Estructura del proyecto

|  |
| --- |
| wxo\_auth\_app/  │── app.py  │── requirements.txt  │── .env.example  │── README.md  │── Dockerfile |

app.py

|  |
| --- |
| import os  import json  from flask import Flask, redirect, request, session, url\_for, jsonify, render\_template  from msal import ConfidentialClientApplication  from dotenv import load\_dotenv  load\_dotenv()  app = Flask(\_\_name\_\_)  app.secret\_key = os.getenv("FLASK\_SECRET", "supersecret")  # Configuración Microsoft Identity  CLIENT\_ID = os.getenv("CLIENT\_ID")  CLIENT\_SECRET = os.getenv("CLIENT\_SECRET")  TENANT\_ID = os.getenv("TENANT\_ID")  AUTHORITY = f"https://login.microsoftonline.com/{TENANT\_ID}"  REDIRECT\_PATH = os.getenv("REDIRECT\_PATH", "/redirect")  SCOPE = ["User.Read"] # Scopes básicos  @app.route("/")  def index():  if not session.get("user"):  return redirect(url\_for("login"))  return render\_template("home.html", user=session["user"])  @app.route("/login")  def login():  cca = ConfidentialClientApplication(  CLIENT\_ID, authority=AUTHORITY, client\_credential=CLIENT\_SECRET  )  auth\_url = cca.get\_authorization\_request\_url(  SCOPE, redirect\_uri=url\_for("authorized", \_external=True)  )  return redirect(auth\_url)  @app.route(REDIRECT\_PATH)  def authorized():  code = request.args.get("code")  if not code:  return "Error: no se recibió el código de Microsoft", 400  cca = ConfidentialClientApplication(  CLIENT\_ID, authority=AUTHORITY, client\_credential=CLIENT\_SECRET  )  result = cca.acquire\_token\_by\_authorization\_code(  code, scopes=SCOPE, redirect\_uri=url\_for("authorized", \_external=True)  )  if "id\_token\_claims" in result:  session["user"] = result["id\_token\_claims"]  return redirect(url\_for("index"))  return "Error al autenticar", 400  @app.route("/logout")  def logout():  session.clear()  return redirect(  f"{AUTHORITY}/oauth2/v2.0/logout?post\_logout\_redirect\_uri={url\_for('index', \_external=True)}"  )  @app.route("/me")  def me():  if "user" in session:  return jsonify({"authenticated": True, "user": session["user"]})  return jsonify({"authenticated": False})  if \_\_name\_\_ == "\_\_main\_\_":  app.run(host="0.0.0.0", port=8000, debug=True) |

requirements.txt

|  |
| --- |
| flask  msal  python-dotenv |

.env.example

|  |
| --- |
| CLIENT\_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  CLIENT\_SECRET=your\_secret\_here  TENANT\_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  FLASK\_SECRET=supersecretkey  REDIRECT\_PATH=/redirect |

Perfecto 🚀 vamos a registrar tu aplicación en **Azure Entra ID (Azure Active Directory)** para que el flujo de login con Microsoft funcione. Aquí están los pasos detallados:

:

**🔹 1. Acceder al portal**

1. Ve a 👉 https://portal.azure.com.
2. En el buscador escribe **Entra ID** o **Azure Active Directory** y entra.

**🔹 2. Registrar una aplicación**

1. En el panel lateral, selecciona **App registrations** → **New registration**.
2. Dale un nombre (ej: WatsonxAuthApp).
3. **Supported account types**:
   * Si solo se usará en tu organización → elige *Accounts in this organizational directory only*.
   * Si quieres permitir login de otros tenants → elige *Multitenant*.
4. **Redirect URI** → muy importante:
   * Tipo: *Web*
   * URI: http://localhost:8000/redirect (luego podrás añadir más para producción).
5. Clic en **Register**.

**🔹 3. Obtener credenciales**

Una vez registrada:

* **Application (client) ID** → lo copias en CLIENT\_ID de tu .env.
* **Directory (tenant) ID** → lo copias en TENANT\_ID.

**🔹 4. Crear un secreto de cliente**

1. En el menú de la app → **Certificates & secrets** → **New client secret**.
2. Dale un nombre y fecha de expiración.
3. Copia el valor (¡se muestra una sola vez!) y guárdalo en .env como CLIENT\_SECRET.

**🔹 5. Configurar permisos de API**

1. Menú → **API permissions** → **Add a permission**.
2. Elige **Microsoft Graph**.
3. Agrega el permiso **User.Read** (delegado).
4. Haz clic en **Grant admin consent** si eres admin.

**🔹 6. Configurar Redirect URI (si agregas más entornos)**

* Si luego publicas tu app en la nube (ej: en un servidor público), deberás **añadir otro Redirect URI** con esa URL. Ejemplo:
  + https://midominio.com/redirect

**🔹 7. Verificar el flujo**

1. Corre tu app con python app.py.
2. Ve a http://localhost:8000.
3. Te redirigirá a Microsoft para loguearte.
4. Si todo está correcto → vuelve a http://localhost:8000/ mostrando el usuario autenticado.

README.md

|  |
| --- |
| # Watsonx Auth App  Aplicación Flask para autenticación con Microsoft (Azure Entra ID) antes de acceder al agente de Watsonx Orchestrate.  ## Instalación  ```bash  python -m venv .venv  source .venv/bin/activate # en Windows usar .venv\Scripts\activate  pip install -r requirements.txt  cp .env.example .env |

`Dockerfile` (opcional)

|  |
| --- |
| ---  ## 🔹 5. `Dockerfile` (opcional)  ```dockerfile  FROM python:3.10-slim  WORKDIR /app  COPY requirements.txt .  RUN pip install -r requirements.txt  COPY . .  CMD ["python", "app.py"] |

**Template básico para mostrar al usuario autenticado**

Crea la carpeta templates/ y dentro pon home.html:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Watsonx Orchestrate</title>  </head>  <body>  <h1>Bienvenido {{ user.get("name") }}</h1>  <p>Email: {{ user.get("preferred\_username") }}</p>  <a href="/logout">Cerrar sesión</a>  <hr />  <iframe src="https://url-de-tu-agente" style="width:100%;height:600px;border:0;"></iframe>  </body>  </html> |

Cuando en el README puse **"Ejecución"** y **"Visita"**, me refiero a que una vez que tengas los archivos listos en tu computadora, necesitas **levantar el servidor Flask** (la aplicación Python que maneja la autenticación).

**🔹 Paso 1: Abrir el proyecto en VS Code**

* Abre Visual Studio Code.
* Crea una carpeta nueva (ej: wxo\_auth\_app).
* Dentro de esa carpeta copia/crea los archivos que te pasé (app.py, requirements.txt, .env, etc.).

**Paso 2: Crear un entorno virtual (opcional pero recomendado)**

En la terminal de VS Code (o PowerShell/CMD):

|  |
| --- |
| python -m venv .venv |

Luego actívalo:

* En Windows (CMD/PowerShell):

|  |
| --- |
| .venv\Scripts\activate |

En macOS/Linux:

|  |
| --- |
| source .venv/bin/activate |

**Paso 3: Instalar dependencias**

Con el entorno activo, instala las librerías necesarias:

|  |
| --- |
| pip install -r requirements.txt |

**Paso 4: Configurar tu .env**

Crea un archivo llamado .env en la raíz del proyecto con tus credenciales de Azure:

|  |
| --- |
| CLIENT\_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  CLIENT\_SECRET=xxxxxxxxxxxxxxxxxxxxxxxx  TENANT\_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  FLASK\_SECRET=supersecretkey  REDIRECT\_PATH=/redirect |

**Paso 5: Ejecutar la aplicación**

Corre este comando en la terminal:

Ejecución

|  |
| --- |
| python app.py |

Vas a ver algo como:

|  |
| --- |
| \* Running on http://127.0.0.1:8000/ (Press CTRL+C to quit) |

**Paso 6: Visitar la app**

Abre tu navegador y entra en 👉 <http://localhost:8000>

* Si **no estás autenticado**, te redirigirá al login de Microsoft.
* Cuando pongas tus credenciales correctas, Microsoft te regresará a tu app (http://localhost:8000/redirect).
* Luego volverás a la página principal, donde ya verás tu nombre/email y el **iframe con Watsonx Orchestrate**. 🎉