



BACKEND CON PYTHON



AUTENTICACIÓN

- Importancia de la autenticación
- Mecanismos
- Implementación en Flask



AUTENTICACIÓN

- Importancia de la autenticación
- Mecanismos
- Implementación en Flask



IMPORTANCIA AUTENTICACIÓN

Ahora pensemos en que en un futuro nuestra aplicación será desplegada en un servidor web, para que pueda ser consultada por diferentes usuarios.

Teniendo esto en cuenta, es importante tener un mecanismo para revisar la identidad del usuario, es decir verificar que el usuario que se conecta a nuestra aplicación es quien dice ser.

- Seguridad: Es el primer paso para asegurar que los usuarios no autorizados no realicen ciertas acciones (lo complementaremos mas adelante cuando hablemos de autorización)
- Personalización: Al identifica usuarios se les pueden brindar experiencias diferentes según sus preferencias y roles.

IMPORTANCIA AUTENTICACIÓN

Ventajas:

- Seguridad: es el primer paso para manejar permisos y autorización. Si no identificamos que usuario está conectado intentando realizar cambios en nuestra aplicación, no podemos restringir el acceso a ciertos recursos.
- Personalización: al identificar el usuario, podremos saber si es administrador, moderador, usuario cliente, usuario vendedor, entre muchos otros y de esta manera ofrecerle diferentes funcionalidades al usuario.

Desventajas:

- Complejidad: entre más complejo sea el mecanismo de autenticación, más seguro será, pero, más difícil de usar para el usuario.
- Pérdida de credenciales: puede que los usuarios pierdan sus credenciales, esto implicaría tener que implementar mecanismos de recuperación y/o de autenticación alternativos.

AUTENTICACIÓN

- Importancia de la autenticación
- Mecanismos
- Implementación en Flask



MECANISMOS

Existen diferentes mecanismos de autenticación, algunos son más seguros que otros, pero usualmente entre más seguros, son más difíciles de usar:

- Nombre de usuario y contraseña: es uno de los más sencillo y también uno de los más usados, sin embargo, es vulnerable a ataques de fuerza bruta si las contraseñas no son lo suficientemente seguras.
- 2FA (2 Factor Authentication): agrega un paso adicional a la autenticación por usuario y contraseña (este se vuelve el paso 1), el nuevo suele ser un código aleatorio generado por una aplicación y enviado por mensaje de texto o correo electrónico.
- OAuth (Open Authorization): Permite a un usuario entregar su información sin entregar su contraseña. Es muy común en las redes sociales.

MECANISMOS

- OAuth 2.0 con tokens de acceso: extensión del protocolo OAuth que utiliza tokens de acceso para permitir que las aplicaciones accedan a recursos del usuario.
- Biometría: utiliza características físicas o comportamientos únicos de un individuo, como huellas dactilares, reconocimiento facial o voz.
- Tokens JWT (JSON Web Tokens): estándar abierto (RFC 7519) que define una manera compacta y autónoma para representar información entre dos partes. Eficiente y escalable, comúnmente utilizado en aplicaciones web y API RESTful.
- SAML (Security Assertion Markup Language): protocolo estándar para la autorización y autenticación de usuarios entre servicios. Ampliamente utilizado en entornos empresariales y federaciones de identidad.

AUTENTICACIÓN

- Importancia de la autenticación
- Mecanismos
- Implementación en Flask



IMPLEMENTACIÓN EN FLASK

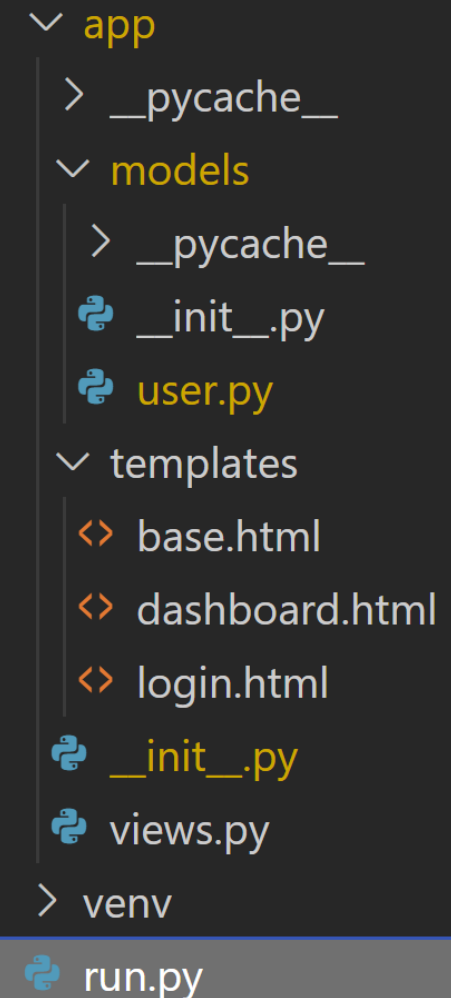
En Flask utilizaremos la librería flask.login para implementar la autenticación. Para esto, tendremos la siguiente organización dentro del proyecto “autenticación”.

Dentro del proyecto llamado autenticación tenemos:

- Una carpeta llamada app que tiene el proyecto de flask
- La carpeta venv que tiene el ambiente virtual (virtual environment)
- El archivo run.py, que utilizaremos para ejecutar la aplicación:

```
from app import app

if __name__ == '__main__':
    app.run(debug=True)
```



The image shows a file explorer view of a project named 'app'. The structure is as follows:

- app
 - __pycache__
 - models
 - __pycache__
 - __init__.py
 - user.py
 - templates
 - base.html
 - dashboard.html
 - login.html
 - __init__.py
 - views.py
 - venv
 - run.py

IMPLEMENTACIÓN EN FLASK

El archivo run.py hace un llamado al `__init__.py` dentro de app

Acá podemos ver como `__init__.py` utiliza `LoginManager` de la librería `flask_login`, para utilizar el sistema de autenticación usuario-contraseña.

La `SECRET_KEY` debe ser única de cada usuario y generada por el usuario, por ejemplo, con:

```
import os

# Genera una clave secreta segura
secret_key = os.urandom(24)
print(secret_key.hex())
```

```
from flask import Flask
from flask_login import LoginManager

app = Flask(__name__)
app.config['SECRET_KEY'] = '55e842c34f5a

login_manager = LoginManager(app)

from app import views
from app.models.user import users_db
@login_manager.user_loader
def load_user(user_id):
    return users_db.get(user_id)
```

IMPLEMENTACIÓN EN FLASK

El `__init__.py` dentro de `models` genera los imports necesarios

```
from .user import User, users_db
```

El archivo `user.py` genera la base de datos con los usuarios y sus contraseñas. Usa `UserMixin` de `flask_login` para que la contraseña sea almacenada de manera segura:

```
from flask_login import UserMixin

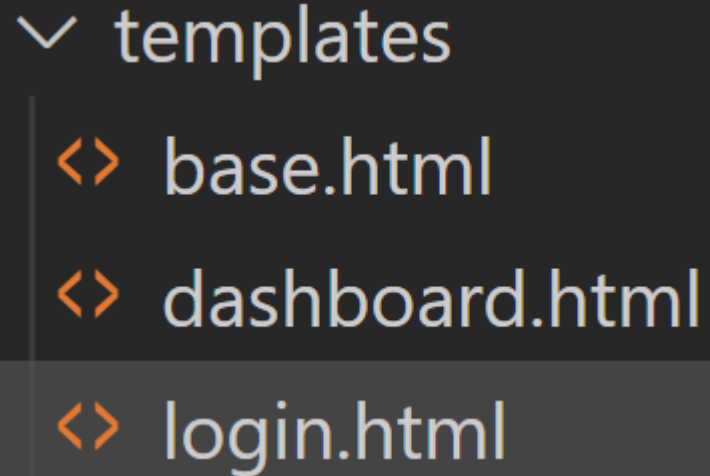
class User(UserMixin):
    def __init__(self, user_id, username, password):
        self.id = user_id
        self.username = username
        self.password = password

# Base de datos simulada para el ejemplo
users_db = {
    'zeus': User('zeus', 'Zeus', 'guau123'),
    'hera': User('hera', 'Hera', 'hera_password'),
    'poseidon': User('poseidon', 'Poseidon', 'ocean123'),
    'athena': User('athena', 'Athena', 'wisdom456'),
}
```

IMPLEMENTACIÓN EN FLASK

Los archivos dentro de templates, son los HTML que contiene 3 páginas:

- La primera la inicial, apenas uno se conecta al servidor web (base.html).
- La segunda, le pide al usuario ingresar su nombre y contraseña (login.html)
- La tercera, es la página a la que llega el usuario una vez se autentica.

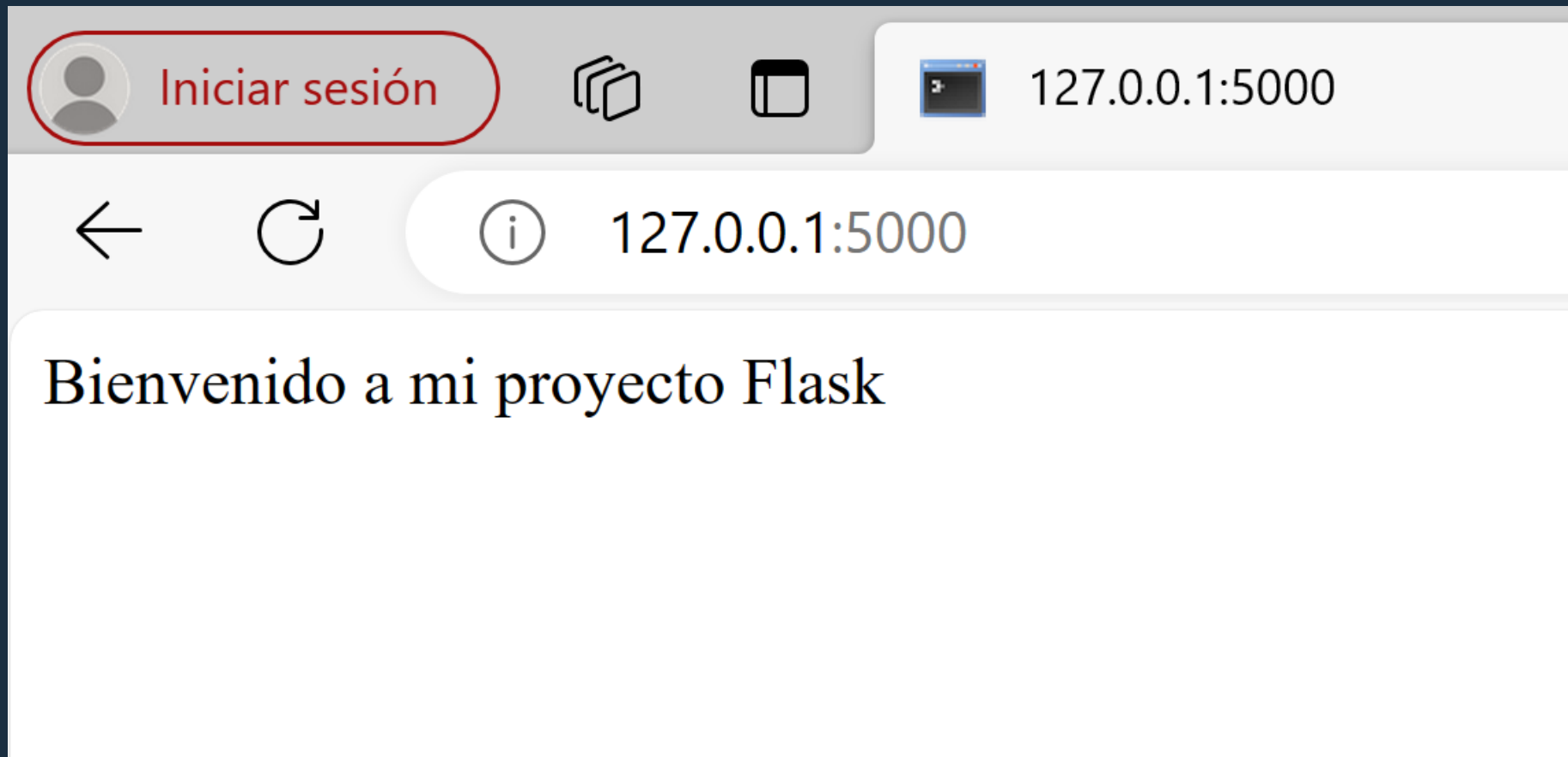


```
✓ templates
  <> base.html
  <> dashboard.html
  <> login.html
```

IMPLEMENTACIÓN EN FLASK

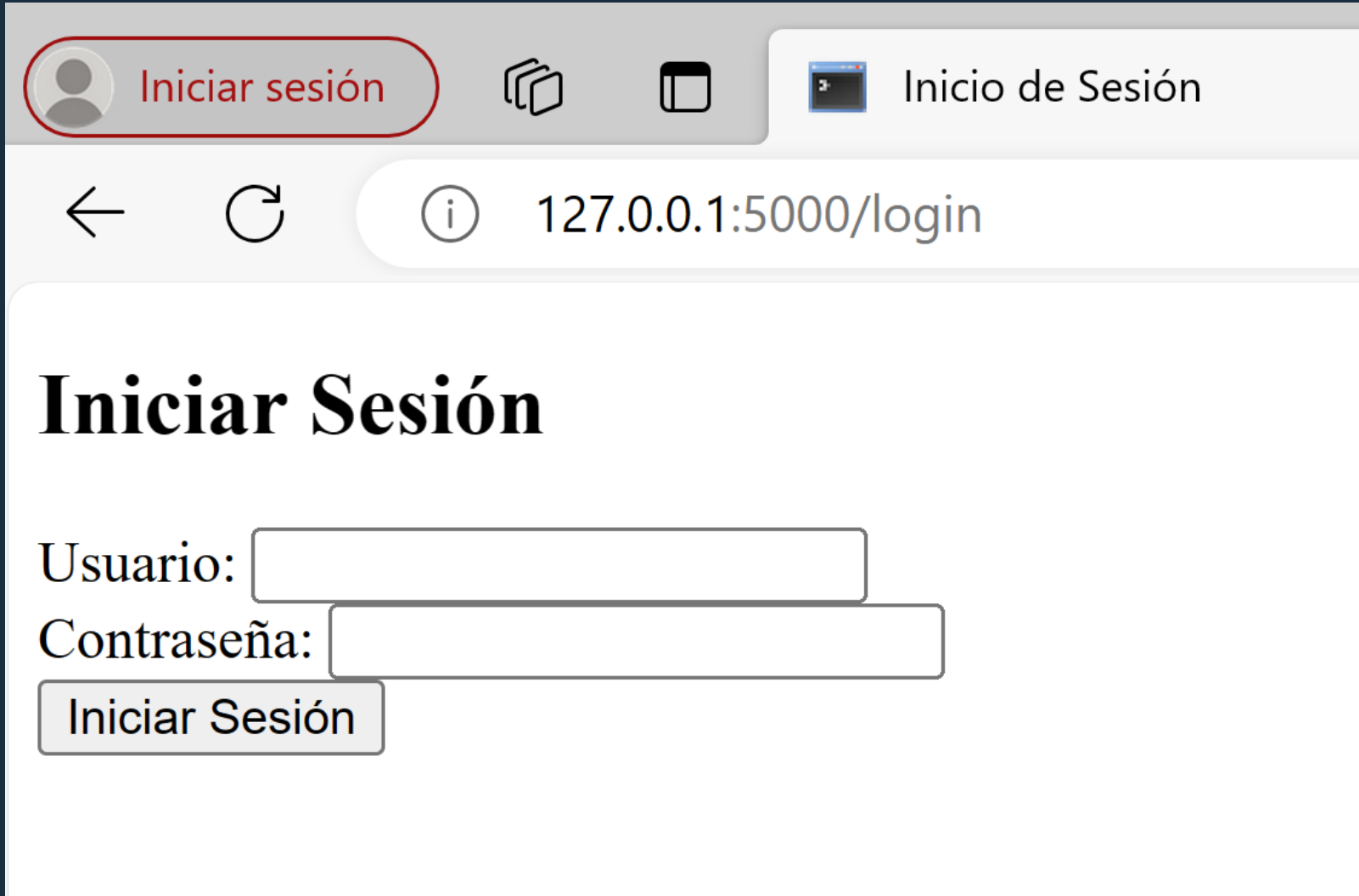
Para correr este proyecto no olvidemos instalar nuestro ambiente virtual, instalar flask y flask.login

Para ejecutarlo usaremos el comando “python run.py” y podemos abrir nuestra aplicación en un navegador web, y nos dirigirá a base.html



IMPLEMENTACIÓN EN FLASK

Para poder iniciar sesión incluimos /login

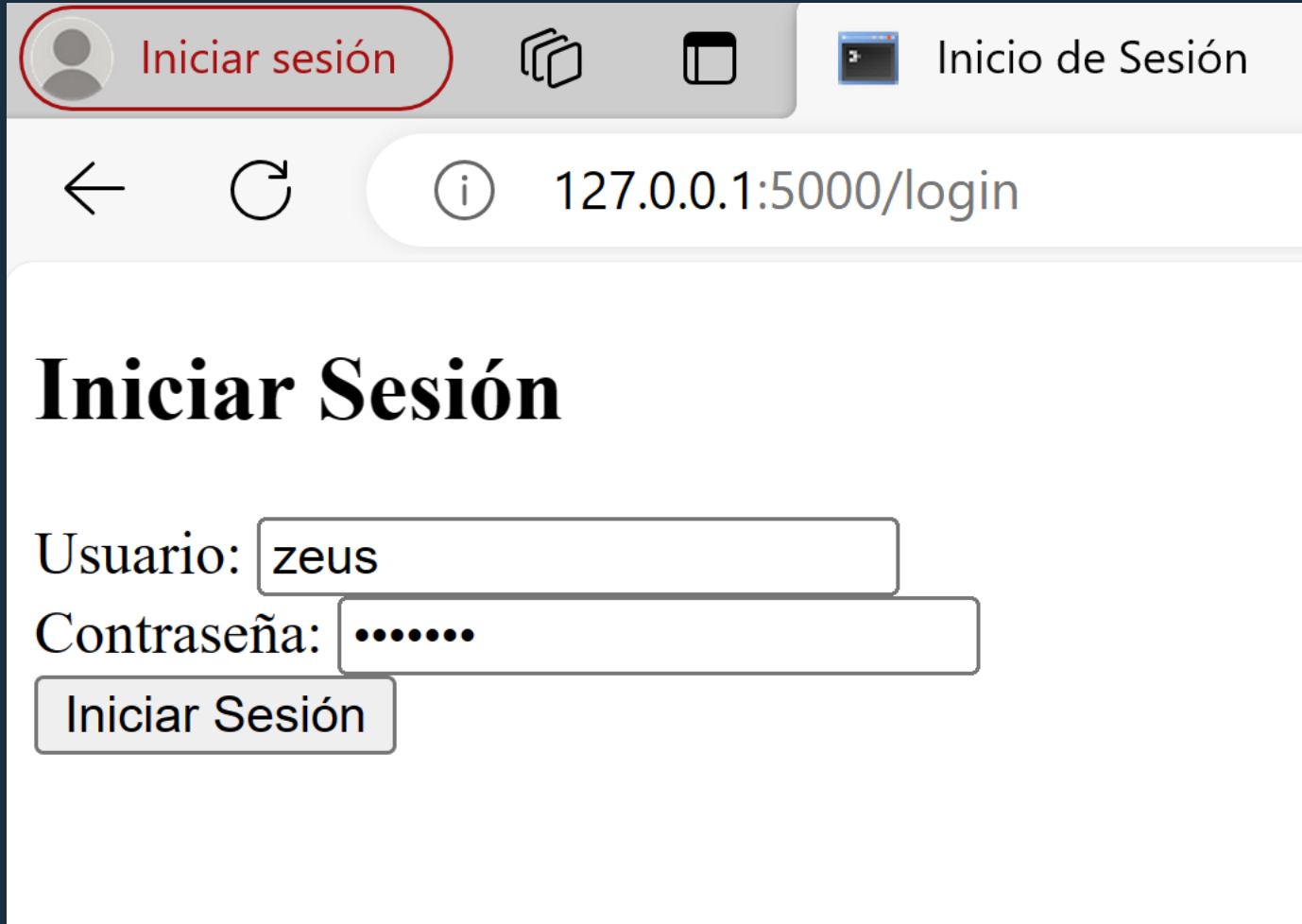


The screenshot shows a web browser window with the following elements:

- Browser Tabs:** The first tab is titled "Iniciar sesión" with a user icon. The second tab is titled "Inicio de Sesión" with a terminal icon.
- Address Bar:** Displays the URL "127.0.0.1:5000/login".
- Page Title:** "Iniciar Sesión" in a large, bold, black serif font.
- Form Fields:**
 - "Usuario:" followed by a text input field.
 - "Contraseña:" followed by a text input field.
- Submit Button:** A button labeled "Iniciar Sesión" located below the password field.

IMPLEMENTACIÓN EN FLASK

Llenamos el formulario (contraseña:guau123) y damos clic al botón iniciar sesión



The screenshot shows a web browser interface. At the top, there is a navigation bar with a red-outlined button labeled 'Iniciar sesión' next to a user icon, and a tab labeled 'Inicio de Sesión'. Below the navigation bar is the address bar showing the URL '127.0.0.1:5000/login'. The main content area has a heading 'Iniciar Sesión'. Below the heading are two input fields: 'Usuario:' with the text 'zeus' and 'Contraseña:' with masked characters '.....'. At the bottom left of the form is a button labeled 'Iniciar Sesión'.

Inicio de Sesión

← ↻ ⓘ 127.0.0.1:5000/login

Iniciar Sesión

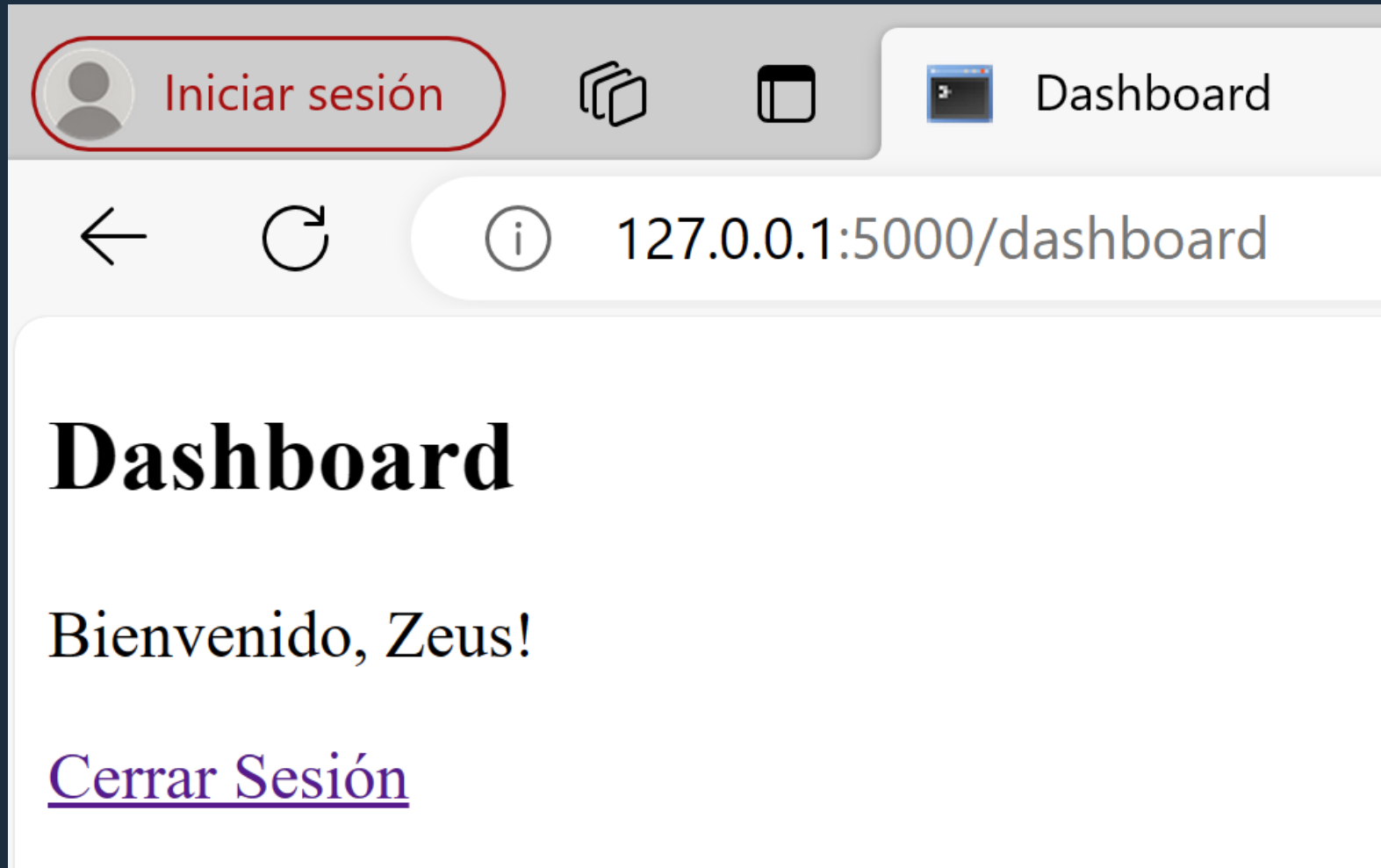
Usuario: zeus

Contraseña:

Iniciar Sesión

IMPLEMENTACIÓN EN FLASK

Al hacer clic en iniciar sesión, nos dirigirá a /dashboard.html



MANOS A LA OBRA

Taller 1 disponible en BloqueNeon

