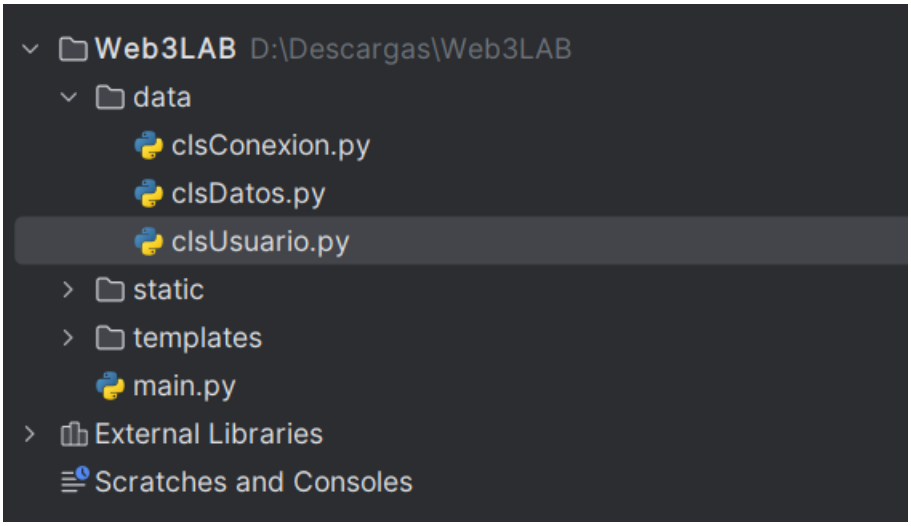
Codigo para crear lo siguiente

a. Registrar un usuario nuevo.

b. Autenticar el usuario, conocido como login

c. Terminar sesión de usuario, conocido como logout

d. Validar que la página de mantenimiento solo se presente si el usuario se autentico ante el sistema.



Código de clsConexion.py

```
      import pyodbc
from data.clsDatos import clsDatos
from data.clsUsuario import clsUsuario


class clsConexion():
    # Declara las variables para la conexion
    _servidor = '10.90.30.32'  # Recuerde cambiar la dirección y
contraseña
    _basedatos = 'datos'
    _usuario = 'sa'
    _contra = 'Utn123**'

    def __init__(self):
        pass

    def _conectar(self):
        try:
            _conex = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL
Server};'
                                    'SERVER=' + self._servidor +
```

```
                                    ';DATABASE=' + self._basedatos +
                                    ';UID=' + self._usuario +
                                    ';PWD=' + self._contra)
        except Exception as err:
            print(err)
        return _conex

    def logged(self, dato):
        AuxSql = "select * from usuarios where (nombre = '{0}' and
contrasena = '{1}' )".format(dato.Usuario, dato.Contrase)
        try:
            _conex = self._conectar()
            with _conex.cursor() as cursor:
                cursor.execute(AuxSql)

            _conex.close()

        except Exception as err:
            print(err)
        if cursor.rowcount == 0:
            return False
        else:
            return True

    def agregar(self, dato):
        estado = False
        AuxSql = "insert into datos(texto, descripcion)
values('{0}','{1}')".format(dato.Texto, dato.Descripcion)
        try:
            _conex = self._conectar()
            with _conex.cursor() as cursor:
                cursor.execute(AuxSql)

            _conex.close()
            estado = True
        except Exception as err:
            print(err)
        return estado

    def registrar(self, dato):
        estado = False
        AuxSql = "insert into usuarios(nombre, contrasena) values('{0}',
'{1}')".format(dato.Usuario, dato.Contrase)
        try:
            _conex = self._conectar()
            with _conex.cursor() as cursor:
                cursor.execute(AuxSql)

            _conex.close()
            estado = True
        except Exception as err:
            print(err)
        return estado

    def editar(self, dato):
        estado = False
        AuxSql = "update datos set texto = '{1}', descripcion = '{2}'
```

```
where id = {0}".format(dato.ID, dato.Texto,

dato.Descripcion)
        try:
            _conex = self._conectar()
            with _conex.cursor() as cursor:
                cursor.execute(AuxSql)

            _conex.close()
            estado = True
        except Exception as err:
            print(err)
        return estado

    def borrar(self, ide):
        estado = False
        AuxSql = "delete datos where id = {0}".format(ide)
        try:
            _conex = self._conectar()
            with _conex.cursor() as cursor:
                cursor.execute(AuxSql)

            _conex.close()
            estado = True
        except Exception as err:
            print(err)
        return estado

    def consultar(self, ide=None):
        data = ''
        salida = []

        try:
            _conex = self._conectar()
            with _conex.cursor() as cursor:
                if ide is None:
                    cursor.execute("Select * from datos")
                else:
                    cursor.execute("Select * from datos where id =
{0}".format(ide))
                data = cursor.fetchall()

            _conex.close()
        except Exception as err:
            print(err)

        for tupla in data:
            salida.append(clsDatos(tupla[0], tupla[1], tupla[2]))

        return salida
```

Código de clsUsuario.py

```
class clsUsuario():
    def __init__(self, usu=None, contra=None):
        self._Usuario = usu
        self._Contrasena = contra

    def _getUsu(self):
        return self._Usuario

    def _setUsu(self, usu):
            self._Usuario = usu

    def _getContra(self):
        return self._Contrasena

    def _setContra(self, contra):
            self._Contrasena = contra


    # Encapsulated property
    Usuario = property(_getUsu, _setUsu)
    Contrase = property(_getContra, _setContra)
```

Codigo de clsDatos.py

```
class clsDatos():
    # Constructor function
    def __init__(self, id=None, texto=None, descripcion=None):
        self._id = id
        self._text = texto
        self._descrip = descripcion

    # Local get/set function set
    def _getID(self):
        return self._id

    def _setID(self, id):
        self._id = id

    def _getText(self):
        return self._text

    def _setText(self, texto):
        self._text = texto

    def _getDescrip(self):
        return self._descrip

    def _setDescrip(self, descripcion):
        self._descrip = descripcion
```

```
    # Encapsulated property
    ID = property(_getID, _setID)
    Texto = property(_getText, _setText)
    Descripcion = property(_getDescrip, _setDescrip)
```

Código del main

```
from flask import Flask, render_template, request, redirect, url_for,
session

app = Flask(__name__)

app.secret_key = "Secure_key"
from data.clsConexion import clsConexion
from data.clsDatos import clsDatos
from data.clsUsuario import clsUsuario

conex = clsConexion()

@app.route('/', methods=['POST', 'GET'])
def index():
    if 'user' in session != None:
        return render_template('index.html', datos=conex.consultar())
    else:
        return render_template('login.html')

@app.route('/login', methods= ['POST','GET'])
def login():
    if request.method == "POST":
        if conex.logged(clsUsuario(request.form['txtUsu'],
request.form['txtContra'])):
            user = request.form['txtUsu']
            session['user'] = user
            app.logger.debug("usario")
        else:
            app.logger.debug("usario no registrado")
        return redirect(url_for("index"))
    else:
        return redirect(url_for('login'))


@app.route('/registrar',methods=['GET', 'POST'])
def registrar():
    if request.method == "POST":
        if conex.registrar(clsUsuario(request.form['txtUsu'],
request.form['txtContra'])):
            app.logger.debug("Usuario ha sido registrado")
        else:
            app.logger.debug("Error al registrar usuario")
        return redirect(url_for('index'))
    else:
        return render_template('registrar.html')
```

```
@app.route('/agregar', methods=['GET', 'POST'])
def agregar():
    if request.method == "POST":
        if conex.agregar(clsDatos(0, request.form['txtTexto'],
request.form['txtDescrip'])):
            app.logger.debug("Datos almacenados correctamente")
        else:
            app.logger.debug("Se presentó un problema con los datos")
        return redirect(url_for("index"))
    else:
        return render_template('agregar.html')

@app.route('/modificar/<int:ide>', methods=['GET'])
def modificar(ide):
    return render_template('modificar.html', datos=conex.consultar(ide))


@app.route('/exec_modificar', methods=['POST'])
def exec_modificar():
    if conex.editar(clsDatos(request.form['txtID'],
request.form['txtTexto'], request.form['txtDescrip'])):
        app.logger.debug("Datos modificados correctamente")
    else:
        app.logger.debug("Se presentó un problema con los datos")
    return redirect(url_for('index'))


@app.route('/exec_eliminar/<int:ide>', methods=['GET'])
def exec_eliminar(ide):
    if conex.borrar(ide):
        app.logger.debug("Datos eliminado correctamente")
    else:
        app.logger.debug("Se presentó un problema con los datos")
    return redirect(url_for('index'))


@app.route('/logout')
def logout():
    session.pop('user', default=None)
    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(host='localhost', port=5001, debug=True)
```

Guía en Django

1. **Instalación de Python**: Asegúrate de tener Python instalado en tu sistema. Si no lo tienes, puedes descargarlo desde [python.org](https://www.python.org/downloads/).

2. **Instalación de Django**: Una vez que tengas Python, abre tu terminal o consola de comandos y escribe:

```bash
pip install django
```

Creación de un nuevo proyecto Django:

3. **Crear un nuevo proyecto**:

```bash
django-admin startproject myproject
```

4. **Cambiar al directorio del proyecto**:

```bash
cd myproject
```

5. Validar sesión antes de presentar página de datos:

5. **Configura las aplicaciones necesarias en `INSTALLED_APPS`**:

Abre el archivo `myproject/settings.py` y verifica que tienes:

```python
INSTALLED_APPS = [
```

```
    ...
    'django.contrib.auth',
    'django.contrib.contenttypes',
    ...
]
```

6. **URLs**:

   Abre el archivo `myproject/urls.py` y agrega las rutas de autenticación:

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),
]
```

8. **Crear una nueva app**:

   En la terminal, estando en el directorio `myproject`, escribimos:

```bash
python manage.py startapp demo
```

9. **Instala Bootstrap** para un diseño sencillo:

```bash
pip install django-bootstrap4
```

Luego, añadimos `'bootstrap4'` a `INSTALLED_APPS` en `myproject/settings.py`.

10. **Definir un modelo**:

En `demo/models.py`, crea un modelo sencillo:

```python
from django.db import models

class Item(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
```

11. **Habilitar la app y migrar**:

En `myproject/settings.py`, añade `'demo'` a `INSTALLED_APPS`.

Luego, en la terminal:

```
python manage.py makemigrations
```

python manage.py migrate

12. **Crear vistas y formularios CRUD**:

En `demo/forms.py` (deberás crear este archivo):

```
from django import forms
from .models import Item

class ItemForm(forms.ModelForm):
    class Meta:
        model = Item
        fields = ['name', 'description']
```

En `demo/views.py`, añade las vistas CRUD (es un ejemplo básico, puedes expandirlo según tus necesidades):

```python
from django.shortcuts import render, redirect
from .models import Item
from .forms import ItemForm

def create_item(request):
    if request.method == "POST":
        form = ItemForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('items_list')
    else:
```

```
        form = ItemForm()
    return render(request, 'create_item.html', {'form': form})

# Agrega vistas para leer, actualizar y eliminar siguiendo una lógica similar.
```

13. **Configura URLs**:

   En `demo/urls.py` (deberás crear este archivo):

```python
from django.urls import path
from . import views

urlpatterns = [
    path('create/', views.create_item, name='create_item'),
    # Añade rutas para leer, actualizar y eliminar.
]
```

   En `myproject/urls.py`, añade:

```python
path('demo/', include('demo.urls')),
```

14. **Plantillas**:

Crea un directorio llamado `templates` dentro de `demo/`. Allí, puedes agregar plantillas HTML para cada vista. Por ejemplo, para la vista `create_item`, crea un archivo `create_item.html` que contenga el formulario.

8. Módulo de usuarios:

Ya tenemos las rutas de autenticación gracias a `django.contrib.auth.urls`, que proporciona rutas para iniciar sesión, cerrar sesión, restablecer la contraseña, etc.

Este punto es más teórico y no se traduce directamente en código. Dependerá de las tecnologías que estés comparando y el tipo de análisis que desees hacer.