azure SQL Database se ejecuta siempre en la última versión estable del motor de base de datos de SQL Server y en un sistema operativo revisado con el 99,99 % de disponibilidad. Las funcionalidades PaaS integradas en Azure SQL Database permiten centrarse en las actividades de administración y optimización de bases de datos específicas del dominio que son críticas para el negocio.

  Azure SQL proporciona una manera rápida y sencilla de acceder a todos los recursos de SQL en Azure Portal, incluidas bases de datos únicas y agrupadas en Azure SQL Database, así como al servidor lógico que las hospeda, instancias de Azure SQL Managed Instance y SQL Server en máquinas virtuales de Azure. Azure SQL no es un servicio ni un recurso, sino una familia de servicios relacionados con SQL.

- **¿Cómo configurarla?**

  Para configurar un servicio en una base de datos de Azure, se tienen que seguir los siguientes pasos:

Inicie sesión en su cuenta de Azure Portal.

Seleccione "Bases de datos SQL" en el menú de la izquierda.

Seleccione la base de datos en la que desea configurar un servicio.

Seleccione la pestaña "Servicios" en la parte superior de la pantalla.

Para administrar los recursos existentes, seleccione el elemento deseado en la lista. Para crear nuevos recursos de Azure SQL, seleccione **"+ Crear"**. Después de seleccionar **"+ Crear"**, vea información adicional sobre las diferentes opciones al seleccionar **"Mostrar detalles"** en cualquier icono.

Creación de una Base de datos ejemplo proporcionada por Microsoft/Azure:

- **¿Qué es?**

  Es una base de datos útil para dar los primeros pasos con SQL y sus consultas. Proporciona unas tablas de datos que tienen relación entre sí para dar esos primeros pasos. Es decir, es una base de datos relacional.

  Esta base de datos se llama *Adventureworks*.

- **¿Cómo crear/configurar la base de datos de ejemplo en Azure?**

  Es una opción adicional dentro de la configuración cuando estamos creando el servicio de base de datos SQL Server. Para encontrar esta opción es necesario desplazarse hasta la pestaña de "Ajustes adicionales" y seleccionamos la opción de "base de datos de ejemplo de Azure".

Cómo ejecutar consultas a través de la propia plataforma de Azure:

Hay diferentes herramientas para este fin, como podría ser SQL Server Management Studio (SSMS), aunque Microsoft ha creado Azure Data Studio que está más orientada a conectar con los servicios de base de datos de Azure.

Azure Data Studio es una herramienta gratuita de Microsoft para trabajar con bases de datos en la nube de Azure y en otras plataformas de bases de datos. Esta herramienta ofrece una amplia variedad de funcionalidades, como la ejecución de consultas, la creación de gráficos, la administración de bases de datos, exportación de los datos consultados, etc.

- **¿Cómo configurar la plataforma para realizar consultas?**

  Para ejecutar consultas a través de la plataforma de Azure, es posible utilizar Azure Data Studio o la propia interfaz de usuario de Azure Portal. En ambos casos, se pueden realizar consultas SQL a la base de datos seleccionada.

  Para ejecutar consultas a través de Azure Data Studio, siga los siguientes pasos:

  Descargue e instale Azure Data Studio en su ordenador.

  Inicie Azure Data Studio y seleccione "Conectar" en la pantalla de inicio.

  Proporcione los detalles de la conexión a la base de datos:

  - El nombre del servidor donde está alojada la base de datos (obligatorio).
  - Tipo de autenticación (obligatorio).
  - Nombre de la base de datos (opcional).
  - Las credenciales: usuario y contraseña (obligatorios)

  Haga clic en "Conectar".

  Abra una nueva pestaña de consulta haciendo clic en "Nuevo archivo" en la barra de herramientas.

  Escriba la consulta SQL que desea ejecutar en la pestaña de consulta del estilo SELECT * FROM *NOMBRE_TABLA_A_CONSULTAR*.

  Haga clic en "Ejecutar" para ejecutar la consulta.

  En la plataforma de Azure Portal, para ejecutar consultas a través de la propia interfaz de usuario, continue con los siguientes pasos:

  Seleccione la base de datos a la que desea conectarse en la lista de bases de datos en su cuenta de Azure Portal.

  Seleccione la pestaña "Consulta" en la parte superior de la pantalla.

  Escriba la consulta SQL que desea ejecutar en el cuadro de texto.

  Haga clic en "Ejecutar" para ejecutar la consulta.

  En resumen, Azure Data Studio es una herramienta de Microsoft que se utiliza para trabajar con bases de datos en Azure y en otras plataformas de bases de datos. Para ejecutar consultas a través de la plataforma de Azure, es posible utilizar tanto Azure Data Studio como la propia interfaz de usuario de Azure Portal, y en ambos casos se pueden realizar consultas SQL a la base de datos seleccionada.

Robot automatización en Python para montar la infraestructura en Azure de los Servicios deseados.

En este apartado, procederemos a explicar el código Python implementado.

Pasos de la demostración de lo que realiza el código implementado de manera didáctica para que un alumno pueda crearse una base de datos de pruebas de Azure para realizar sus tareas, **se necesita que el alumno esté dado de alta en la base de datos de la UPCT y tenga acceso al portal web de la Universidad**. Esto nos garantiza que el alumno con su DNI y una contraseña pueda acceder a dicha web. Ya que **dentro de esta web de la universidad podremos localizar el email educativo que será necesario para la configuración gratuita dentro de la plataforma de Azure**.

Pasos del código desarrollado:

- En el primer paso del software desarrollado, se procede a abrir el navegador web Chrome.

- A continuación, en este segundo paso, el programa se dirige a la web de la universidad. Dentro de la web de la universidad, se busca el perfil del alumno y se localiza el email educativo guardándolo en una nueva variable.

Enlace del campus de la UPCT:

- https://campusvirtual.upct.es/uportal/modules/user/profile.xhtml?faces-redirect=true [(10)]

- A continuación, en este tercer paso, la web de la UPCT solicitará las credenciales del alumno, usuario (que será el DNI) y contraseña.

Enlace de la ventana de login del alumno de la UPCT:

- https://autentica.upct.es/cas/login?service=https%3A%2F%2Fcampusvirtual.upct.es%2Fuportal%2Fj_spring_cas_security_check [(11)]

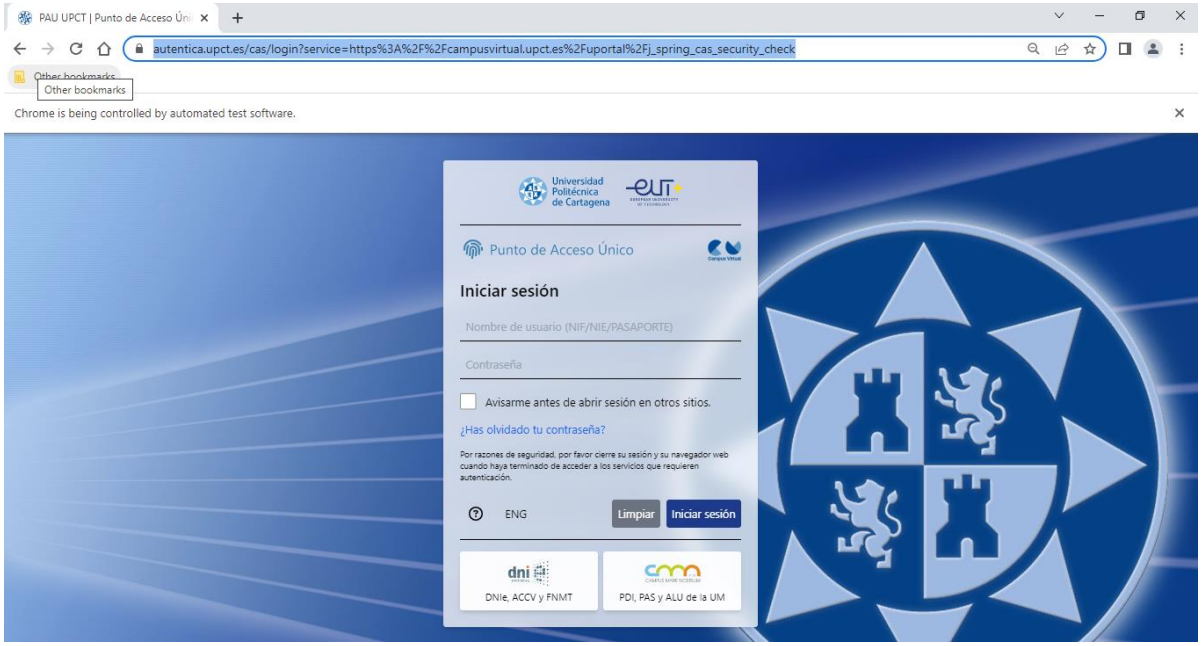

Ilustración 26: Página de login de la UPCT.

En los pasos del 4 al 10:

Como se puede apreciar en la *Ilustración 26*, la propia página de la UPCT solicita que se introduzca el NIE/DNI/PASAPORTE del alumno como 'Nombre de usuario'.

En la siguiente imagen (ilustración 27) se puede ver la parte del código implementado donde se solicitan dichas credenciales:

- En el paso 4, se solicita por ventana de comandos que se introduzca el valor del usuario. Para esto se utiliza la función de Python "input()". El programa queda a la espera hasta que alguien introduce el NIE/DNI/PASAPORTE en la ventana de comandos y se pulsa la tecla "Enter" el programa lo recoge dicho valor gracias a esta función "input()" y se guarda en una variable llamada "usuario".
- En el paso 5, muestra por pantalla (como salida en la ventana de comandos) el usuario introducido gracias a la función "print".
- En el paso 6, se manda dicho usuario al formulario de la página web gracias a la función "send_keys" (el valor lo envía al "path" que se le indica) y en la ventana de comandos nos da como salida el resultado si el usuario se ha añadido correctamente en dicho formulario web.
- En el paso 7, se solicita por ventana de comandos la contraseña (del mismo modo que se ha realizado con el "usuario"). El programa queda a la espera hasta que alguien introduzca la contraseña, una vez introducida y pulsado el "Enter" el programa la guarda el valor de dicha contraseña introducida en otra variable.

- En el paso 8, muestra por pantalla en la ventana de comandos la contraseña.
- En el paso 9, se manda la contraseña al formulario de login/credenciales de la página web de la universidad y nos comunica como salida en la ventana de comandos si la contraseña ha sido añadida correctamente en dicho formulario web.
- En el siguiente paso, el paso 10, se pulsa el botón "Iniciar Sesión".

```
print("Paso 2: Vamos la web de la universidad, al perfil de un alumno y tomamos su email educativo")
driver.get("https://campusvirtual.upct.es/uportal/modules/user/profile.xhtml?faces-redirect=true")  #Pasamos la url que deseamos al navegador web de chrome

print("Paso 3: La página de la universidad nos pedirá usuario / contraseña para poder acceder")
#usuario
print("Paso 4: Introduzca el Nombre de usuario (NIF/NIE/PASAPORTE): ")
usuario = input()
print ("Paso 5: Mostramos el valor del usuario introducido manualmente: " + usuario)
while True:
    try:
        driver.find_element(By.XPATH,"//*[@id='username']").send_keys(usuario)
        print("Paso 6: Se ha añadido el usuario correctamente")
        break
    except:
        time.sleep(0.1)
#contraseña
print("Paso 7: Introduzca su Contraseña: ")
contraseña = input()
print ("Paso 8: Mostramos el valor de la contraseña introducida manualmente: " + contraseña)
while True:
    try:
        driver.find_element(By.XPATH,"//*[@id='password']").send_keys(contraseña)
        print("Paso 9: Se ha añadido el contraseña correctamente")
        break
    except:
        time.sleep(0.1)
#pulsamos el botón de Iniciar sesión
while True:
    try:
        driver.find_element(By.XPATH,"//*[@id='fml']/div[2]/div[1]/div/input[2]").click()
        print("Paso 10: Se ha pulsado el botón Iniciar sesión")
        break
    except:
```

Ilustración 27: Parte del código donde se solicitan las credenciales.

Usamos esos valores introducidos de "Usuario" y "Contraseña", para guardarlos en variables y lo enviamos al formulario de la página web.

Para enviar el valor de las variables a los formularios de la página web de la universidad donde están las credenciales, debemos localizar el path xml del objeto web con ayuda de la herramienta de desarrollador web "Developer tool" (Ctrl+Shift+I) o directamente con el botón derecho del ratón le podemos dar a "inspeccionar" el elemento que deseamos. Una vez seleccionado debemos copiar el path.

Se añade un tutorial del funcionamiento de XPATH de Selenium utilizado en este proyecto para localizar y realizar acciones sobre los distintos elementos de una página web.

Enlace a un tutorial de XPATH de Selenium para Python:

- https://www.tutorialSelenium.com/2018/09/20/xpath-Selenium-webdriver-tutorial/ (12)

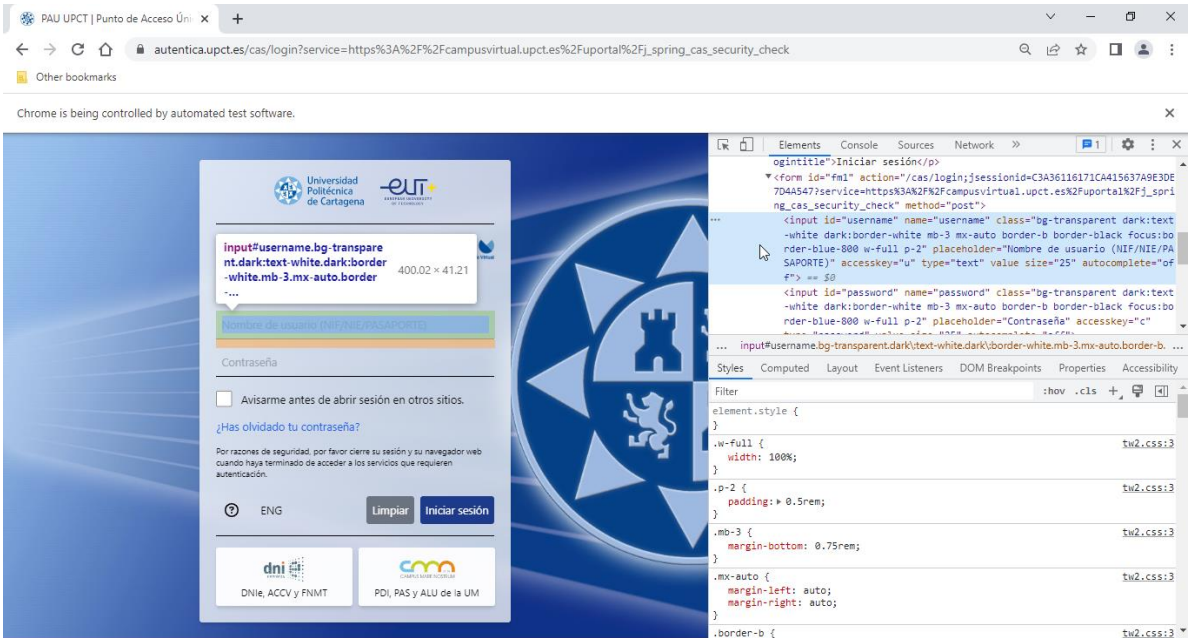

Ilustración 28: Ejemplo de path de un elemento concreto en este caso el nombre de usuario (NIE/DNI/PASAPORTE).

Para obtener, el XPath, que se necesita para que el software sepa localizar los objetos o elementos web de forma inequívoca buceando dentro del código HTML, para ello, pulsamos sobre la parte marcada en el botón derecho del ratón, elegimos la opción de "Copy" y en el siguiente desplegable elegimos normalmente entre dos opciones la de XPath o full XPath para obtener ese código del elemento exacto que necesitamos.

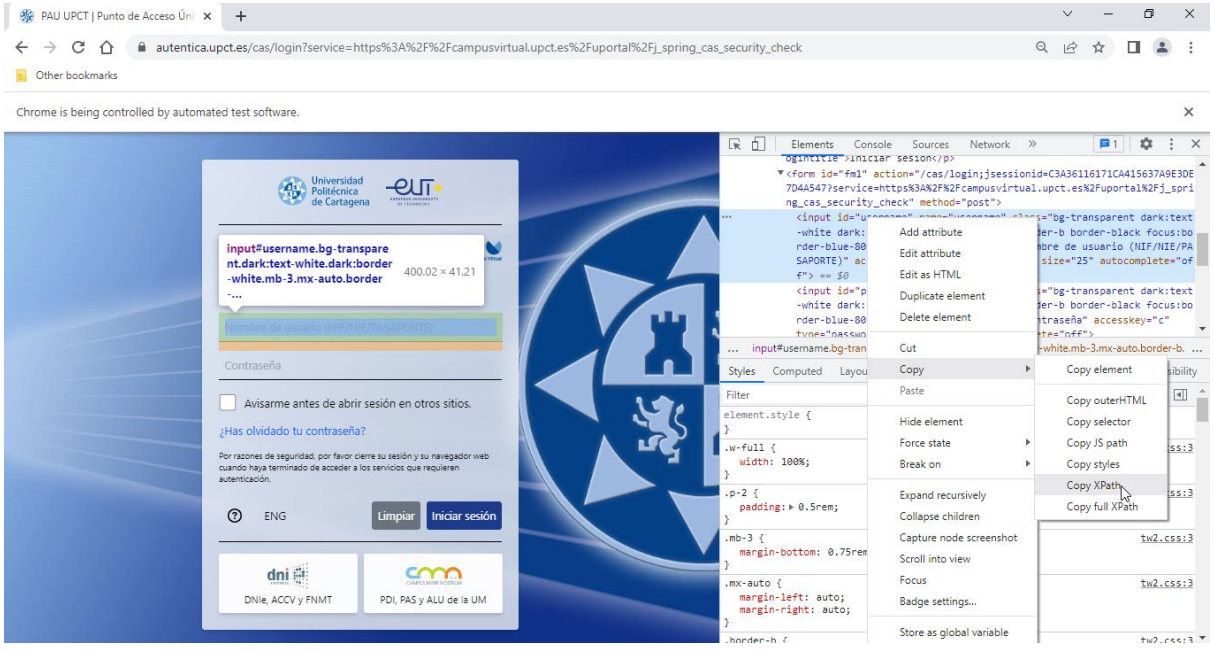

Ilustración 29: Pasos a seguir para obtener el enlace necesario para añadir al código la sentencia con la cual podremos automatizar los siguientes pasos.

Código obtenido en la casilla donde escribir el nombre del alumno:

- //*[@id="username"] → //*[@id='username']

IMPORTANTE: Cuando se realiza "Copy XPath" de los trozos de código, hay que recordar que se necesita cambiar las comillas dobles por comillas simples para que el código Python no de error cuando lo interprete.

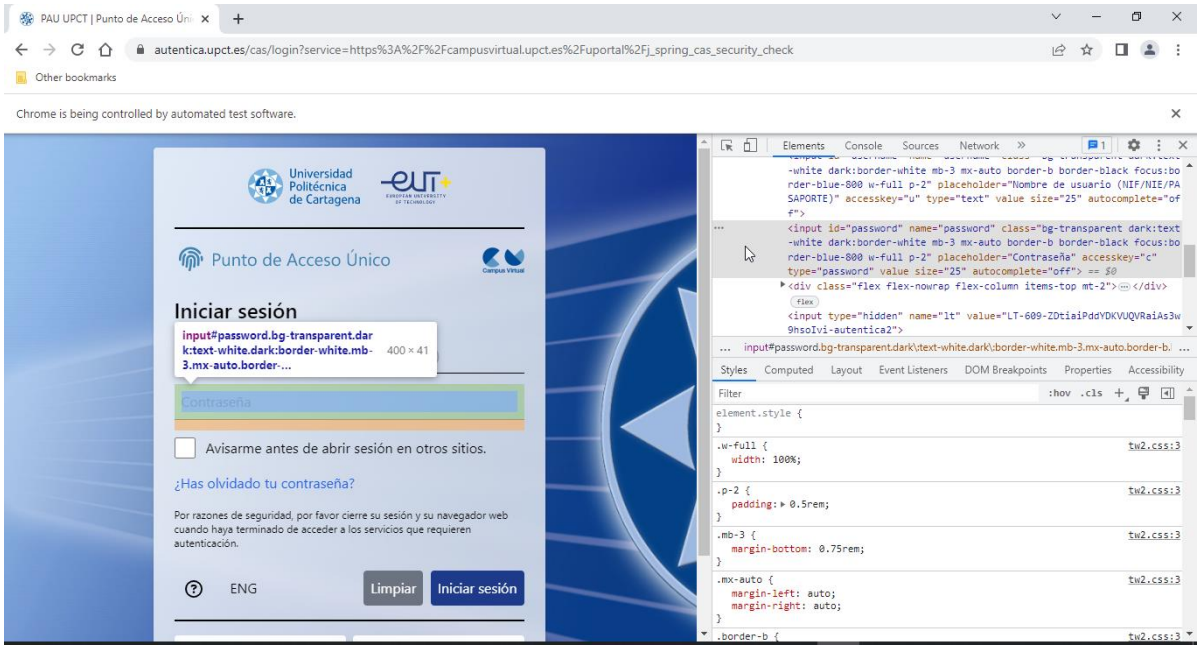

Ilustración 30: Parte del código HTML donde se obtiene la contraseña a introducir.

El path de la región donde se introducirá la contraseña es:

- //*[@id="password"]

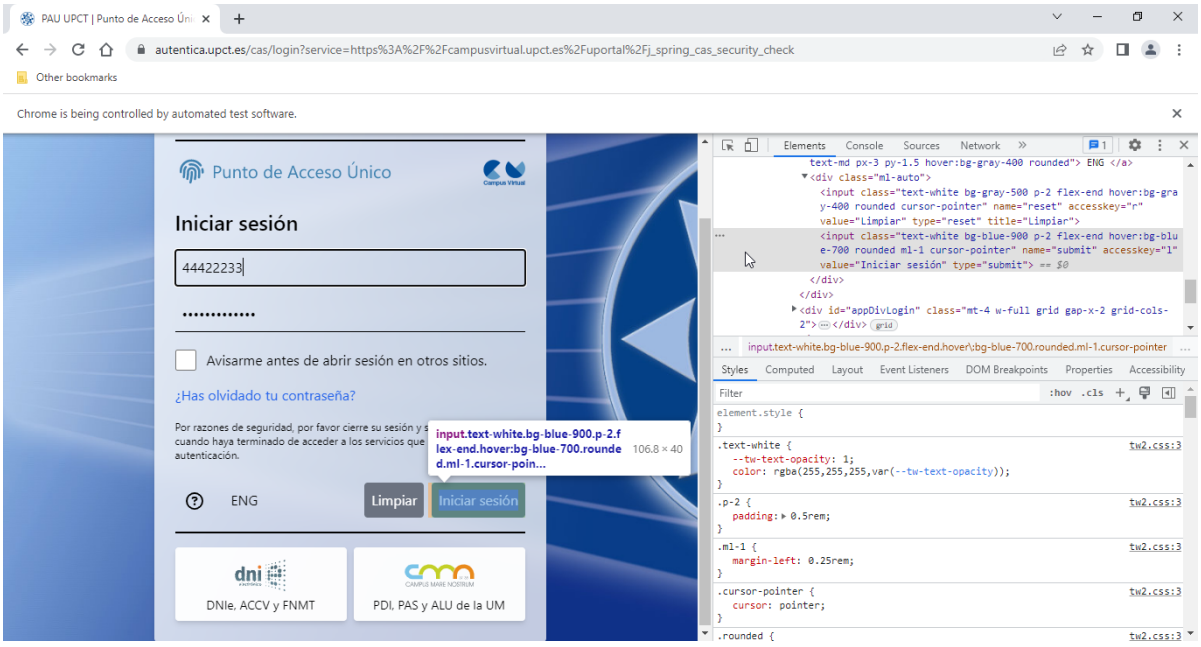

Ilustración 31: Código de muestra del botón Iniciar Sesión.

El path recuperado para iniciar sesión es el siguiente:

- //*[@id="fm1"]/div[2]/div[1]/div/input[2]

- En el paso 11, primero localizamos el XPath donde está ubicado el email del alumno, después, se recoge el valor localizado a través de la función "get_attribute" y, por último, se guarda en una variable llamada "correo_electronico" el email del alumno.
- En el paso 12, se modifica el valor del email recogido en el paso anterior para obtener solamente la primera parte del correo educativo, guardando este nuevo valor en una nueva variable llamada "primera_parte_email".

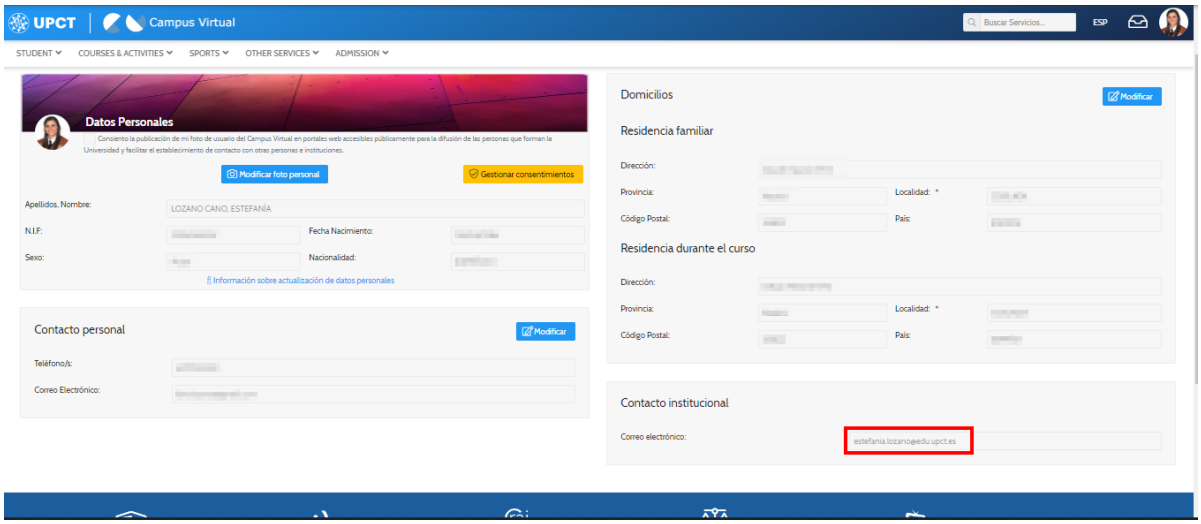

Ilustración 32: Página web de la UPCT de donde se obtiene el email de la ficha del alumno.

- En el paso 13, se procede a cerrar el navegador web sobre el que se ha estado trabajando hasta ahora y abierto en el paso 1, del cual se ha obtenido el email, que es clave para continuar, desde de la página de la UPCT.
- En el paso 14, se vuelve a abrir un nuevo navegador web Chrome, ahora para empezar a montar toda la infraestructura Azure que se desea.
- En el paso 15, vamos a la web de Azure.

  Con la cuenta de alumno, Azure proporciona 100 créditos gratuitos por parte de Azure. Estos 100 créditos son equivalentes a 100 euros o 100 dólares o 100 de la moneda en la que se vaya a facturar. La moneda de facturación depende de la región que se seleccione donde después se alojarán los servicios. Por ello, anteriormente se ha recogido entre los pasos 1 y 13 el email del alumno, porque Azure valida el email con el que se registra en Azure y verifica que es una cuenta educativa.

Enlace a la web de Azure para empezar gratis con una cuenta de estudiante:

- https://Azure.Microsoft.com/es-es/free/students/ (13)

- En el paso 16, se realiza la porción de código Python necesario para pulsar el botón de "Empiece gratis". El path recuperado para pulsar dicho botón es el siguiente:
  - //*[@id="main"]/div[1]/div/div/div[3]/ul/li[1]/a

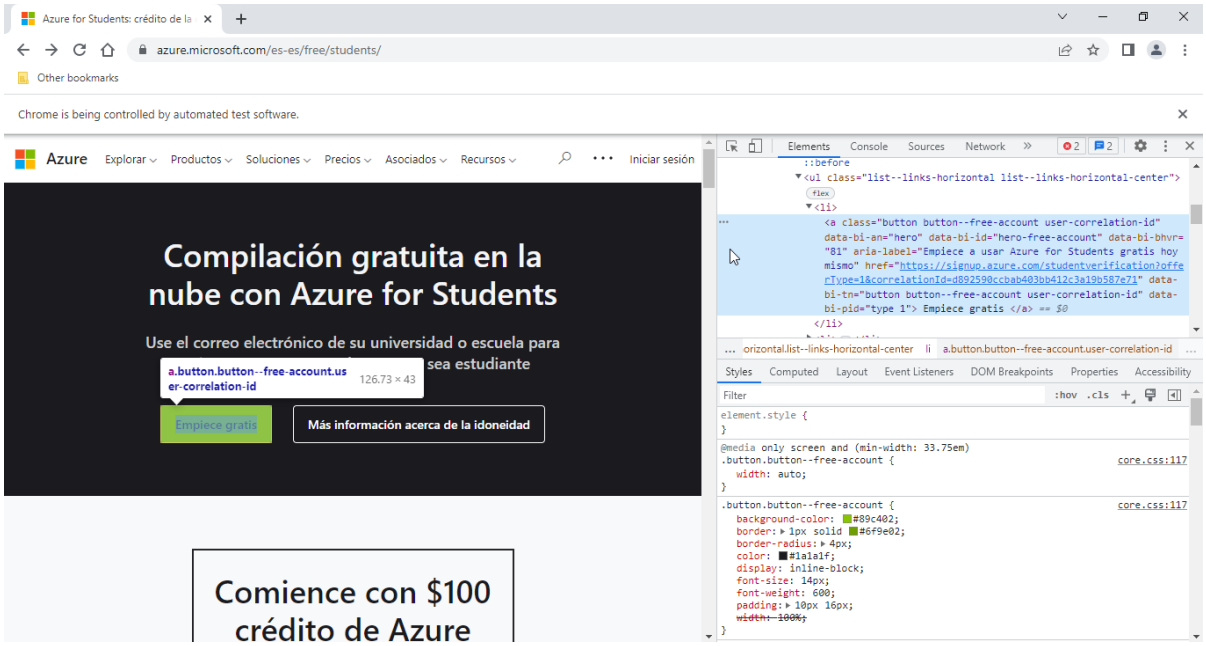

Ilustración 33: Acceso a cuenta gratuita de Azure con 100$ de crédito para estudiantes.

La página nos muestra los siguientes pasos para crear una cuenta de este tipo.

- En el paso 17, solo comunica por ventana de comandos la situación que se va a comenzar a realizar, es decir, que en este paso la web de Azure nos solicitará el usuario y contraseña para poder acceder.
- En el paso 18, mostramos el valor del usuario que en el siguiente pasó se va a introducir, en este caso el correo electrónico.
- En el paso 19, se añade el usuario en el formulario web y se muestra por ventana de comandos si se ha añadido correctamente

  Este es el path recuperado para añadir el usuario:

  - //*[@id="i0116"]

- Y en el paso 20, se procede a pulsar el botón Next.

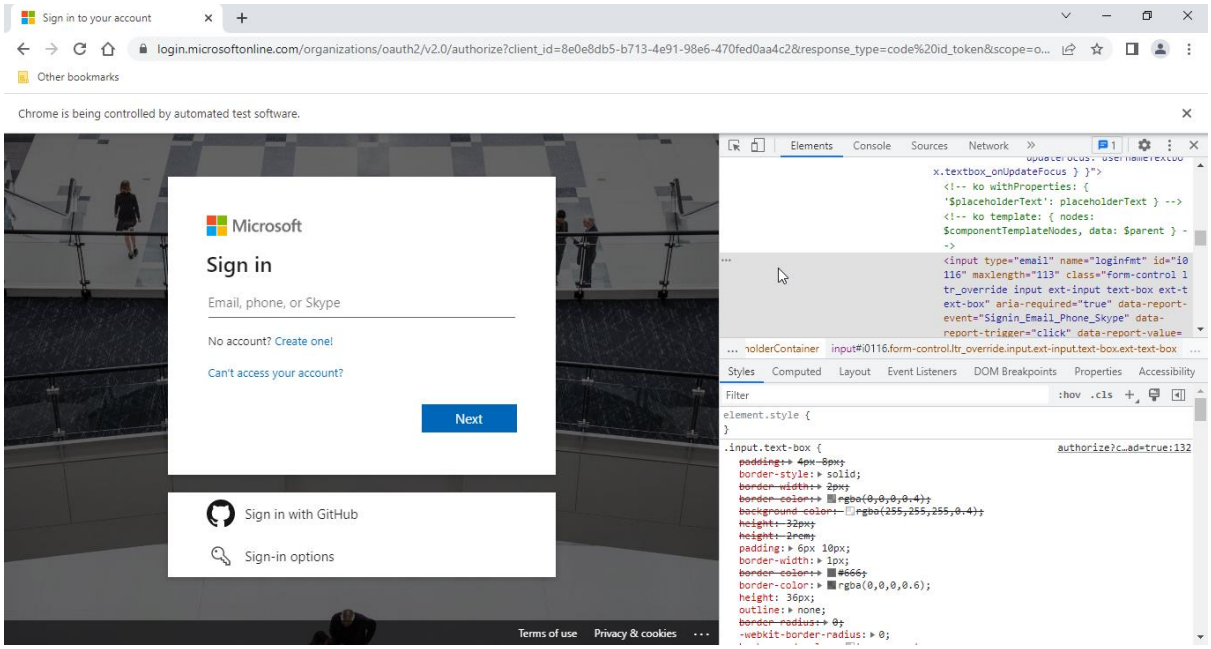

Ilustración 34: Página de login de Microsoft.

Dado que utilizamos una cuenta educativa, en este caso la cuenta universitaria, vuelve a dirigir a la web de credenciales de la UPCT, ya que requiere volver a logarse porque lo hace mediante una pasarela intermedia entre Azure y la página de la Universidad (pero como ya tenemos registradas en variables el usuario y contraseña, introducidos en los primeros pasos para recoger el email, ahora se loga en esta pasarela intermedia sin necesidad de volver a solicitar al usuario que introduzca las credenciales gracias al automatismo que se está generando con el código Python implementado en unos pocos segundos).

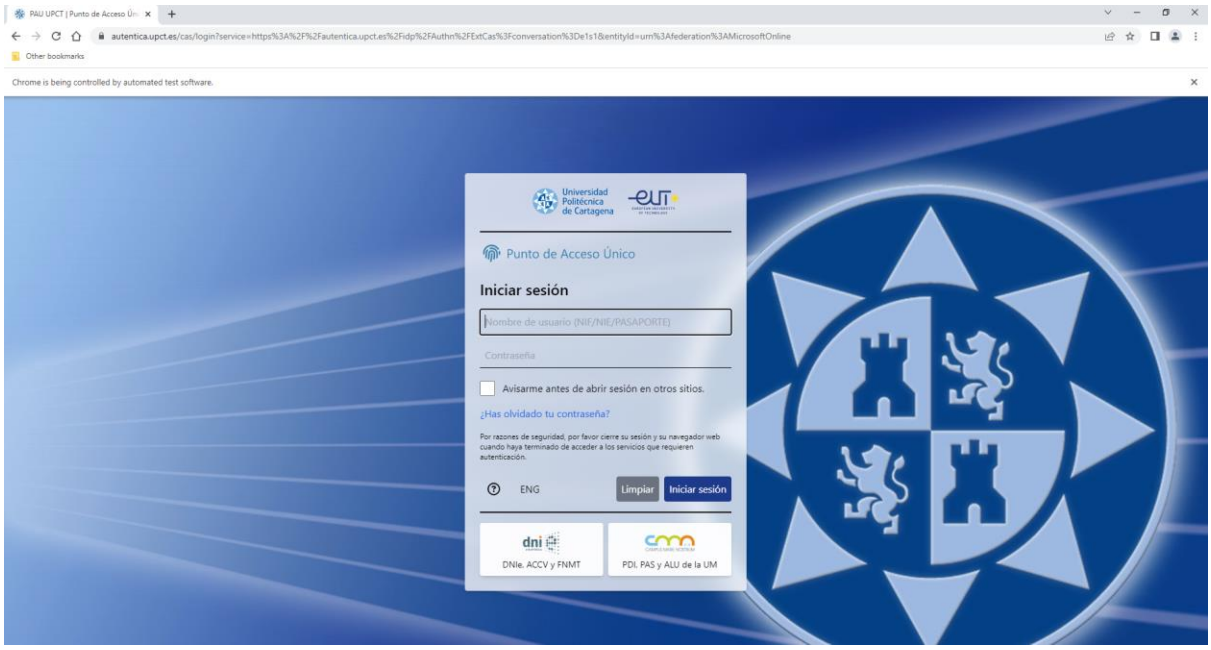

Ilustración 35: Enlace a la web de login de la UPCT, pasarela intermedia con Azure.

- En el paso 21 y 22 se añade el usuario y la contraseña correctamente y en el paso 23 se procede a pulsar el botón de Iniciar sesión.
- Tras realizar el login correctamente, nos muestra el siguiente mensaje: "**Stay signed in?**" (Ilustración 36), en este caso como lo que queremos es continuar, le decimos que sí queremos permanecer conectados, a través del código Python, y en el paso 24 pulsamos el botón "Yes".

Este es el path recuperado para pulsar el botón Yes:

- //*[@id="_weave_e_86"]

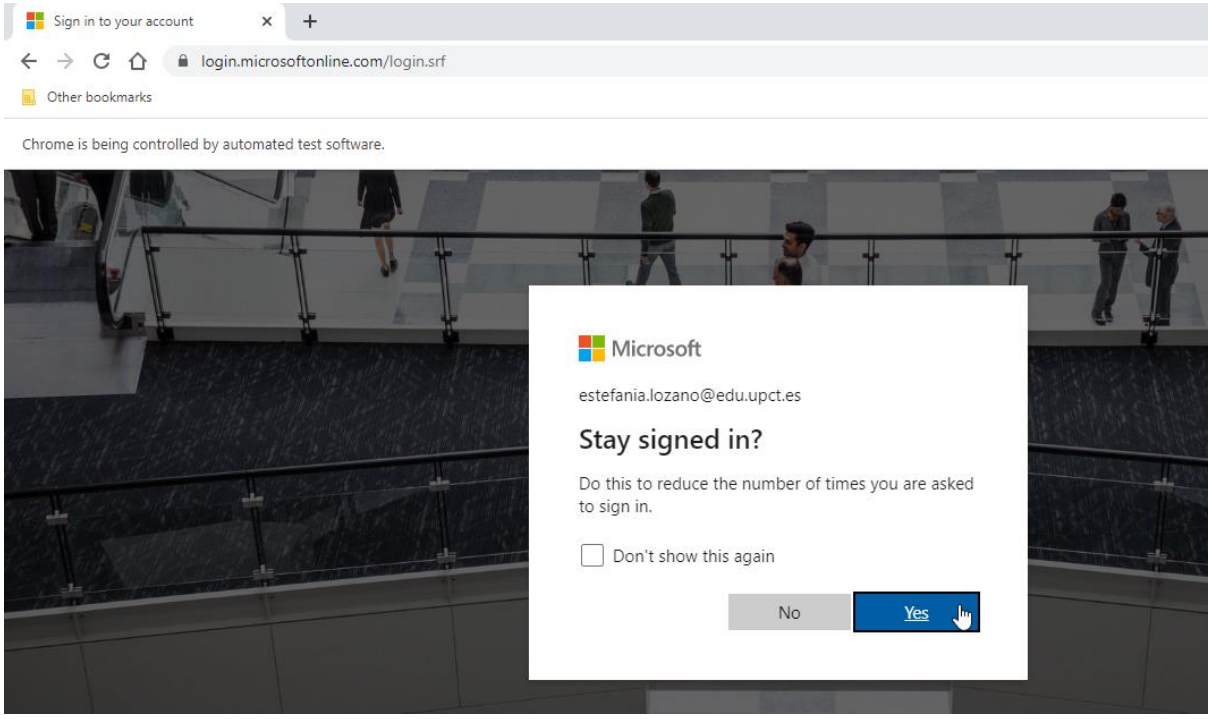

Ilustración 36: Login realizado con el correo educativo y pulsamos Yes para continuar conectados.

Nota: Como se pudo observar (Ilustración 37) en la creación de la cuenta gratuita de Azure, si se está un tiempo sin entrar (más de 30 días), o se han consumido los créditos gratuitos que se proporcionan en la web de Azure, sale un mensaje con la opción de pago por uso, en la que se debe crear una cuenta distinta a la de estudiante y solicita una tarjeta de crédito.

Ilustración 37: Condiciones de la cuenta gratuita.

A TENER EN CUENTA: Además, Microsoft tiene otra garantía de seguridad de la cuenta, a partir de unos cuantos días desde la activación, tiene una doble autenticación. Para esta doble autenticación es necesario instalarse en el móvil una app, Microsoft Authenticator (ver ***Anexo I***), configurarla y con esta te llega un SMS o llamada al móvil para confirmar que la persona que intenta acceder a tu cuenta seas únicamente tú proporcionándote un código de acceso extra.

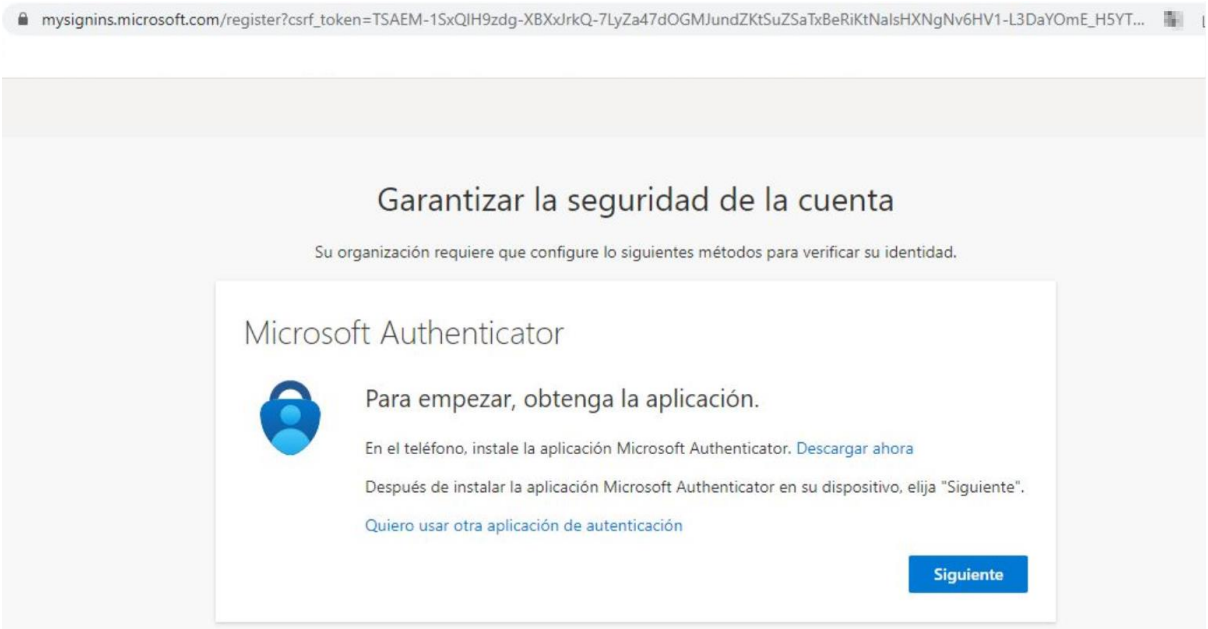

Ilustración 38: Autenticación en dos pasos con Microsoft Authenticator.

Una vez dentro de la cuenta, nos lleva a una página de bienvenida a Azure.

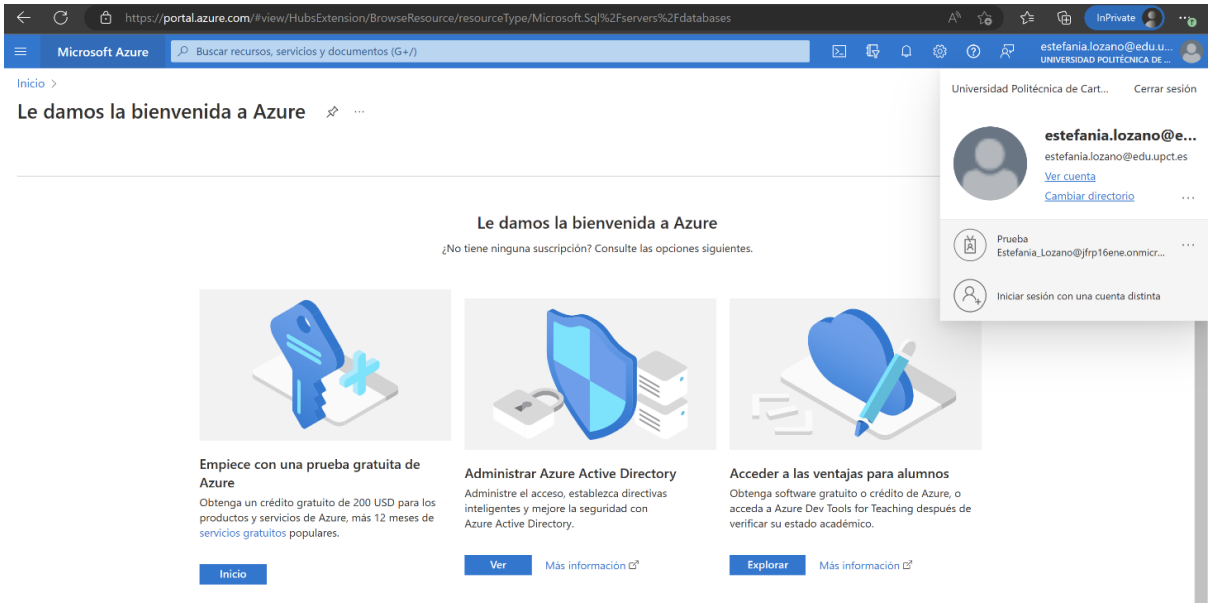

Ilustración 39: Página de Bienvenida de Azure.

En la que en la información general sobre la cuenta puedes ver varios detalles de los gastos consumidos y los servicios gratuitos a los que tienes acceso, entre otras opciones.

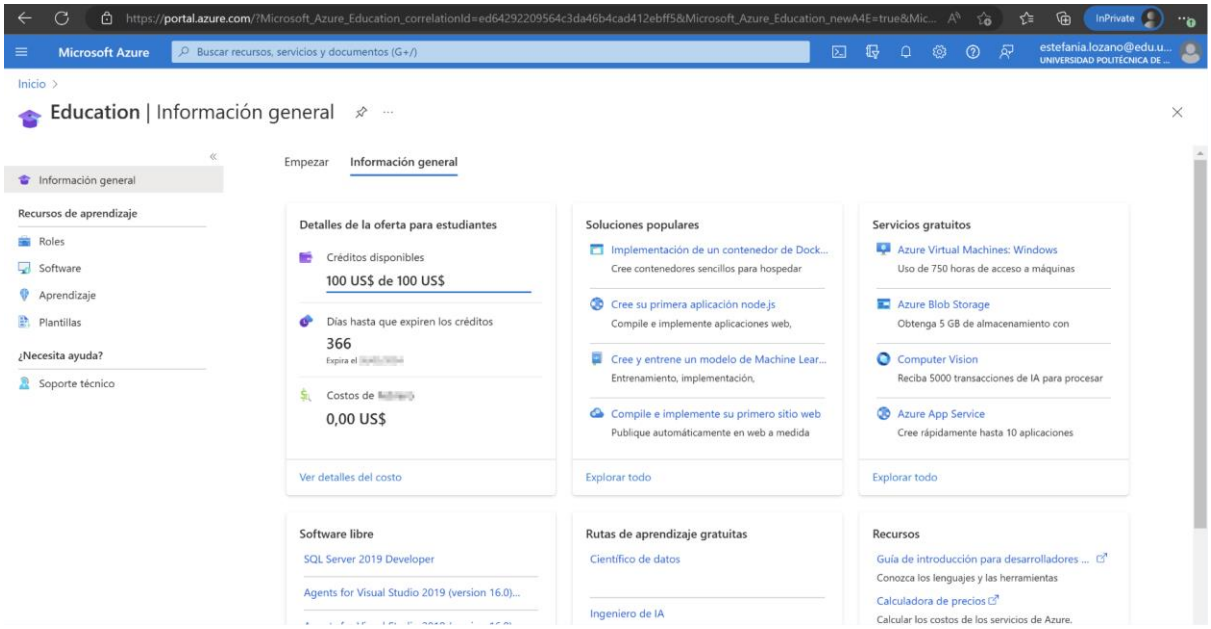

Ilustración 40: Información general sobre la cuenta de Microsoft Azure.

- Una vez dentro de Azure, hacemos que se dirija a la zona de 'Home' de Azure mediante el paso 25 de este software de Python.

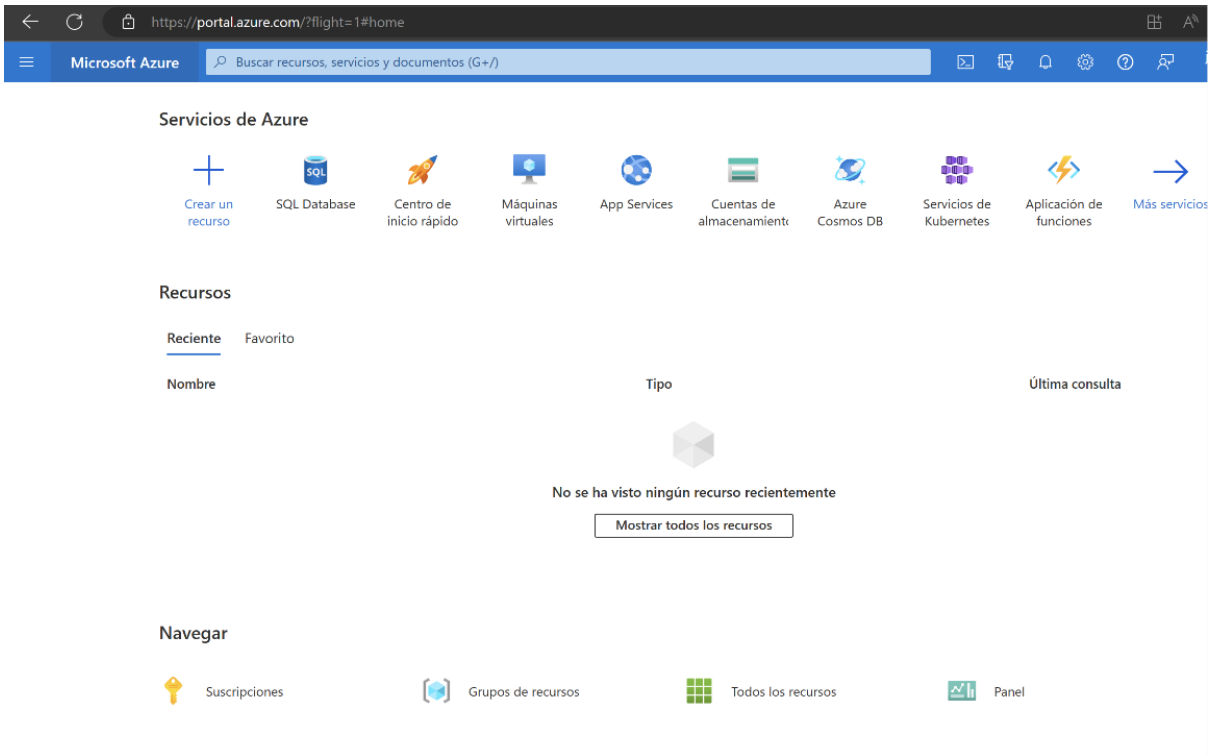

Ilustración 41: Home de Microsoft Azure.

Al entrar por primera vez con un usuario nuevo en la plataforma aparecerá un asistente para enseñarnos las opciones de Azure. Recomendable seguir para nuevos usuarios. Aunque para el caso que nos atañe, en nuestro software de Python, lo ignoramos y continuamos.

- En el paso 26, se pulsa el botón de “SQL Database” donde en primer lugar se creará un servicio de base de datos SQL Server en Azure.
- Con el paso 27, presionamos sobre el botón de “Crear”, que al ser la primera vez lo que hace es comenzar con el asistente guiado para la creación y/o configuración.

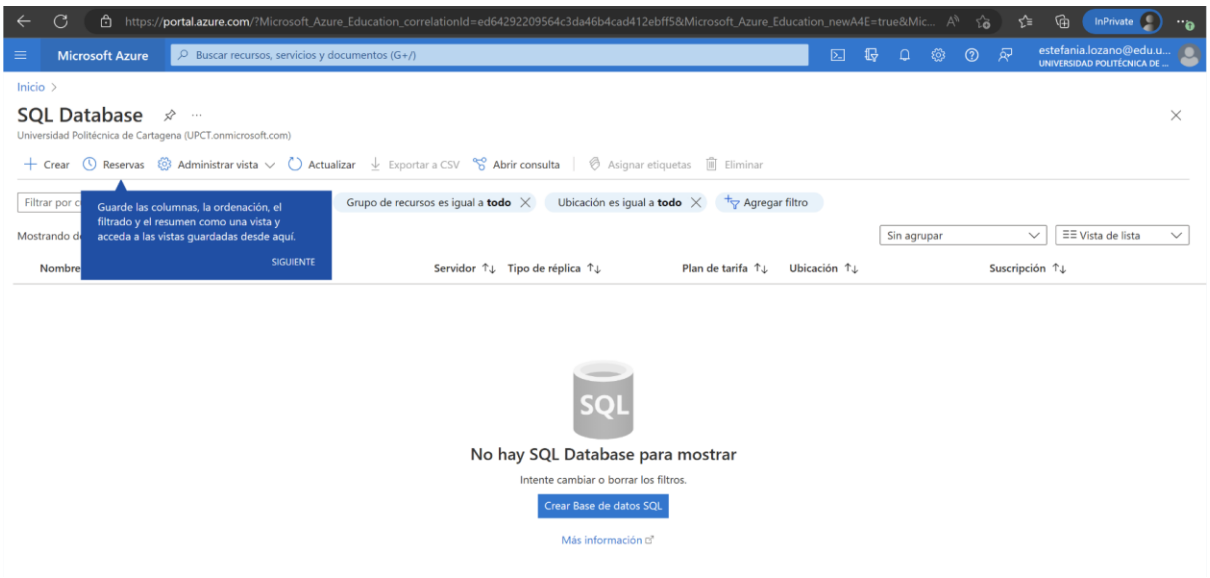

Ilustración 42: Base de datos de SQL en Azure.

- En el paso 28, pulsamos el botón de "Crear nuevo", el cual nos permitirá crear el "grupo de recursos" de una base de datos de SQL.

Nota: Si el nombre del grupo de recursos ya ha sido utilizado, al crear uno nuevo debe ser distinto, ya que no permite la creación de 2 grupos de recursos con el mismo nombre, por eso en este TFG, lo que se ha ido haciendo mientras se realizaban las pruebas fue generar un nombre de recurso fijo, se le añadía un número que iba incrementando con cada prueba y finalizaba con el nombre de usuario (primera parte del email), un ejemplo que se ha usado en este caso: *gruporecursosTFG_13estefania.lozano.*

- El paso 29, nos muestra el nombre del grupo de recursos a generar. Se ha diseñado de tal forma que para cada alumno sea un nombre único y no devuelva un error de que dicho recurso ya existe. Para ello, se ha concatenado como parte del nombre de recursos la primera parte del email de la UPCT del alumno. Tal y como se puede observar en este paso del script Python.
- En los pasos del 30 al 32, se implementa en el automatismo de Python que se vaya moviendo por página web mediante el control del teclado del ordenador gracias a la librería "Selenium" mediante la función "ActionChains" utilizando las teclas "TAB", "Enter" y según el programa va cambiando por el formulario de la página web se va añadiendo la información que necesita en cada sección y cuya información que ya se conoce y se tiene guardada en diferentes variables como por ejemplo para añadir el nombre de recursos que deseamos.

Una vez se ha creado el grupo de recursos, ahora se comienza la creación del servidor para la base de datos.

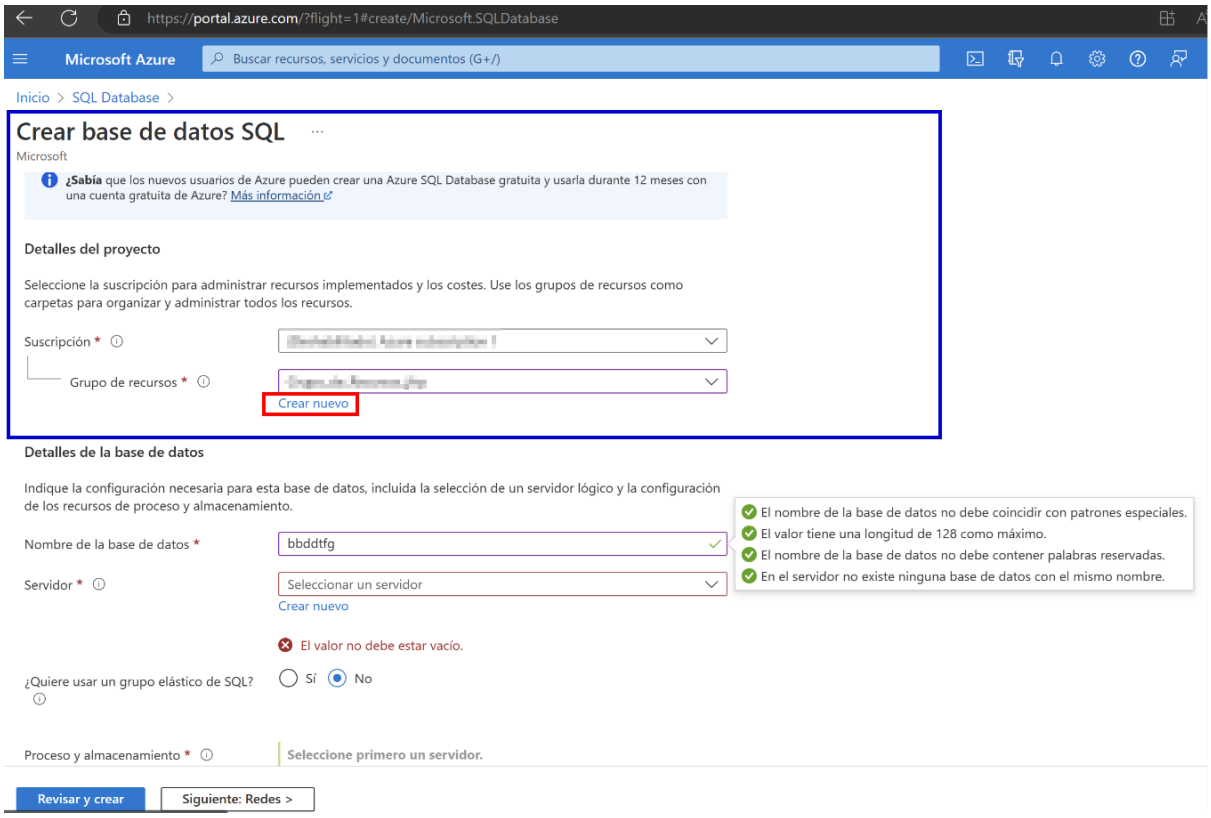

Ilustración 43: Procedemos a la creación de un grupo de recursos en la base de datos SQL.

- En el paso 33, se pulsa sobre el botón de "Crear nuevo", para crear un servidor de una base de datos de SQL y en el paso 34, se añade el nombre del servidor de base de datos, en esta ocasión el nombre elegido ha sido: "*servidorTFGteleco*".
- En el paso 35, el software, se mueve por la página hasta llegar al método de autenticación y selecciona la opción de "Uso de la autenticación de SQL", para que tengamos un usuario y contraseña concreto para acceder a la base de datos y como nombre de usuario de administrador se le llama: "usuadmin", que se guarda en la variable "Contraseña_usuadmin".
- En el paso 36, se procede a enviar el valor del usuario de administrador elegido para la base de datos ("usuadmin"), por medio de la función "send_keys()" en la parte correspondiente del formulario ubicado mediante el XPath correcto.
- Seguidamente, mediante el paso 37, gracias a la combinación de las funciones "Keys.TAB" y "send_keys()" se maneja el teclado del ordenador (con la primera función) para cambiarnos o movernos por el formulario, cambiando de casilla, y con la segunda función, se manda el valor deseado a la parte del formulario donde nos habíamos posicionado previamente con la primera función.

- En el paso 38, volvemos a tabular y nos posicionamos sobre la casilla que nos solicita la confirmación de la contraseña (como en la mayoría de los formularios, la contraseña, se ha de introducir dos veces para confirmar que es correcta, esto se realiza también por motivos de seguridad).
- y en el paso 39, enviamos con la función "send_keys()" el valor para la opción de confirmación de la contraseña del usuario administrador de la base de datos.
- En los pasos 40 y 41 se pulsa o simula la pulsación, mediante comandos Python, la tecla del tabulador del teclado del ordenador hasta que se mueve el cursor dentro de la página web hasta la tecla de "Aceptar" deseado con el que finalmente se crea el servidor de SQL Database tras haber informado/rellenado todos los campos de configuración necesarios.

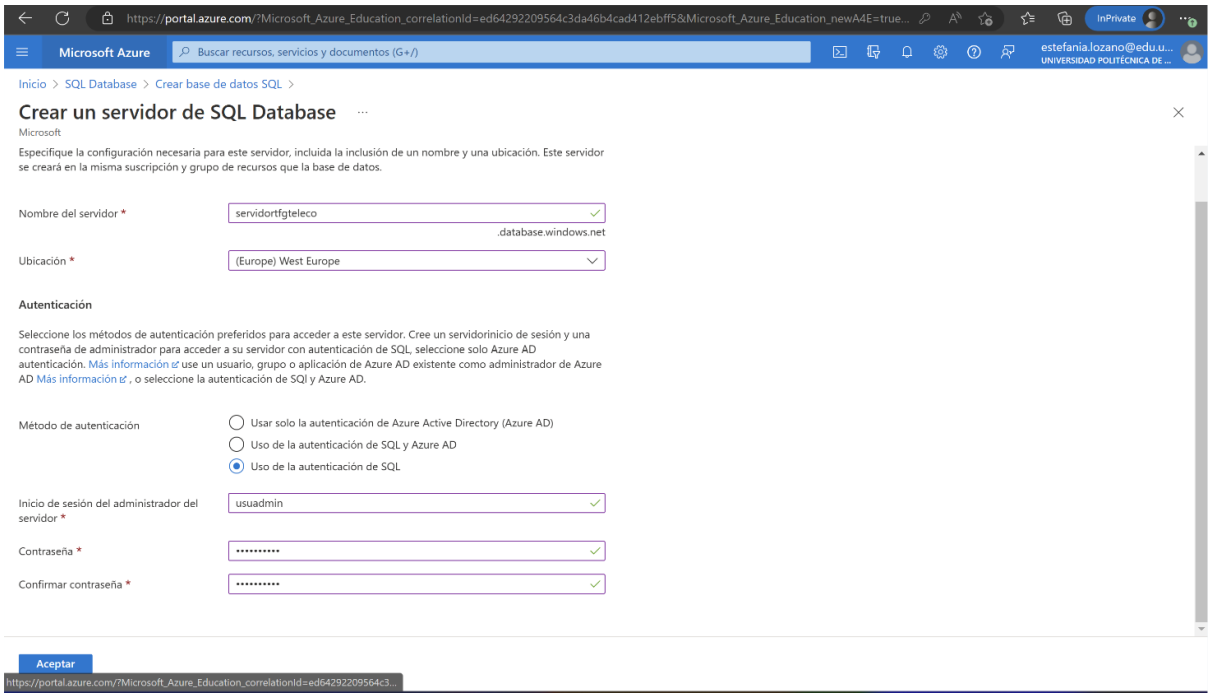

Ilustración 44: Creación de un servidor de base de datos de SQL.

- Tras la creación del servidor, cuando se pulsó el botón de "Aceptar" en el paso anterior, automáticamente se vuelve a la página principal de creación de la base de datos SQL, donde ya quedaría puesto el nombre del servidor en su lugar correspondiente en esta página principal, y ahora, es necesario introducir el nombre de la base de datos. Esto se programa en este paso 42 del script.

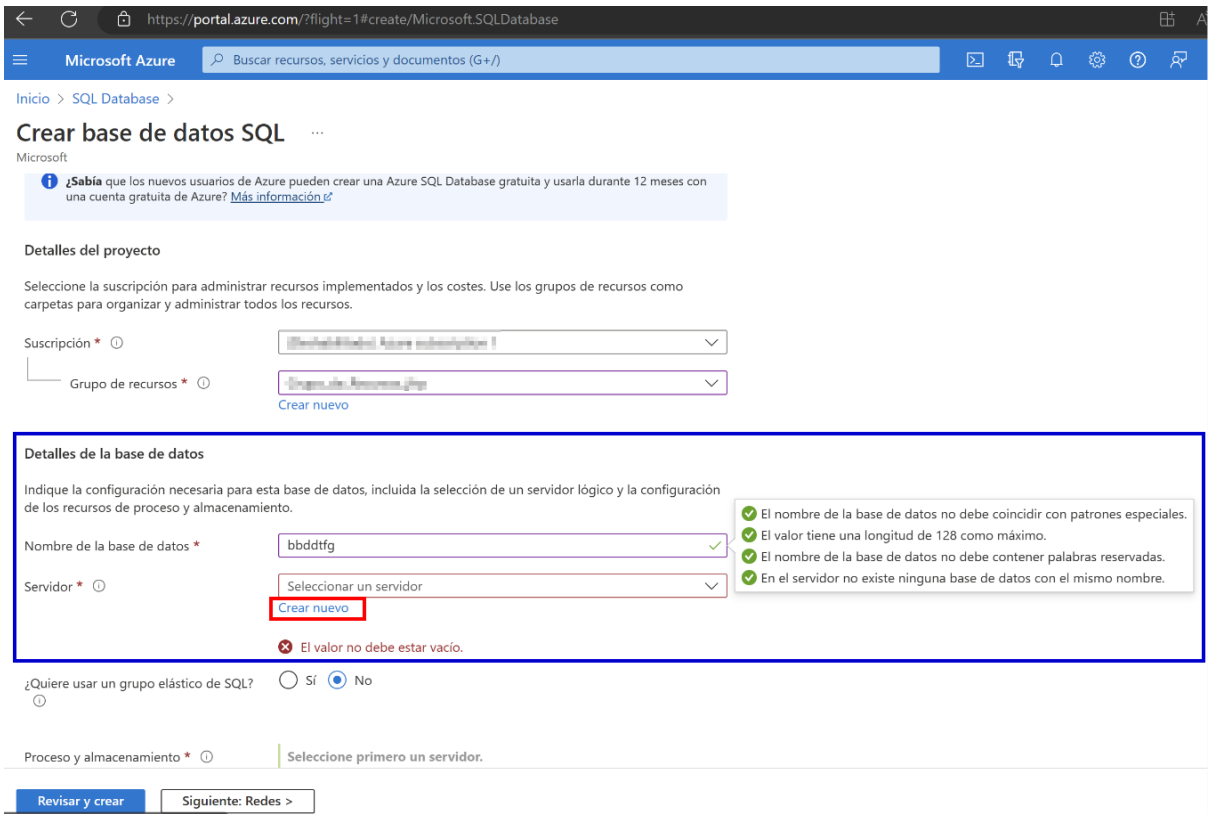

Ilustración 45: Creación de un servidor de base de datos.

- En el paso 43, es necesario que el automatismo, se nos desplace, hasta la pestaña de "Ajustes adicionales" donde permitirá seleccionar si se desea una base de datos en blanco o una base de datos de prueba.
- Mediante el paso 44, se programa el autómata para que seleccione la opción de base de datos de ejemplo de Azure. Una vez realizado esto, se está en disposición de finalizar la creación.
- En el paso 45, se programa la sección del código Python que permite pulsar o presionar el botón "Revisar + Crear", aunque después de pulsar este botón para la finalización de la creación de la base de datos en Azure es necesario confirmar volviendo a pulsar un nuevo botón con el descriptivo "Aceptar" que se acciona en la sección del código del script del paso 46.
- En el paso 47, se configura que el código espere durante un tiempo para la creación de los servicios en las que se creará el grupo de recursos, el servidor de base de datos y la base de datos de ejemplo en sí. Esto puede llevar aproximadamente unos 3 minutos, de hecho, en este paso se le está indicando un "time.sleep(220)" para que no falle al intentar continuar con el siguiente paso antes de que finalice este mismo.

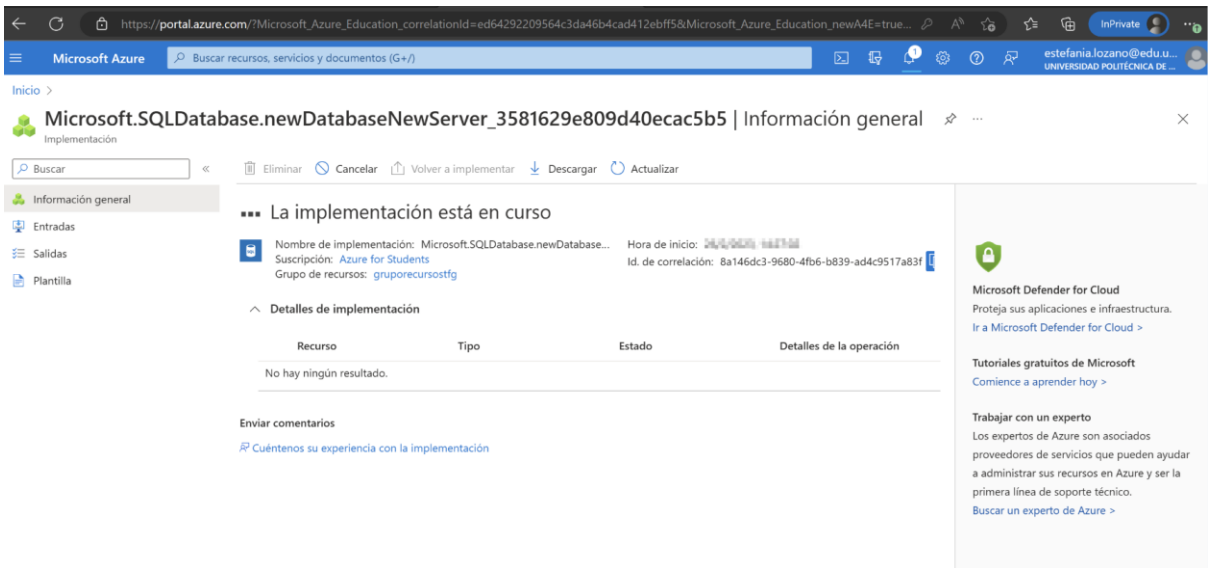

Ilustración 46: Detalles de implementación, donde nos muestra los servicios implementados.

En la siguiente Ilustración (47), se puede observar la finalización de la creación de los servicios que se han considerado en Azure.

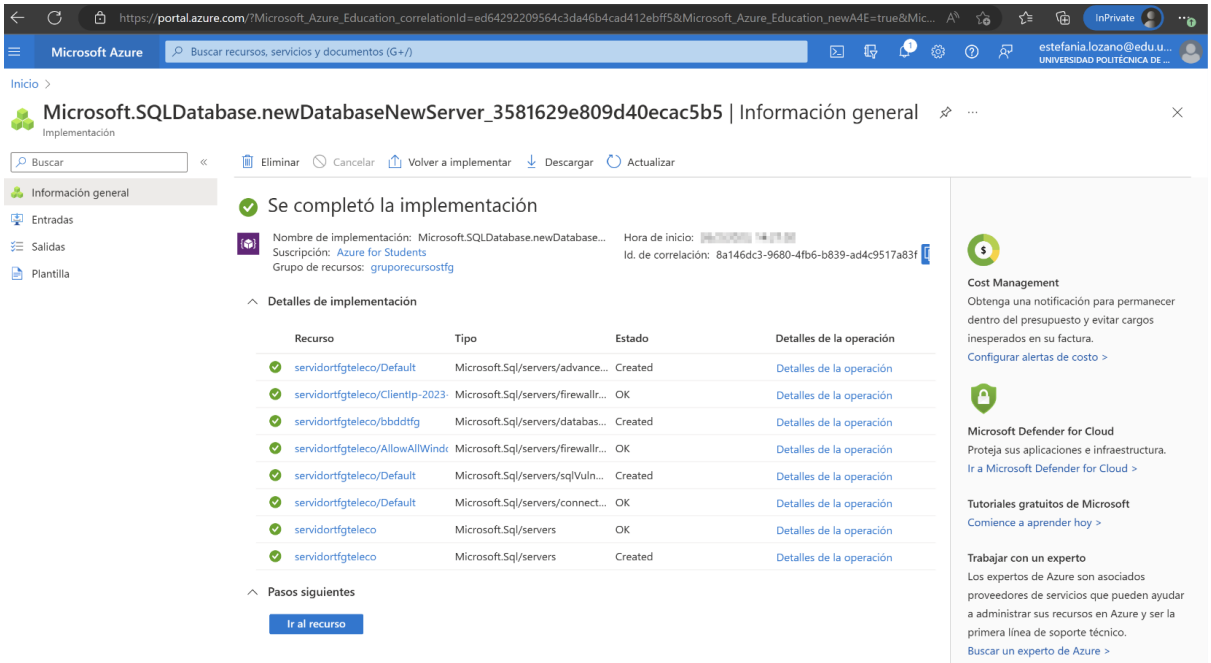

Ilustración 47: Implementación completa de la creación de BASE DE DATOS SQL de prueba.

- En el paso 48, se presiona o pulsa en el botón "Inicio" para ir a la página principal de Azure, y el paso 49 se accede a la base de datos recientemente creada.

El objetivo que se necesita conseguir a continuación es añadir la IP del ordenador que se esté usando dentro de la seguridad de la base de datos. Porque al ser un servicio de base de datos en una plataforma Cloud, además de tener la propia base de datos un usuario y contraseña es necesario habilitar las IPs de los ordenadores que accederán a dicha base de datos como un doble método de autenticación. Sin añadir la IP del ordenador, dentro de la seguridad de la base de datos, solo se podrán consultar los datos de dicha base de datos desde la propia plataforma de Azure, es decir, desde el propio portal de Azure. Sin embargo, si se ha habilitado la IP se podrá acceder a dicha base de datos desde cualquier programa externo al portal de Azure, bien sea con herramientas como Azure Data Studio (instalado en dicho ordenador con la IP añadida en la seguridad) o bien, como se va a realizar en este proyecto, desde el propio programa Python se va a lanzar una consulta SQL a la base de datos. Y como dicho programa Python está instalado en el pc indicado al inicio de este proyecto, es necesario añadir dicha IP dentro de la base de datos sino nos devolverá error de permisos cuando se intenta lanzar dicha consulta de SQL sin esos permisos.

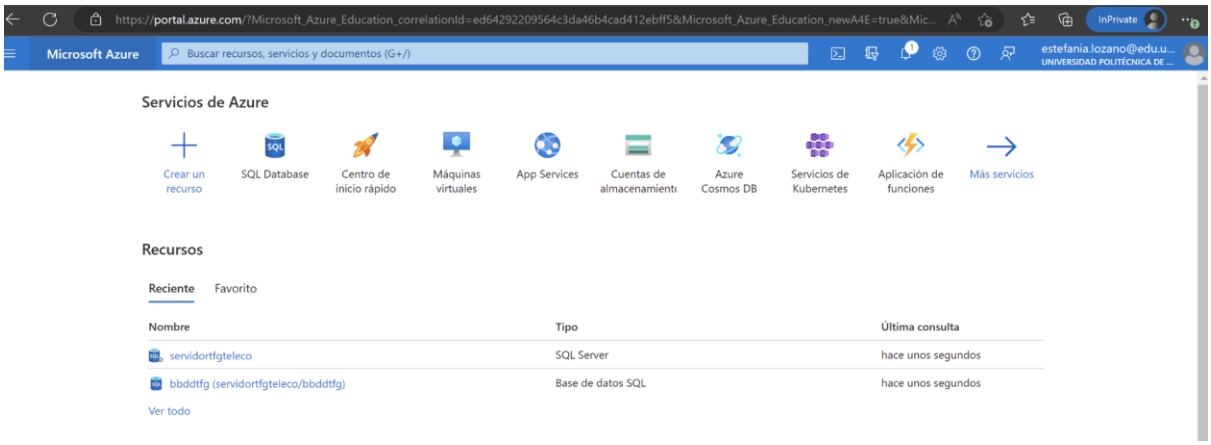

Ilustración 48: Servicios de Azure una vez creados un grupo de recursos, un servidor SQL y una base de datos SQL de ejemplo proporcionada por Azure.

- En el paso 50, se programa que se presione el botón "Empezar" y es necesario que el cursor del automatismo se mueva hasta la opción de Down. En este caso se ha utilizado las secciones de código dentro del script nombrados como pasos 51 y 52, los cuales se realizan a través de tabuladores (Keys.TAB) y presionando "Enter" (Keys.ENTER).
- En el paso 53, se vuelve a desplazar el automatismo, por la página mediante los tabuladores, y se pulsa la opción de añadir **nuestra IP en la seguridad** de Azure en concreto para el acceso a la base de datos en cuestión.
- En el paso 54, se programa que el automatismo vuelva a desplazarse mediante tabuladores para pulsar el botón de "Guardar la configuración de seguridad" recientemente cambiada en el paso anterior 53 de acceso a la base de datos.

- En el paso 55, se estable la conexión correctamente con la base de datos de Azure desde el código Python. En este paso, ya se ha dejado todo listo para ejecutar consultas desde Python a la base de datos ubicada en Azure.
- Por lo tanto, en el paso 56 se construye un cursor en Python para ir mostrando por pantalla de la ventana de comandos la información de una tabla registro a registro que se ha consultado mediante SQL desde Python
- Y por último en el paso 57 se muestra dicho resultado de la consulta SQL lanzada desde Python.

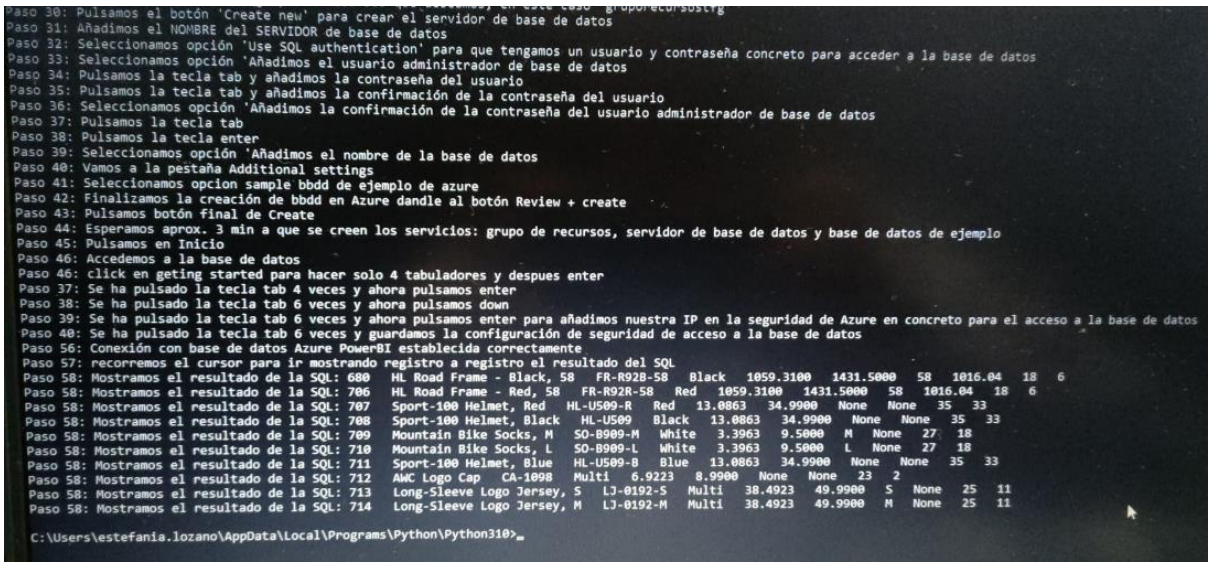

Ilustración 49: Muestra por pantalla el resultado de la consulta SQL lanzada desde Python.

Conclusiones

- Uso de código Python para automatización de tareas.
- Manejo profundo de librerías Python:
    - Selenium
    - Pyodbc
    - Datetime
- Búsquedas y localización de elementos web en HTML.
- Creación y configuración automática de plataforma Cloud, en este caso en Azure, para el manejo de base de datos, para alumnos de la UPCT con créditos gratuitos.

Ventajas

Tiempo estimado de ahorro

Para una persona que comienza con la utilización de una plataforma Cloud de Azure pasan varios días hasta que lee documentación y aprende a configurar y crear un entorno mínimo para su uso. Con este automatismo que se proporciona en menos de 5 min queda todo plataformado y listo para usar.

Si una empresa media utiliza por ejemplo unos 50 servicios, podríamos estimar cuál sería su tiempo estimado de ahorro si utilizara un automatismo similar al de este proyecto:

> Si para un servicio de Azure tenemos un ahorro medio de 6 horas, para 50 servicios tendríamos un ahorro de unas 300 horas lo que se traduce en aproximadamente 2 meses de trabajo de un recurso.

Este proyecto no sirve sólo como base para montar plataformas en Cloud, sino **lo importante es la mecanización de procesos rutinarios en las empresas que suele ocupar la mayor parte de los recursos que se disponen y no les permite enfocarse en otros aspectos** que los necesitan para seguir creciendo. Siguen enfocados en hacer tareas que son necesarias, pero siempre del mismo modo, de esta forma se pueden automatizar liberando a los recursos de su tiempo para mejorar otros aspectos y seguir evolucionando.

Riesgos

Es posible que con el tiempo el código en Python desarrollado necesite ajustes dado que las páginas web suelen sufrir mejoras o pequeños cambios. Esto puede provocar que el código desarrollado no funcione como en un principio. Pero con pequeñas adaptaciones en el código sería fácil ajustarlo. No es necesario que lo realice un perfil muy experimentando, con tener unas nociones básicas de programación sería suficiente.

Líneas futuras

- Se podría crear un listado de usuarios en el que se recojan sus emails y se establezca un nuevo bucle dentro del script que genere una plataforma para cada uno de los diferentes emails o alumnos que se necesiten. Por lo tanto, con pocas líneas de código adicionales se puede crear una plataforma individual en Cloud para todo un conjunto de alumnos que pueden ser una clase completa o varias clases.

- Se podría hacer un script de borrado de la plataforma de Cloud. Si en vez de crear una plataforma gratuita, se creara de pago, y dicha plataforma se pagará no solo por su uso sino simplemente por tenerla creada, se podría desarrollar el eliminar de forma automática toda la plataforma. De modo que, si se tiene automatizada la creación, se puede crear y borrar según se utilice o no. Minimizando costes tras cada prueba realizada.

    - A tener en cuenta: si se modifican tablas o datos dentro de la base de datos creada, cuando se ejecute el script de borrado se perderán las modificaciones realizadas. Para no perder estos cambios sería necesario crear backups de la base de datos y en vez de volver a generar una base de datos nueva o de ejemplo, sería conveniente restaurar la base de datos que tuvieras guardada como backups.

- En el Anexo II se proporciona cómo conectar con SharePoint desde Python lo que permite mucha flexibilidad a la hora de automatizar muchas tareas habituales en las empresas. Dado que trabajar con ficheros, son trabajos muy habituales y gracias a las librerías de Selenium + SharePoint + pandas se pueden automatizar y liberar mucho tiempo de las personas que aún están acostumbradas a seguir haciendo este tipo de trabajos de forma manual.

Bibliografía

- https://www.Python.org/
- https://aprendePython.es/pypi/scraping/Selenium/
- https://www.Selenium.dev/documentation/webdriver/
- https://learn.Microsoft.com/en-us/Azure/?product=popular
- https://www.tutorialSelenium.com/2018/09/20/xpath-Selenium-webdriver-tutorial/
- https://www.Microsoft.com/es-es/SQL-Server/SQL-Server-downloads
- https://Azure.Microsoft.com/es-es/free/students/
- https://learn.Microsoft.com/es-es/SQL/Azure-data-studio/download-Azure-data-studio?view=SQL-Server-ver16&tabs=redhat-install%2Credhat-uninstall

Anexo I – Microsoft Authenticator

Microsoft Authenticator es una aplicación gratuita, puedes iniciar sesión en tu cuenta Microsoft personal, en la profesional o en la educativa sin usar una contraseña. Por motivos de seguridad, tendrás que usar huella digital, reconocimiento facial o PIN.

¿Por qué usar la aplicación Microsoft Authenticator?

1. La aplicación de autenticación es una forma práctica y segura de demostrar quién eres.
2. Puedes usar la aplicación Authenticator para iniciar sesión si olvidas la contraseña.
3. La aplicación se puede usar para realizar una copia de seguridad y restaurar todas las demás credenciales de la cuenta.
4. También puedes usar Microsoft Authenticator para iniciar sesión en tus cuentas que no son de Microsoft.

¿Cómo configurar la aplicación Microsoft Authenticator?

1. Descargar e instalar la aplicación Microsoft Authenticator en tu dispositivo móvil.
2. Inicia sesión en el panel de seguridad de tu cuenta.
3. Selecciona Agregar una nueva forma de iniciar sesión o verificar y elige Usar una aplicación.
4. Si ya has instalado la aplicación, selecciona Siguiente para mostrar un código QR en la pantalla.
5. En la aplicación de autenticación, selecciona [tres puntos] y, luego + Agregar cuenta.
6. Elige el tipo de cuenta y selecciona Escanear un código QR.
7. Digitaliza el código que se muestra en la pantalla en el paso 4.
8. Selecciona Finalizar en el PC para completar la configuración.

Anexo II – Conexión a SharePoint y manejo de ficheros con Python.

```
## 0- IMPORTAMOS LAS LIBRERIAS QUE NECESITAMOS DE PYTHON ######################################
from datetime import datetime, date, time, timedelta
import time
from selenium import webdriver
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
# import Action chains
from selenium.webdriver import ActionChains  # Para hacer click con el botón derecho del ratón.
import pyodbc # Para conectar a base de datos, para utilizar esta libería es necesario instalar: pip install pyodbc

import pyodbc
import pandas as pd
import os # Para manejo de ficheros en local

# Para trabajar con sharepoint. Para ello es necesario instalar previamente las librerías: pip install Office365-REST-Python-Client
# from shareplum import Site, Office365 #Instalar librería: pip install SharePlum
from office365.runtime.auth.authentication_context import AuthenticationContext
from office365.sharepoint.client_context import ClientContext, UserCredential #, ClientCredential
from office365.sharepoint.files.file import File
# from office365.sharepoint.files.file_system_object_type import FileSystemObjectType
from office365.sharepoint.listitems.listitem import ListItem

# Para subir ficheros:
from shareplum import Office365
from shareplum import Site
from shareplum.site import Version
from requests_ntlm import HttpNtlmAuth
```

```
nombre_servidor = 'servidortfgteleco'

database = 'bbdd_tfg'

usuario_administrador = 'usuadmin'

Contraseña_usuadmin = 'passwordadmin_1'

driverSQL = '{ODBC Driver 17 for SQL Server}'


server = nombre_servidor +'.database.windows.net'


connection =
pyodbc.connect('DRIVER='+driverSQL+';SERVER=tcp:'+server+';PORT=1433;DATABASE='+database+';UID='+usuario_ad
ministrador+';PWD='+ Contraseña_usuadmin)


query = "SELECT TOP (11) [ProductID]
,[Name],[ProductNumber],[Color],[StandardCost],[ListPrice],[Size],[Weight],[ProductCategoryID],[ProductModelID]
FROM [SalesLT].[Product]"


df = pd.read_sql(query, connection)


# Exportar los resultados de la consulta SQL a un archivo Excel

df.to_excel('C:/Users/estefania.lozano/AppData/Local/Programs/Python/Python310/nombre_archivo.xlsx',
index=False)

if
os.path.exists("C:/Users/estefania.lozano/AppData/Local/Programs/Python/Python310/Datos_Extraidos_desde_SQL_
a_Excel.xlsx"):


os.remove("C:/Users/estefania.lozano/AppData/Local/Programs/Python/Python310/Datos_Extraidos_desde_SQL_a_E
xcel.xlsx")

  print("El archivo ha sido borrado exitosamente.")

else:

  print("El archivo no existe.")



Nombre_Archivo_Descargado =
"C:/Users/estefania.lozano/AppData/Local/Programs/Python/Python310/nombre_archivo.xlsx"

Renombrado_Archivo_Descargado =
"C:/Users/estefania.lozano/AppData/Local/Programs/Python/Python310/Datos_Extraidos_desde_SQL_a_Excel.xlsx"

os.rename(Nombre_Archivo_Descargado, Renombrado_Archivo_Descargado)

print("Paso 1: El archivo ",Nombre_Archivo_Descargado,"ha sido renombardo por
",Renombrado_Archivo_Descargado)
```

```
#CONECTAMOS CON SHAREPOINT:  --> EL USUARIO DE LA UNIVERSIDAD EN SHAREPOINT NO TIENE LOS PERMISOS SUFICIENTES PARA CONECTAR DESDE PYTHON....alguien administrador debería otorgar más permiso para poder automatizar la subida de un fichero desde un pc local a sharepoint.

# CONECTAMOS CON SHAREPOINT: --> EL USUARIO DE LA UNIVERSIDAD EN SHAREPOINT NO TIENE LOS PERMISOS SUFICIENTES PARA CONECTAR DESDE PYTHON... alguien administrador debería otorgar más permiso para poder automatizar la subida de un fichero desde un pc local a sharepoint.

sharepoint_user = 'estefania.lozano@edu.upct.es'

sharepoint_password = 'Telecomu_XX!'

sharepoint_base_url = 'https://upct-my.sharepoint.com/personal/estefania_lozano_edu_upct_es'

folder_in_sharepoint  = 'Documents'

folder_in_sharepoint2 = 'XLSX_subidos_con_Python'

path =
"C:\\Users\\estefania.lozano\\AppData\\Local\\Programs\\Python\\Python310\\Datos_Extraidos_desde_SQL_a_Excel
.xlsx" #local path


auth = AuthenticationContext(sharepoint_base_url)

auth.acquire_token_for_user(sharepoint_user,  sharepoint_password)

ctx = ClientContext(sharepoint_base_url, auth)

web = ctx.web

ctx.load(web)

ctx.execute_query()

print('Paso 2: Se ha conectado al sitio de SharePoint: '+str(web.properties['Title']))


authcookie = Office365('https://ufinet.sharepoint.com', username=sharepoint_user,
password=sharepoint_password).GetCookies()

site = Site('https://upct-my.sharepoint.com/personal/estefania_lozano_edu_upct_es/', version=Version.v2016,
authcookie=authcookie)

url_target='Documents/XLSX_subidos_con_Python/'+row[0]

url_target_site = "/personal/estefania_lozano_edu_upct_es/Documents/XLSX_subidos_con_Python/"+row[0]
```

Anexo III – Inicialización en Python

Se recomienda realizar cursos en la plataforma Udemy para programadores principiantes de Python.

Por ejemplo:

- **Introducción a la Programación con Python**
  - https://www.udemy.com/course/introduccion-a-la-programacion-con-Python-juan-de-la-torre/

- **Fundamentos de Python**
  - https://www.udemy.com/course/fundamentos-de-Python-intro/

- **Programación para principiantes en Python**
  - https://www.udemy.com/course/programacion-para-principiantes-en-Python/