

PROGRAMACIÓN ORIENTADA A OBJETOS

Departamento de Informática
FCEFNY – UNSJ

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

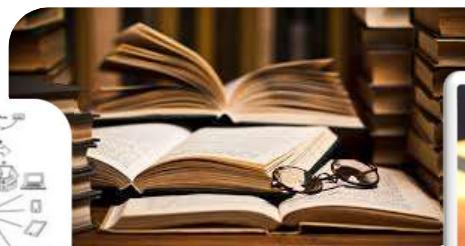
Año 2021

Unidad 5

**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/>



Flask

web development,
one drop at a time

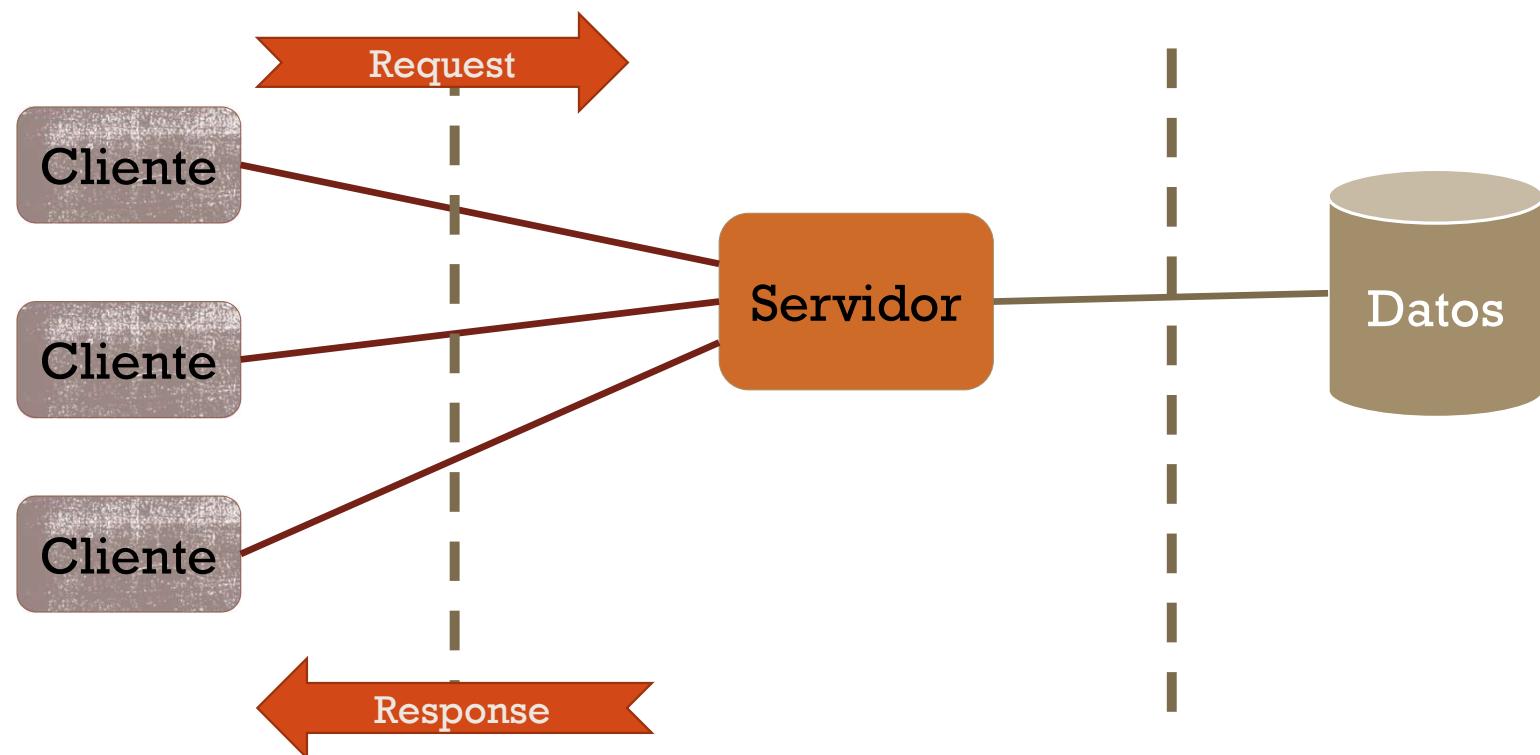


SQLAlchemy

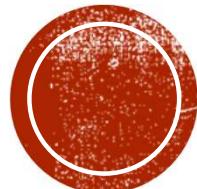
Un sistema basado en la web es una colección de páginas web que se vinculan entre sí y que contienen distintos recursos (texto, imágenes, videos y otros) para brindar un servicio a los usuarios.

ESTRUCTURA DE UNA APLICACIÓN WEB

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

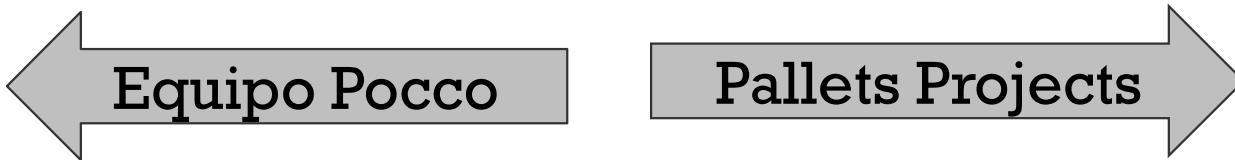


Aplicaciones Web con Python



Framework Flask

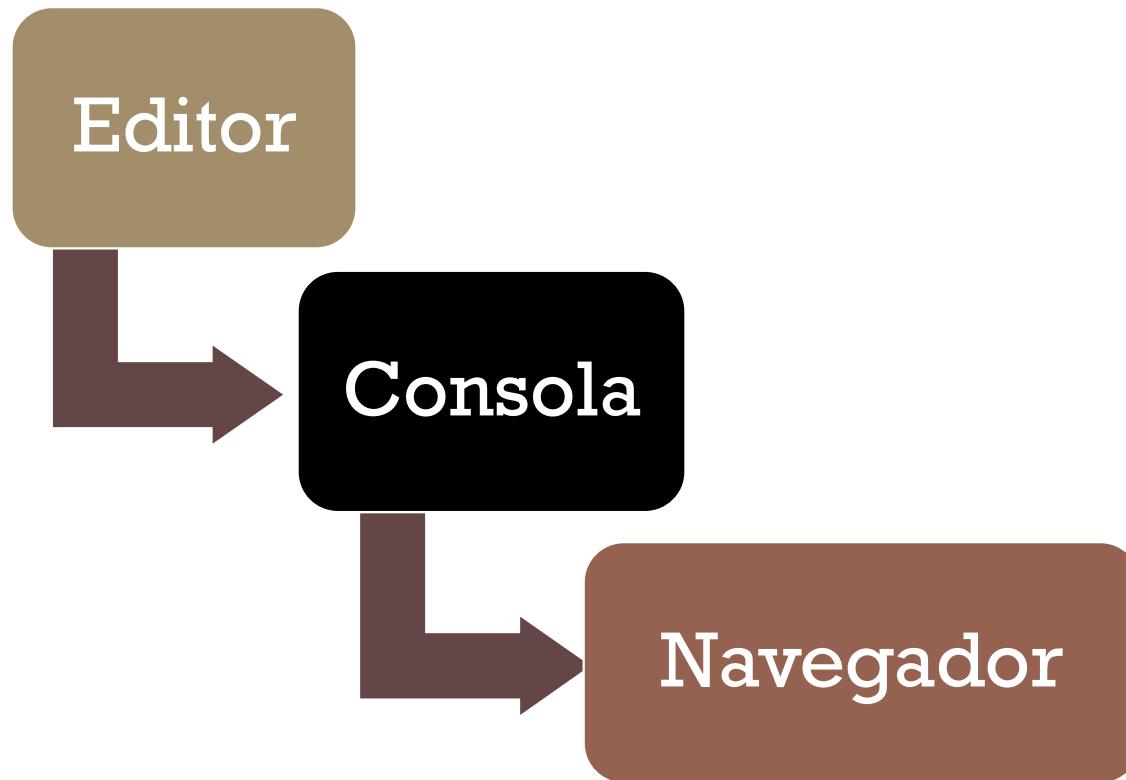
Armin Ronacher



\$ pip install Flask

Librerías como Werkzeug y Jinja2

EN EL DESARROLLO DE UNA APLICACIÓN FLASK SE UTILIZAN DISTINTOS ENTORNOS



Editor

```
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

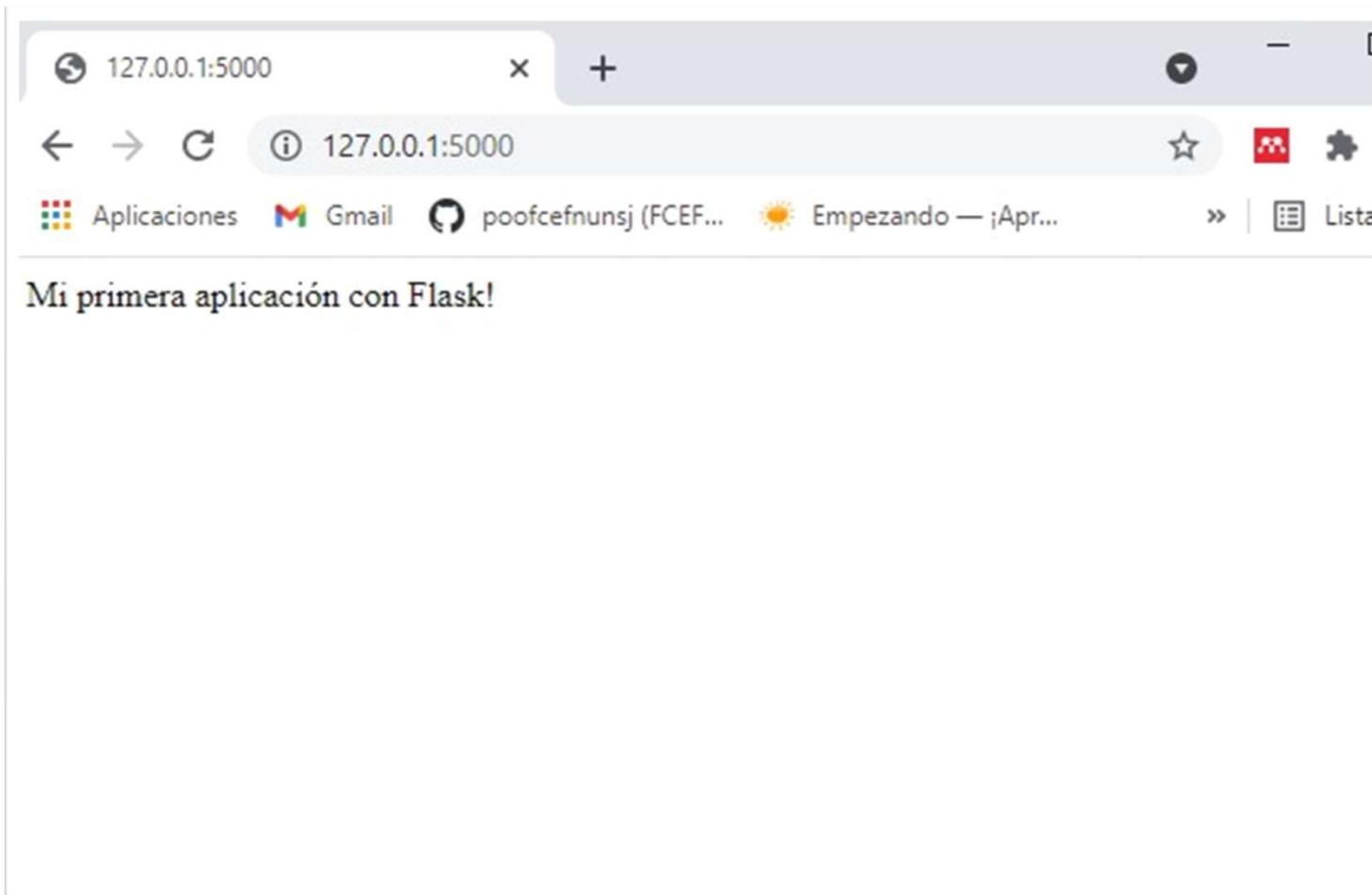
Consola

```
■ 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

Aplicación web



Conjunto de páginas web



Cada página tiene una dirección



Decorador app.route

```

from flask import Flask

app = Flask(__name__)

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

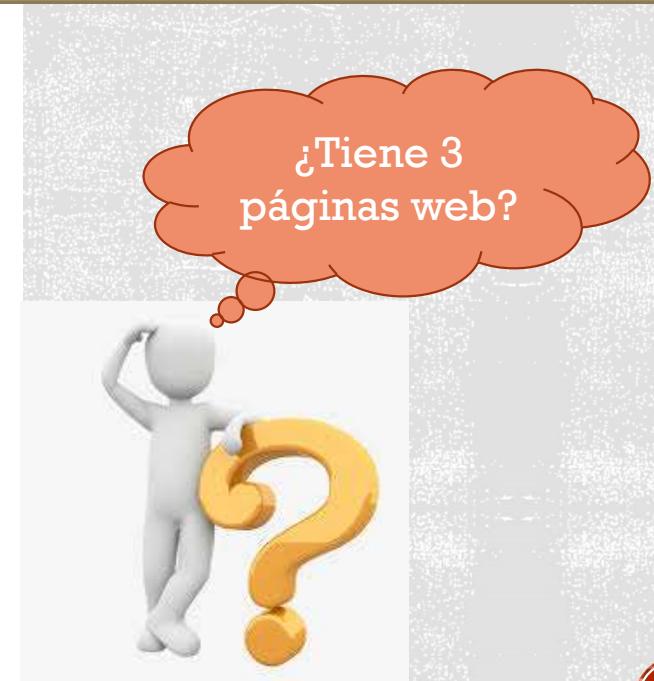
@app.route('/saludo_bienvenida')
def saludoBienvenida():
    return 'Bienvenido a mi aplicación Flask!'

@app.route('/saludo_despedida')
def saludoDespedida():
    return 'Gracias por visitar mi aplicación Flask!'

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

```

- <http://127.0.0.1:5000>
- http://127.0.0.1:5000/saludo_bienvenida
- http://127.0.0.1:5000/saludo_despedida



HTML

HyperText Markup Language

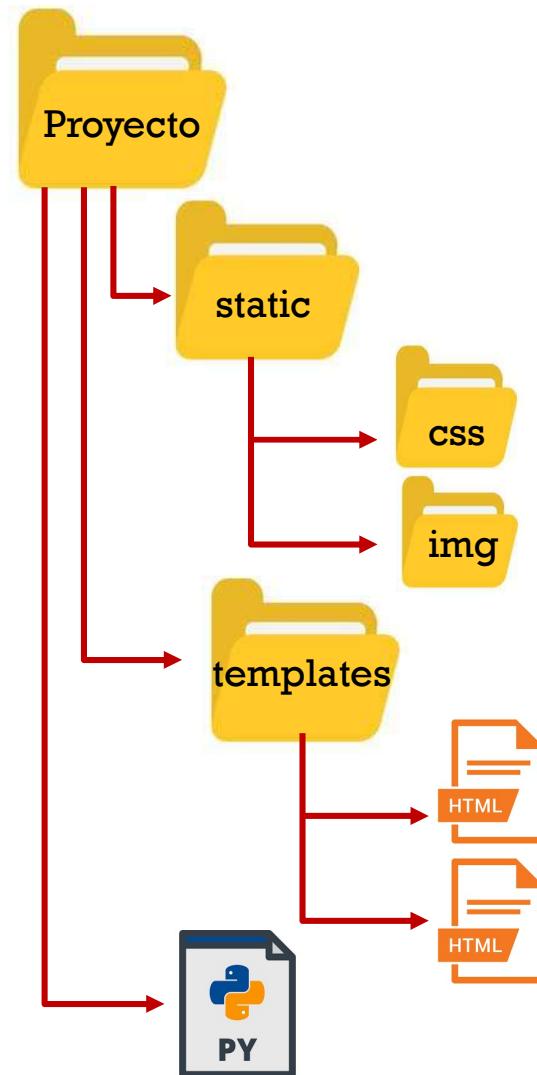
14

```
<!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

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

LA URL PUEDE CONTENER VARIABLES

Rutas dinámicas

```
@app.route('/user/<username>')
def perfil(username):
    ....
    ....
```

```
@app.route('/poo/<int:ciclo>/<int:unidad>/<titulo>')
def unidad(ciclo, unidad, titulo):
    ....
    ....
```

Las variables de la ruta se pasan a la función como argumento.

FORMULARIOS

Campus Virtual - Universidad NACIONAL de San Juan

campusvirtual.unsj.edu.ar/login/

Aplicaciones Gmail poofcefunsj (FCEF... Empezando — ;Apr...

Acceder

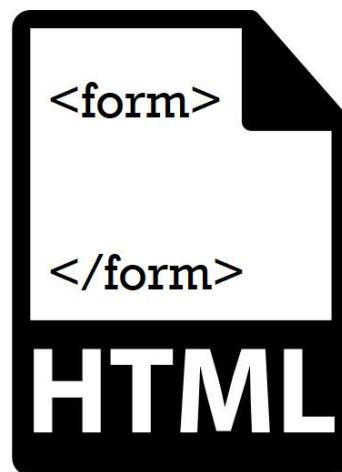
Nombre de usuario

Contraseña

Recordar nombre de usuario

Iniciar Sesión

Registrarse



Área personal

campusvirtual.unsj.edu.ar/my/

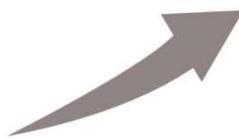
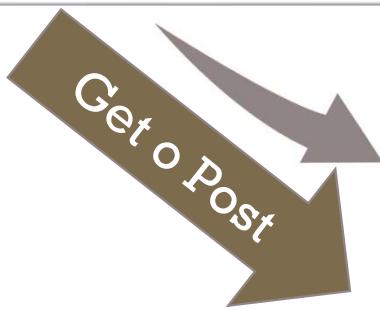
Aplicaciones Gmail poofcefunsj (FCEF... Empezando — ;Apr...

Campus Virtual de la UNSJ

Sistema Institucional de Educación a Distancia
Universidad Nacional de San Juan

Área personal

Personalizar esta página



FORMULARIOS

```
<!DOCTYPE html>
<html>
<body>
<h3>Nuevo Usuario</h3>

<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>
```

The screenshot shows a web browser window with the URL 127.0.0.1:5000. The page title is "Nuevo Usuario". There are three input fields: "Nombre" with the value "Juan", "Correo Electrónico" with the value "juancito@gmail.com", and "Contraseña" with a redacted value. A "Guardar" button is at the bottom.

OBJETO REQUEST

Nuevo Usuario

Nombre
Juan

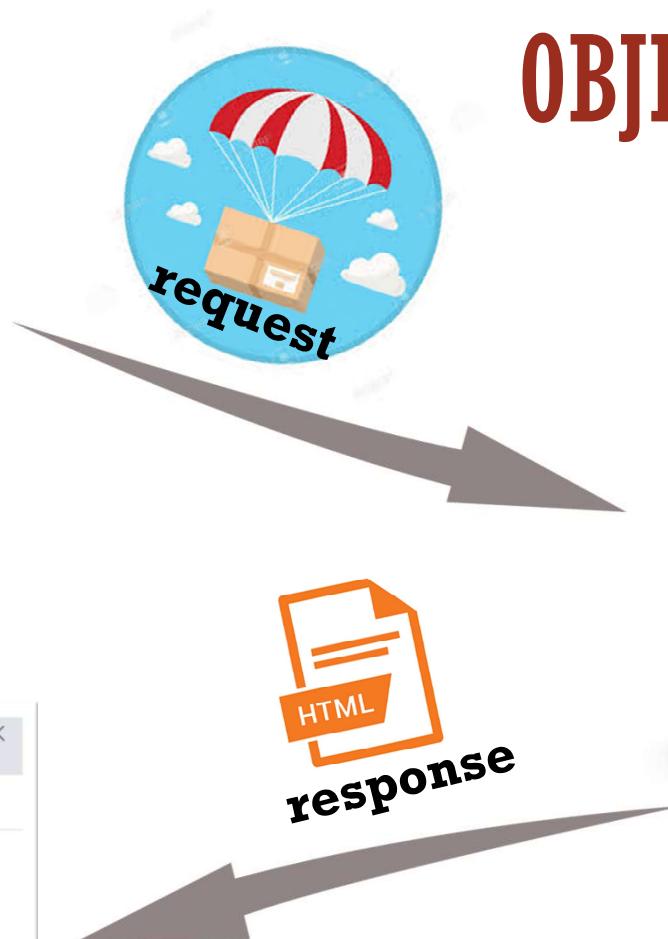
Correo Electrónico
juancito@gmail.com

Contraseña
.....

Guardar

Buenas tardes Juan

Recibirás las notificaciones en el correo
juancito@gmail.com



```
from flask import request  
  
if request.method == 'POST':  
  
    request.form['nombre_componente']
```

EL OBJETO
REQUEST
CONTIENE
INFORMACIÓN
DEL
FORMULARIO

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.

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)
```

nuevo_usuario.html

```
<!DOCTYPE html>
<html>
  <body>
    <h3>Nuevo Usuario</h3>
    <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>
```

Nuevo Usuario

Nombre
Juan

Correo Electrónico
juancito@gmail.com

Contraseña

Guardar

```
<!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>
```

bienvenida.html

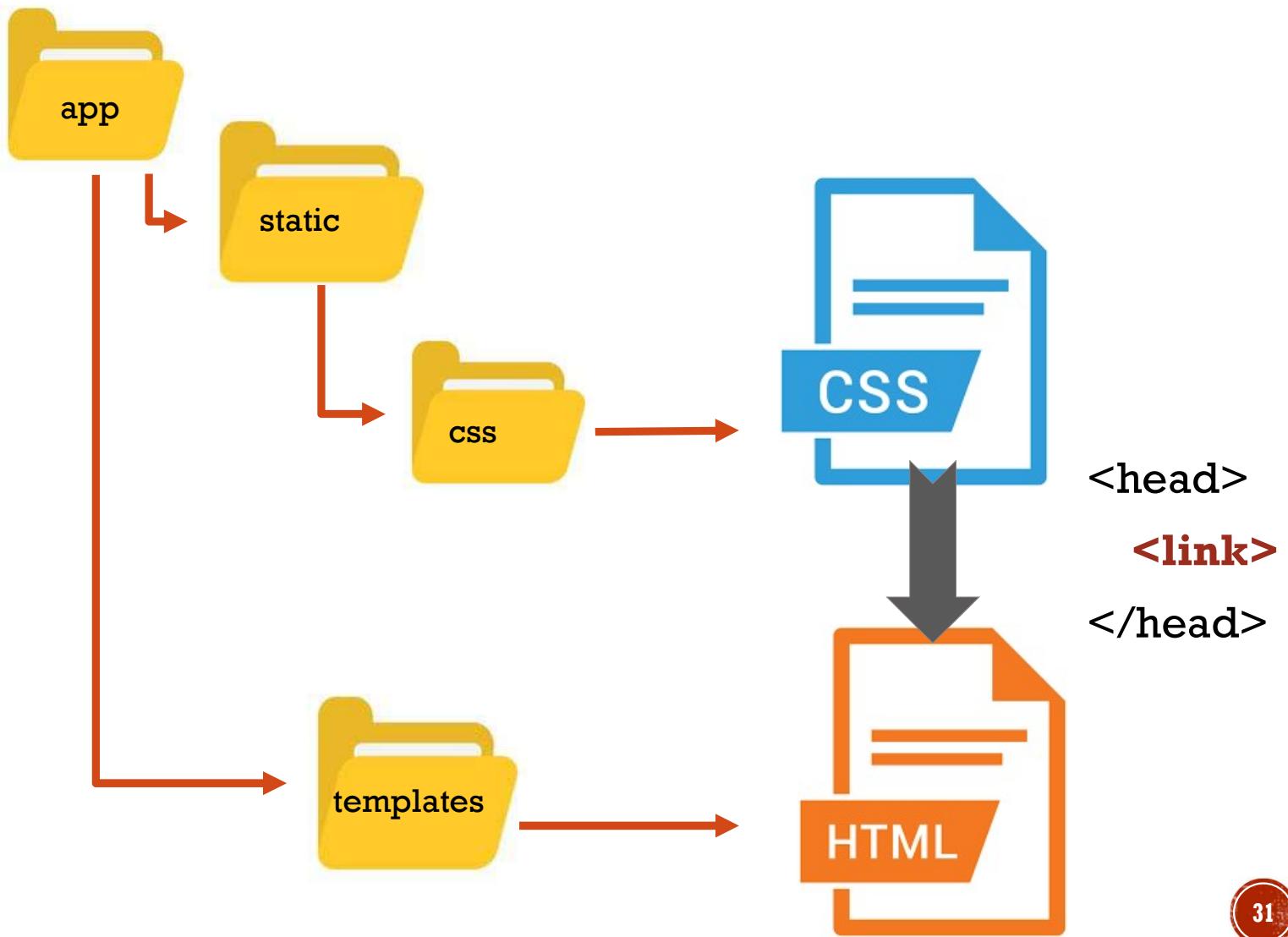






Cascading Style Sheet

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>
<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

The screenshot shows a web browser window titled "Crear Usuario". The address bar displays the URL "127.0.0.1:5000". The main content area is titled "Nuevo Usuario" and contains a form for creating a new user. The form fields are:

- Nombre**: A text input field containing "Nombre".
- Correo Electrónico**: A text input field containing "email".
- Contraseña**: A text input field containing "password".
- Guardar**: A button labeled "Guardar" at the bottom of the form.

PLANTILLA BASE

35

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

Jinja2 provee la etiqueta

{% block %}

...

{% endblock %}

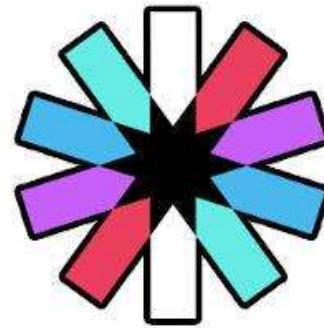
```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <title>{% block title %}{% endblock %}</title>
6      <link rel="stylesheet" href="{{ url_for("static", filename="css/estilos.css") }}>
7  </head>
8  <body>
9      {% block content %}{% endblock %}
10     <div class="myDiv">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

```
1  {% extends "base_template.html" %}  
2  {% block title %}Crear Usuario{% endblock %}  
3  {% block content %}  
4      <h1>Registro de Usuario</h1>  
5      <hr/>  
6      <form action = "{{ request.path }}" method = "post" class = "container">  
7          <label for = "nombre">Nombre</label><br>  
8          <input type = "text" name = "nombre" placeholder = "Nombre" /><br>  
9  
10         <label for = "email">Correo Electrónico</label><br>  
11         <input type = "text" name = "email" placeholder = "email" /><br>  
12  
13         <label for = "password">Contraseña</label><br>  
14         <input type = "password" name = "password" placeholder = "password"/><br>  
15         <br>  
16         <input type = "submit" value = "Guardar" />  
17  
18     </form>  
19  {% endblock %}
```

nuevo_usuario.html

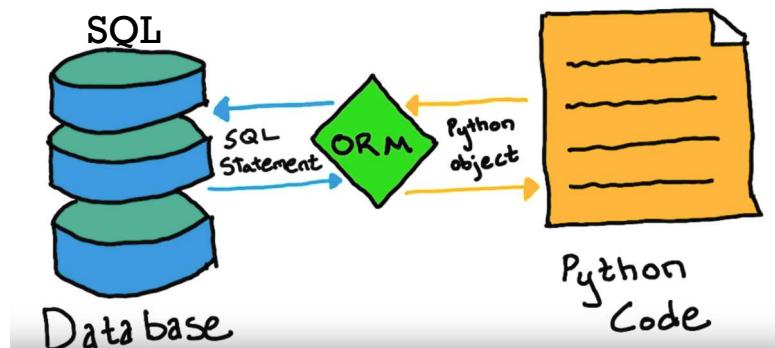


Flask



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



Juan : Persona
-String nombre = "Juan";
-String apellido = "Pérez";
-int edad = 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.

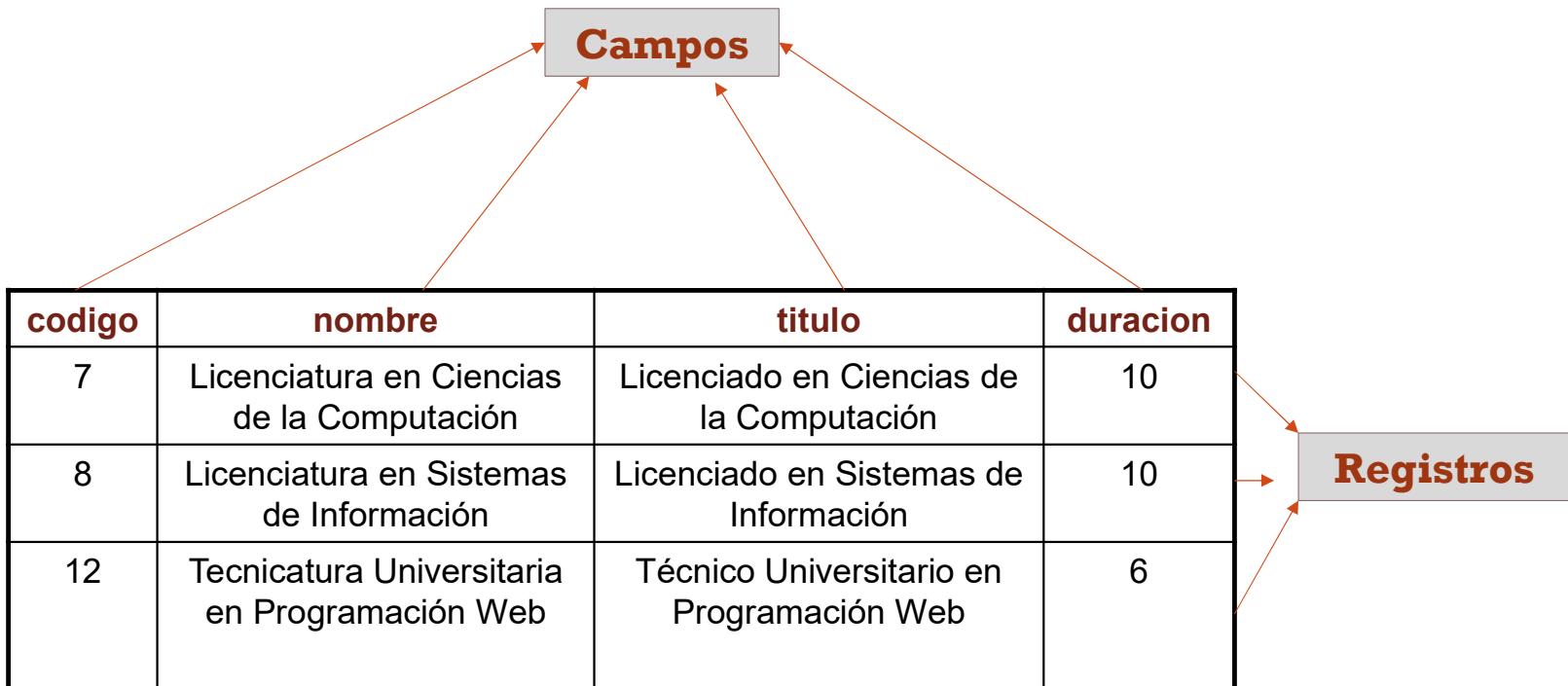


Tabla “Carreras”



- 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

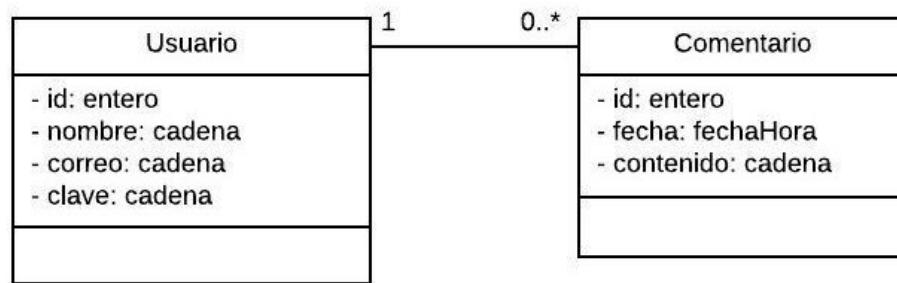
- PostgreSQL
- MySQL
- Oracle
- Microsoft SQL Server
- SQLite



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

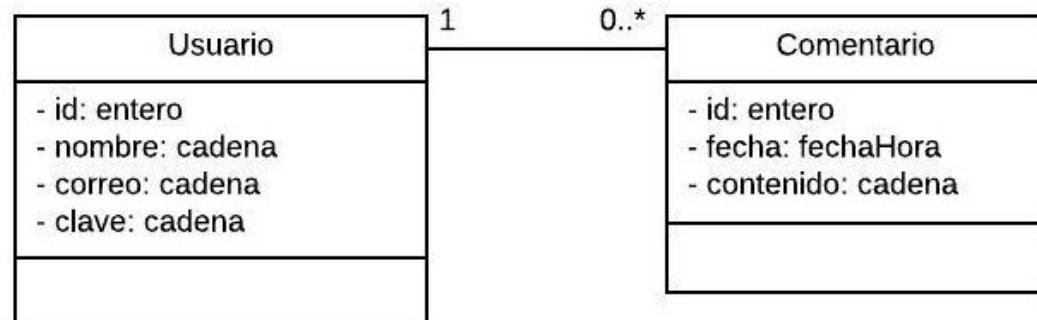
config.py

```
SECRET_KEY = "GDtfDCFYjD"  
  
SQLALCHEMY_DATABASE_URI = 'sqlite:///datos.sqlite3'  
  
SQLALCHEMY_TRACK_MODIFICATIONS = False
```

DEFINIR LOS MODELOS

models.py

```
1  from __main__ import app
2  from flask_sqlalchemy import SQLAlchemy
3
4  db = SQLAlchemy(app)
5
6  class Usuario(db.Model):
7      id = db.Column(db.Integer, primary_key=True)
8      nombre = db.Column(db.String(80), nullable=False)
9      correo = db.Column(db.String(120), unique=True, nullable=False)
10     clave = db.Column(db.String(120), nullable=False)
11     comentario = db.relationship('Comentario', backref='usuario', cascade="all, delete-orphan", lazy='dynamic')
12
13 class Comentario(db.Model):
14     id = db.Column(db.Integer, primary_key=True)
15     fecha = db.Column(db.DateTime)
16     contenido = db.Column(db.Text)
17     usuario_id = db.Column(db.Integer, db.ForeignKey('usuario.id'))
```



LA APLICACION - 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 APLICACION - 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():
17
18     @app.route('/nuevo_comentario', methods = ['GET', 'POST'])
19     def nuevo_comentario():
20
21     @app.route('/ingresar_comentario', methods = ['GET', 'POST'])
22     def ingresar_comentario():
23
24     @app.route('/listar_comentarios')
25     def listar_comentarios():
26
27     @app.route('/listar_comentarios_usuario', methods = ['GET', 'POST'])
28     def listar_comentarios_usuario():
29
```

LA APLICACION - III

app.py

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

LA APLICACION

```
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
12 @app.route('/')
13 def inicio():
14     return render_template('inicio.html')
15 @app.route('/nuevo_usuario', methods = ['GET', 'POST'])
16 def nuevo_usuario():
17
18     @app.route('/nuevo_comentario', methods = ['GET', 'POST'])
19     def nuevo_comentario():
20
21         @app.route('/ingresar_comentario', methods = ['GET', 'POST'])
22         def ingresar_comentario():
23
24             @app.route('/listar_comentarios')
25             def listar_comentarios():
26
27                 @app.route('/listar_comentarios_usuario', methods = ['GET', 'POST'])
28                 def listar_comentarios_usuario():
29
30                     if __name__ == '__main__':
31                         db.create_all()
32                         app.run(debug = True)
```

inicio.html

```
1  <!DOCTYPE html>
2  └<html lang = "en">
3    └<head>
4      <title>Aplicación</title>
5      <link rel="stylesheet" href="{{ url_for('static', filename='css/estilos.css') }}">
6    </head>
7    └<body>
8      <h1> Comentarios</h1>
9      └<ul>
10        <li> <a href = "{{ url_for('nuevo_usuario') }}"> Registrarse como Usuario </a></li><br>
11        <li> <a href = "{{ url_for('nuevo_comentario') }}"> Comentar</a></li><br>
12        <li> <a href = "{{ url_for('listar_comentarios') }}"> Listar Todos los Comentarios </a></li> <br>
13        <li> <a href = "{{ url_for('listar_comentarios_usuario') }}"> Listar Comentarios </a></li><br>
14      </ul>
15    </body>
16  </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

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

listar_comentarios.html

```
1  {% extends "base_template.html" %} 
2  {% block title %}Comentarios{% endblock %} 
3  {% block content %} 
4      <h1>Comentarios registrados </h1> 
5      <hr/> 
6      <div class="container"> 
7          <table> 
8              <thead> 
9                  <tr> 
10                     <th>Usuario</th> 
11                     <th>Comentario</th> 
12                     <th>Fecha y hora</th> 
13                 </tr> 
14             </thead> 
15 
16             <tbody> 
17                 {% for comentario in comentarios %} 
18                     <tr> 
19                         <td>{{ comentario.usuario.nombre }}</td> 
20                         <td>{{ comentario.contenido }}</td> 
21                         <td>{{ comentario.fecha }}</td> 
22                     </tr> 
23                 {% endfor %} 
24             </tbody> 
25         </table> 
26         <br><br> 
27     </div> 
28 {% 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 
21         {% else %} 
22             <p> {{ usuario_selec.nombre}} ({{ usuario_selec.correo}}) escribió los siguientes comentarios:</p> 
23             {% for comentario in usuario_selec.comentario %} 
24                 <ul> 
25                     <li>{{ comentario.contenido }} </li> 
26                 </ul> 
27                 {% endfor %} 
28             {% endif %} 
29             <br><br> 
30         </form> 
31 
31  {% endblock %}
```

Ejemplo de ruta dinámica

Una aplicación solo permite el acceso a sus usuarios registrados. De cada uno de ellos se registra el nombre, correo, contraseña e idioma preferido.

Cuando un usuario inicia sesión, el saludo inicial se emite muestra en el idioma de preferencia del usuario.

app.py

```
@app.route('/ingresar', methods = ['GET','POST'])
def ingresar():
    if request.method == 'POST':
        if not request.form['email'] or not request.form['password']:
            return render_template('error.html', error="Por favor ingrese los datos requeridos")
        else:
            usuario_actual= Usuario.query.filter_by(correo= request.form['email']).first()
            if usuario_actual is None:
                return render_template('error.html', error="El correo no está registrado")
            else:
                verificacion = check_password_hash(usuario_actual.clave, request.form['password'])
                if (verificacion):
                    return redirect(url_for('bienvenida',leng = usuario_actual.lenguaje))
                else:
                    return render_template('error.html', error="La contraseña no es válida")
            else:
                return render_template('ingresar.html')
```

app.py

```
@app.route('/bienvenida/<leng>')
def bienvenida(leng):
    if leng == 'es':
        return render_template('bienvenida.html', saludo='Hola!')
    else:
        return render_template('bienvenida.html', saludo='Hello')
```

Bienvenida.html

```
{% extends "base_template.html" %}  
{% block title %}Bienvenida{% endblock %}  
{% block content %}  
  
    <h1> {{ saludo }} </h1>  
  
{% endblock %}
```



FIN