|  |
| --- |
|  |
| Prueba de Caja Blanca |
| ***“Detección de Blancos Biológicos en Rosales”*** |
| Versión 1.1 |
|  |
| **Integrantes:**  Nicolas Cedillo  Alisson Clavijo  Lizzette Zapata  **Fecha 2024-01-11** |

**Ingreso del Usuario (LOGIN)**

**CÓDIGO FUENTE**

@api\_router.post("/login")

async def login(response: Response, form: OAuth2PasswordRequestForm = Depends()):

    """

    Endpoint para autenticar al usuario y generar un token de acceso.

    Args:

        response (Response): Objeto de respuesta HTTP.

        form (OAuth2PasswordRequestForm): Datos del formulario de inicio de sesión.

    Returns:

        Response: Objeto de respuesta HTTP con el token de acceso y tipo de token.

    Raises:

        HTTPException: Si la contraseña es incorrecta.

    """

    user: UserDB = search\_user(form.username)

    print(form.username)

    if not crypt.verify(form.password, user.user\_password):

        raise HTTPException(

            status\_code=status.HTTP\_401\_UNAUTHORIZED, detail="Incorrect password")

    expire = ((datetime.utcnow() + timedelta(minutes=ACCESS\_TOKEN\_DURATION)).timestamp())

    access\_token = {

        "sub": user.user\_name,

        "exp": expire

    }

    encoded\_token = jwt.encode(access\_token, SECRET, algorithm=ALGORITHM)

    response = JSONResponse(

        content={"access\_token": encoded\_token, "token\_type": "bearer"},

        status\_code=status.HTTP\_200\_OK

    )

    response.set\_cookie(

        key="access\_token",

        value=encoded\_token,

        expires=expire,

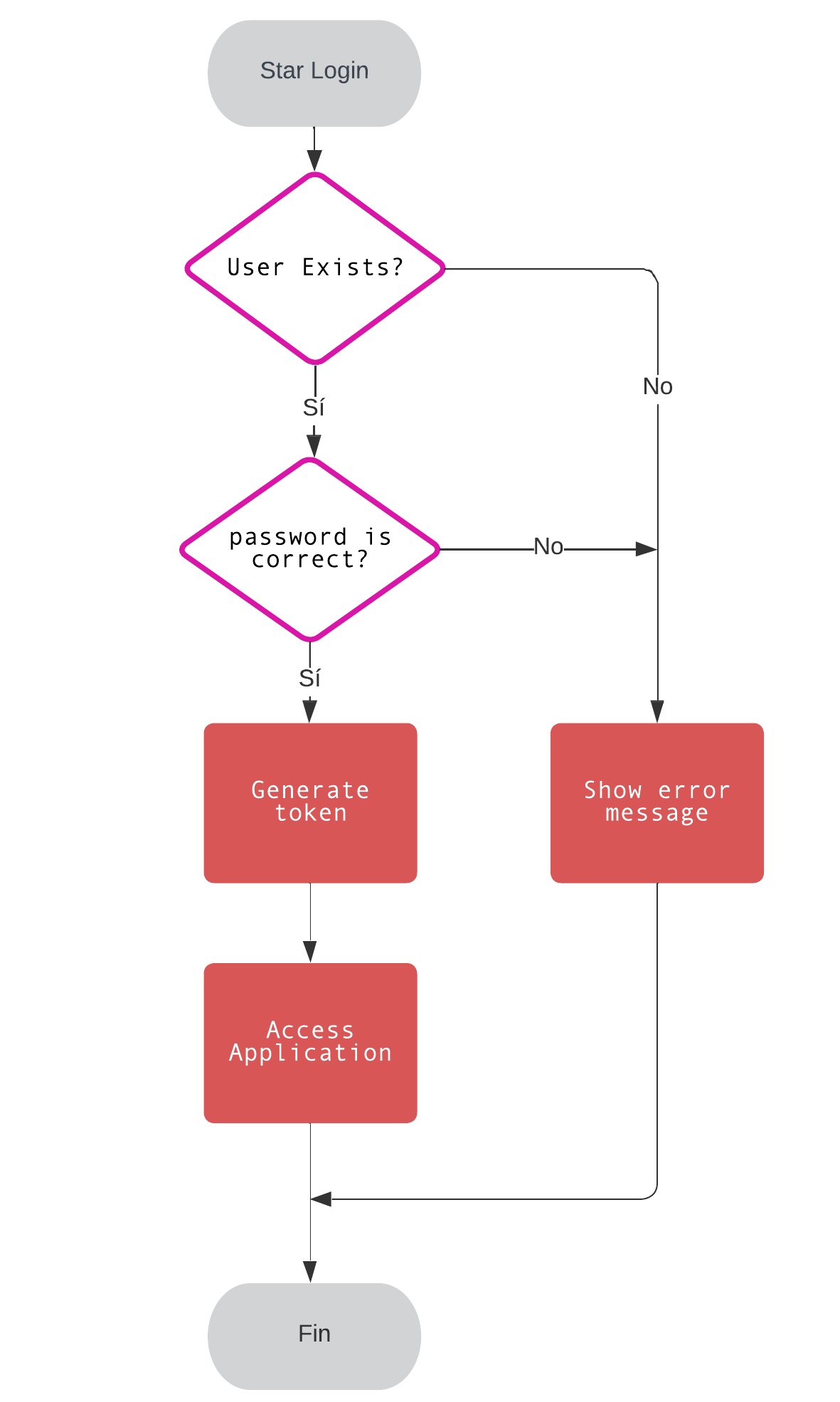
        httponly=True,

        samesite='lax',

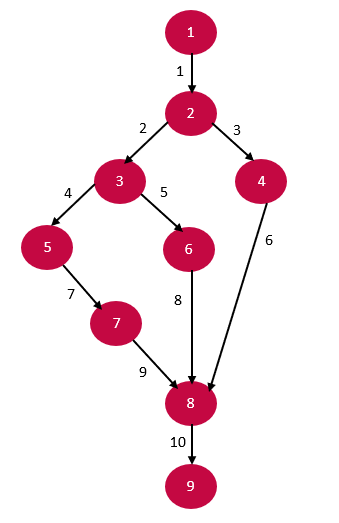
    )

    return response

**Diagrama de Flujo**

****

**GRAFO**

****

**RUTAS**

**R1:** 1**,**2,4,7,9,10

**R2:** 1,2,5,8,10

**R3:** 1,2,3,6,10

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = A – N + 2 = 3
* V(G) = 10 – 9 + 2 = 3

DONDE:

**A:** Número de aristas

**N:** Número de nodos