Unidad 5

PROGRAMACIÓN ORIENTADA A OBJETOS

Departamento de Informática
FCEFN – UNSJ

- Licenciatura en Ciencias de la Computación
- Licenciatura en Sistemas de Información
- Tecnicatura Universitaria en Programación Web

Año 2025

Aplicaciones Web con Python

PROPÓSITO DE LA UNIDAD

Que el estudiante adquiera hábitos para la investigación, a través del análisis y estudio de material vinculado al desarrollo de Aplicaciones Web, y para la discusión activa de sus hallazgos en los grupos de estudio.

Bibliografía

- Documento de cátedra.
- Flask Web Development Developing Web Applications with Python. Miguel Grinberg. O'Reilly Media, segunda edición. 2018.
- Aprendizaje Flask. Ebook. https://riptutorial.com/Download/flask-es.pdf

Enlaces de interés

- https://flask.palletsprojects.com/en/2.0.x/
- http://www.manualweb.net/flask/mi-primer-programa-flask/
- https://www.tutorialspoint.com/flask/index.htm
- https://docs.sqlalchemy.org/en/13/orm/tutorial.html
- https://flask-sqlalchemy.palletsprojects.com/en/2.x/config/

Un sistema basado en la web es una colección de páginas web que se vinculan entre sí y que contienen distintos recursos (texto, imágenes, videos y otros) para brindar un servicio a los usuarios.

Estructura de una aplicación web

Es un caso de arquitectura **cliente-servidor**

Flask
web development,
one drop at a time

CSS

HTML

SQLAlchemy

Aplicaciones Web con Python

Framework

Flask

$ pip install Flask

Compatible con Python 3.8 y versiones posteriores

Librerías como Werkzeug y Jinja2

En el desarrollo de una aplicación flask se utilizan distintos entornos

Editor

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def saludo():
    return 'Mi primera aplicación con Flask!'

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

app.py

APLICACIÓN FLASK

Flask

Consola

Anaconda Prompt - python app.py

```
(base) C:\Users\TEMP\web\primerPrograma>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [03/Mar/2020 17:26:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/Mar/2020 17:26:02] "GET /favicon.ico HTTP/1.1" 404 -
```

Flask

Navegador

Flask

HTML

HyperText Markup Language

Aplicación web

Conjunto de páginas web

Cada página tiene una dirección

Decorador app.route

```
<!DOCTYPE html>
<html>
    <head>
        <title>Flask</title>
    </head>
    <body>
        <h1>Mi primera aplicación Flask</h1>
        <p>Esto es una página web</p>
    </body>
</html>
```

HTML

- Un lenguaje de marcas
- Interpretado por los navegadores

Estructura de la aplicación web con flask

THE ART OF ROUTING IN
FLASK

La vinculación de la dirección con la página web se logra con el método

render_template

```
from flask import render_template

@app.route('/')
def saludo():
    return render_template(‘inicio.html’)
```

Formularios
Objeto request

El motor de template Jinja2 que incluye Flask permite formar el contenido HTML

Delimitador
{{...}}

para incluir datos recibidos en el template

Delimitador
{% ... %}

para incluir expresiones que alteran el flujo secuencial del HTML

Ejemplo

El jefe de proyecto nos ha solicitado implementar parte de una aplicación web.

La solicitud consiste en una página web que permita ingresar el nombre, correo y contraseña de un usuario.

Si se han ingresado los tres datos, se debe dirigir a otra página que muestre un saludo que contenga el nombre y el correo del usuario.

En caso de faltar algún dato se debe redirigir a la página de ingreso.

Cliente

Servidor

templates → nuevo_usuario.html

templates → bienvenida.html

app.py

```
@app.route('/')
def usuario():
```

```
@app.route('/bienvenida')
def bienvenida():
```

render_template('nuevo_usuario.html')

```
<form action = "http://localhost:5000/bienvenida" ...
```

```
render_template('bienvenida.html',
datos=datosform, hora= datetime.now().hour)
```

parámetros

nuevo_usuario.html

```
<!DOCTYPE html>
<html>
   <body>
      <h1>Nuevo Usuario</h1>
      <form action = "http://localhost:5000/bienvenida" method = "post">
         <label for = "nombre">Nombre</label><br>
         <input type = "text" name = "nombre" placeholder = "Nombre" />
         <br>
         <label for = "email">Correo Electrónico</label><br>
         <input type = "text" name = "email" placeholder = "email" />
         <br>
         <label for = "password">Contraseña</label> <br>
         <input type = "password" name = "password" placeholder = "password"/> <br><br>
         <input type = "submit" value = "Guardar" />
         <br><br>
      </form>
   </body>
</html>
```

Aplicación Flask

```
from flask import Flask, render_template, request
from datetime import datetime
app = Flask(__name__)

@app.route('/')
def usuario():
   return render_template('nuevo_usuario.html')

@app.route('/bienvenida',methods = ['POST', 'GET'])
def bienvenida():
  if request.method == 'POST':
     if request.form['nombre'] and request.form['email'] and   request.form['password']:
       datosform = request.form
       return render_template('bienvenida.html', datos=datosform, hora= datetime.now().hour)
     else:
       return render_template('nuevo_usuario.html’)

if __name__ == '__main__':

   app.run(debug = True)
```

bienvenida.html

```
<!doctype html>
<html>
   <body>
       {%  if hora < 12 %}
           <h1> Buenos días {{ datos.nombre }} </h1>
       {% elif hora < 21 %}
           <h1> Buenas tardes {{ datos.nombre }} </h1>
           {% else %}
           <h1> Buenas noches {{ datos.nombre }} </h1>
       {% endif %}
      <h2> Recibirás las notificaciones en el correo {{ datos.email }} </h2>

   </body>
</html>
```

Plantilla base

Plantilla base

La estructura común a un conjunto de páginas se puede definir en una plantilla base.

Jinja2 provee la etiqueta

{% block %}

...

{% endblock %}

base_template.html

```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <title>{% block title %}{% endblock %}</title>
6      <link rel="stylesheet" href="{{ url_for("static", filename="css/estilos.css") }}">
7  </head>
8  <body>
9      {% block content %}{% endblock %}
10       <div class="myDiv">
11           <br><br><a href = "{{ url_for('inicio') }}" > Inicio </a><br><br>
12       </div>
13 </body>
14 </html>
```

nuevo_usuario.html

```
{% extends "base_template.html" %}
{% block title %}Crear Usuario{% endblock %}
{% block content %}
     <h1>Registro de Usuario</h1>
      <hr/>
      <form action = "{{ request.path }}" method = "post" class = "container">
         <label for = "nombre">Nombre</label><br>
         <input type = "text" name = "nombre" placeholder = "Nombre" /><br>

         <label for = "email">Correo Electrónico</label><br>
         <input type = "text" name = "email" placeholder = "email" /><br>

         <label for = "password">Contraseña</label><br>
         <input type = "password" name = "password" placeholder = "password"/><br>
         <br>
         <input type = "submit" value = "Guardar" />

      </form>
{% endblock %}
```

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <link rel="stylesheet" href="{{ url_for("static", filename="css/estilos.css") }}">
</head>
<body>
    {% block content %}{% endblock %}
      <div class="myDiv">
          <br><br><a href = "{{ url_for('inicio') }}" > Inicio </a><br><br>
      </div>
</body>
</html>
```

```jinja2
{% extends "base_template.html" %}
{% block title %}Crear Usuario{% endblock %}
{% block content %}
    <h1>Registro de Usuario</h1>
     <hr/>
     <form action = "{{ request.path }}" method = "post" class = "container">
        <label for = "nombre">Nombre</label><br>
        <input type = "text" name = "nombre" placeholder = "Nombre" /><br>

        <label for = "email">Correo Electrónico</label><br>
        <input type = "text" name = "email" placeholder = "email" /><br>

        <label for = "password">Contraseña</label><br>
        <input type = "password" name = "password" placeholder = "password"/><br>
        <br>
        <input type = "submit" value = "Guardar" />

     </form>
{% endblock %}
```

```html
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title> Crear usuario </title>
    <link rel="stylesheet" href="/static/css/estilos.css">
  </head>
  <body>

<h1> Registro de Usuario</h1>
<hr/>
 <form action="/nuevo_usuario.html" method='post' class= 'container'>
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre" placeholder= "Nombre"><br><br>
  <label for="email">Correo Electrónico:</label>
  <input type="text" name="email" placeholder="email" ><br><br>
  <label for="password">Contraseña:</label>
  <input type="password"  name="password" placeholder= "Contraseña"><br><br>
  <input type="submit" value="Guardar">
</form>

    <div class="myDiv">
      <br> <br> <a href= "/"> Inicio </a> <br> <br>
    </div>
</form>
  </body>
</html>
```

Sesiones

Sesiones

- Tiempo que transcurre desde que un cliente ingresa a la aplicación hasta que cierra la aplicación.
- Los datos de la sesión son almacenados en el servidor.
- El servidor los firma en modo cifrado, para esto Flask requiere que se defina una SECRET_KEY.
- Importar la clase sesión.
- Una sesión es un objeto diccionario

Sesiones - Ejemplo

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

app = Flask(__name__)
app.secret_key = 'CLAVE_SECRETA'

@app.route("/")
def index():
    if not session.get("name"):
        return redirect("/login")
    return render_template('index.html')


@app.route("/login", methods=["POST", "GET"])
def login():
    if request.method == "POST":
        session["name"] = request.form.get("name")
        return redirect("/")
    return render_template("login.html")


@app.route("/logout")
def logout():
    session["name"] = None
    return redirect("/")


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

```jinja
{% extends "plantilla.html" %}

{% block y %}

{% if session.name %}
    Usted se ha registado como {{ session.name }} <br> <br> <a href="/logout">Cerrar sesión</a>.
{% else %}
    No se ha registrado. <a href="/login">Iniciar sesión</a>.
{% endif %}

{% endblock %}
```

Cascading Style Sheet

Al usar css hay que tener en cuenta

estilos.css

```
body {
  margin: 0;  padding: 0;  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #464;}

header {  background-color: #DFB887;  height: 10%;  width: 100%;  opacity: .9;  margin-bottom: 2%;}

h1 {    font-family: serif;    text-align: center;color: #377ba8;    }

.container {padding-top: 2%;  width: 50%;  margin: 0 auto;  background-color: #DFB887;}

.container label { font-weight: bold; margin-bottom: 2em;    margin-left: 2em;  color: #377ba8;}
.container input { margin-bottom: 1em;    margin-left: 2em;}
.container button {margin-bottom: 1em;    margin-left: 2em;}
.container textarea { min-height: 12em;   resize: vertical; }
```

nuevo_usuario.html

```html
<html>
 <head>
   <title>Crear Usuario</title>
   <link rel="stylesheet" href="{{url_for('static', filename= 'css/estilos.css')}}">
 </head>
 <body>
      <h1>Nuevo Usuario</h1>
      <form action = "http://localhost:5000/bienvenida" method = "post" class= "container">
         <label for = "nombre" >Nombre</label><br>
         <input type = "text" name = "nombre" placeholder = "Nombre" /><br>

         <label for = "email">Correo Electrónico</label><br>
         <input type = "text" name = "email" placeholder = "email" /><br>

         <label for = "password">Contraseña</label><br>
         <input type = "password" name = "password" placeholder = "password"/><br>
        <br>
         <input type = "submit" value = "Guardar" />
        <br><br>
      </form>
   </body>
</html>
```

Resultado

Crear Usuario
127.0.0.1:5000
Nuevo Usuario
Nombre
Nombre
Correo Electrónico
email
Contraseña
password
Guardar

Flask

SQLAlchemy

Las librerías de mapeo objeto – relacional
Facilitan la comunicación entre
Una aplicación orientada a objetos y una base de datos relacional.

Id	Nombre	Apellido	edad
1	Juan	Pérez	15

SQL - Structured Query Language

Conceptos de Base de datos

Una base de datos contiene una o más tablas

Una **clave primaria** es un campo o conjunto de campos de una tabla que **identifican cada fila.**

Campos

codigo	nombre	titulo	duracion
7	Licenciatura en Ciencias de la Computación	Licenciado en Ciencias de la Computación	10
8	Licenciatura en Sistemas de Información	Licenciado en Sistemas de Información	10
12	Tecnicatura Universitaria en Programación Web	Técnico Universitario en Programación Web	6

Registros

Tabla "Carreras"

SQLAlchemy

- Asociar clases Python definidas por el usuario con tablas de bases de datos
- Asociar instancias de las clases (objetos) con filas (registros) en sus tablas correspondientes.
- Sincroniza de manera transparente todos los cambios de estado entre los objetos y sus registros relacionadas, denominado unidad de trabajo.
- Sistema para expresar consultas a la base de datos en términos de las clases definidas por el usuario y las relaciones definidas entre sí.

```
pip install Flask-SQLAlchemy
```

Admite los sistemas de administración de bases de datos más comunes

- PostgreSQL
- MySQL
- Oracle
- Microsoft SQL Server
- SQLite

Aplicación Flask-SQLAlchemy

La empresa de software donde trabajan está desarrollando una aplicación Flask para el dictado de un curso on-line con usuarios registrados. Les pide a ustedes que desarrollen el incremento de "Comentarios" para dicha aplicación, cumpliendo los siguientes requerimientos o requisitos:

R1. Registrar un comentario que realiza un usuario registrado.

R2. De un comentario se registra la fecha y hora en que se realiza, el texto del comentario y el usuario que lo realiza.

R3. Un usuario puede hacer varios comentarios.

R4. Al momento de realizar un comentario, el usuario debe ingresar su correo y contraseña, si estos datos se corresponden con un usuario registrado se le permite comentar.

R5. Listar todos los comentarios incluyendo los datos de los usuarios que lo hizo.

R6. Listar los comentarios para un usuario en particular.

PARÁMETROS DE CONFIGURACIÓN

config.py

```
SECRET_KEY = "GDtfDCFYjD“

SQLALCHEMY_DATABASE_URI = 'sqlite:///datos.sqlite3’

SQLALCHEMY_TRACK_MODIFICATIONS = False
```

Definir el modelo

models.py

```python
from __main__ import app
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy(app)

class Usuario(db.Model):
    __tablename__ = 'usuario'
    id = db.Column(db.Integer, primary_key=True)
    nombre = db.Column(db.String(80), nullable=False)
    correo = db.Column(db.String(120), unique=True, nullable=False)
    clave = db.Column(db.String(120), nullable=False)
    comentario = db.relationship('Comentario', backref='usuario', cascade="all, delete-orphan")

class Comentario(db.Model):
    __tablename__ = "comentario"
    id = db.Column(db.Integer, primary_key=True)
    fecha = db.Column(db.DateTime)
    contenido = db.Column(db.Text)
    usuario_id = db.Column(db.Integer, db.ForeignKey('usuario.id'))
```

La aplicación - I

app.py

```
1   from datetime import datetime
2   from flask import Flask, request, render_template
3   from flask_sqlalchemy import SQLAlchemy
4   from werkzeug.security import generate_password_hash, check_password_hash
5
6   app = Flask(__name__)
7   app.config.from_pyfile('config.py')
8
9   from models import db
10  from models import Usuario, Comentario
11
```

La aplicación - II

app.py

```
11
12  @app.route('/')
13  def inicio():
14      return render_template('inicio.html')
15  @app.route('/nuevo_usuario', methods = ['GET','POST'])
16  def nuevo_usuario():
26
27  @app.route('/nuevo_comentario', methods = ['GET','POST'])
28  def nuevo_comentario():
44
45  @app.route('/ingresar_comentario', methods = ['GET', 'POST'])
46  def ingresar_comentario():
56
57  @app.route('/listar_comentarios')
58  def listar_comentarios():
60
61  @app.route('/listar_comentarios_usuario', methods = ['GET', 'POST'])
62  def listar_comentarios_usuario():
71
```

La aplicación - III

app.py

```
72
73 if __name__ == '__main__':
74     db.create_all()
75     app.run(debug = True)
```

La aplicación

```python
from datetime import datetime
from flask import Flask, request, render_template
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)
app.config.from_pyfile('config.py')

from models import db
from models import Usuario, Comentario

@app.route('/')
def inicio():
    return render_template('inicio.html')
@app.route('/nuevo_usuario', methods = ['GET','POST'])
def nuevo_usuario():

@app.route('/nuevo_comentario', methods = ['GET','POST'])
def nuevo_comentario():

@app.route('/ingresar_comentario', methods = ['GET', 'POST'])
def ingresar_comentario():

@app.route('/listar_comentarios')
def listar_comentarios():

@app.route('/listar_comentarios_usuario', methods = ['GET', 'POST'])
def listar_comentarios_usuario():


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

inicio.html

```html
<!DOCTYPE html>
<html lang = "en">
   <head>
    <title>Aplicación</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/estilos.css') }}">
  </head>
   <body>
     <h1> Comentarios</h1>
     <ul>
        <li> <a href = "{{ url_for('nuevo_usuario') }}"> Registrarse como Usuario </a></li><br>
        <li> <a href = "{{ url_for('nuevo_comentario') }}"> Comentar</a></li><br>
        <li> <a href = "{{ url_for('listar_comentarios') }}"> Listar Todos los Comentarios </a></li> <br>
        <li> <a href = "{{ url_for('listar_comentarios_usuario') }}"> Listar Comentarios </a></li><br>
    </ul>
   </body>
</html>
```

Ruta nuevo_comentario

```
33  @app.route('/nuevo_comentario', methods = ['GET','POST'])
34  def nuevo_comentario():
35      if request.method == 'POST':
36          if  not request.form['email'] or not request.form['password']:
37              return render_template('error.html', error="Por favor ingrese los datos requeridos")
38          else:
39              usuario_actual= Usuario.query.filter_by(correo= request.form['email']).first()
40              if usuario_actual is None:
41                  return render_template('error.html', error="El correo no está registrado")
42              else:
43                  verificacion = check_password_hash(usuario_actual.clave, request.form['password'])
44                  if (verificacion):
45                      return render_template('ingresar_comentario.html', usuario = usuario_actual)
46                  else:
47                      return render_template('error.html', error="La contraseña no es válida")
48      else:
49          return render_template('nuevo_comentario.html')
50
```

nuevo_comentario.html

```
1  {% extends "base_template.html" %}
2  {% block title %}Comentar{% endblock %}
3  {% block content %}
4        <h1>Nuevo Comentario</h1>
5        <hr/>
6        <form action = "{{ request.path }}" method = "post" class= "container">
7           <label for = "email">Correo Electrónico</label><br>
8           <input type = "text" name = "email" placeholder = "email" /><br>
9           <label for = "password">Ingrese su contraseña</label><br>
10          <input type = "password" name = "password" placeholder = "password"/><br>
11
12          <br>
13          <input type = "submit" value = "Buscar" />
14
15       </form>
16 {% endblock %}
```

Ruta ingresar_comentario

```
45  @app.route('/ingresar_comentario', methods = ['GET', 'POST'])
46  def ingresar_comentario():
47      if request.method == 'POST':
48          if not request.form['contenido']:
49              return render_template('error.html', error="Contenido no ingresado...")
50          else:
51              nuevo_comentario= Comentario(fecha=datetime.now(), contenido=request.form['contenido'], usuario_id =request.form['userId'])
52              db.session.add(nuevo_comentario)
53              db.session.commit()
54              return render_template('inicio.html')
55      return render_template('inicio.html')
```

ingresar_comentario.html

```
1   {% extends "base_template.html" %}
2   {% block title %}Nuevo Comentario{% endblock %}
3   {% block content %}
4   <body>
5         <h1>Nuevo Comentario</h1>
6         <hr/>
7         <form action = "{{ url_for('ingresar_comentario') }}" method = "post" class = "container">
8            <label for = "contenido">{{ usuario.nombre}} escribe tu comentario</label><br>
9            <textarea name = "contenido" placeholder = "Comentario" cols="60" rows="8"></textarea><br><br>
10          <input id="userId" name="userId" type="hidden" value="{{usuario.id}}">
11
12           <br>
13           <input type = "submit" value = "Enviar" />
14        </form>
15     </body>
16  {% endblock %}
```

Ruta listar_comentarios

```python
@app.route('/listar_comentarios')
def listar_comentarios():
    return render_template('listar_comentario.html', comentarios = Comentario.query.all())
```

listar_comentarios.html

```html
{% extends "base_template.html" %}
{% block title %}Comentarios{% endblock %}
{% block content %}
 <h1>Comentarios registrados </h1>
      <hr/>
      <div class="container">
      <table>
         <thead>
            <tr>
               <th>Usuario</th>
               <th>Comentario</th>
               <th>Fecha y hora</th>
            </tr>
         </thead>

         <tbody>
            {% for comentario in comentarios %}
               <tr>
                  <td>{{ comentario.usuario.nombre }}</td>
                  <td>{{ comentario.contenido }}</td>
                  <td>{{ comentario.fecha }}</td>
               </tr>
            {% endfor %}
         </tbody>
      </table>
      <br><br>
      </div>
{% endblock %}
```

Ruta listar_comentarios_usuario

```
61  @app.route('/listar_comentarios_usuario', methods = ['GET', 'POST'])
62  def listar_comentarios_usuario():
63      if request.method == 'POST':
64          if not request.form['usuarios']:
65              #Pasa como parámetro todos los usuarios
66              return render_template('listar_comentario_usuario.html', usuarios = Usuario.query.all(), usuario_seleccionado = None )
67          else:
68              return render_template('listar_comentario_usuario.html', usuarios= None, usuario_selec = Usuario.query.get(request.form['usuarios']))
69      else:
70          return render_template('listar_comentario_usuario.html', usuarios = Usuario.query.all(), usuario_selec = None )
71
```

listar_comentarios_usuario.html

```
1   {% extends "base_template.html" %}
2   {% block title %}Comentario de Usuario{% endblock %}
3   {% block content %}
4   <body>
5         <h1>Comentarios registrados </h1>
6         <hr/>
7         <form action="{{ request.path }}" method="POST" class= "container">
8
9           {% if usuarios is not none :%}
10              <label for = "usuarios">Usuarios</label> <br>
11              <select id= "usuarios" name="usuarios" width="500px" >
12              {% for usuario in usuarios %}
13                  <option value="{{ usuario.id }}">{{ usuario.nombre}}</option>
14              {% endfor %}
15              </select>
16
17              <br><br>
18
19              <input class="button1" type="submit" value="Buscar comentarios">
20          {% else %}
21              <p> {{ usuario_selec.nombre}} ({{ usuario_selec.correo}}) escribió los siguientes comentarios:</p>
22               {% for comentario in usuario_selec.comentario %}
23                 <ul>
24                     <li>{{ comentario.contenido }} </li>
25                 </ul>
26              {% endfor %}
27          {% endif %}
28        <br><br>
29        </form>
30
31  {% endblock %}
```

FIN