PROYECTO DE SOFTWARE

Cursada 2021

TEMARIO

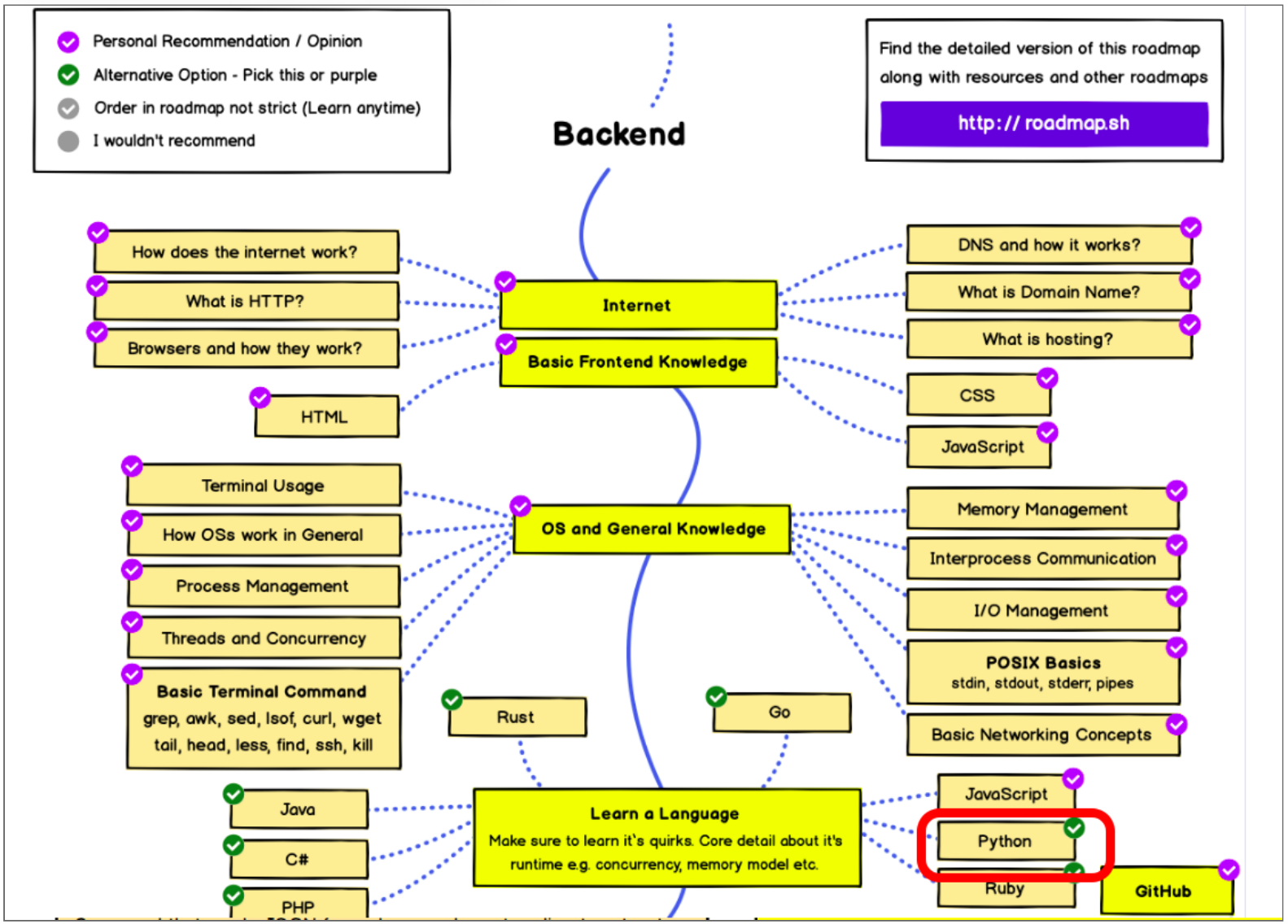
- Repaso Python.
- Flask
- MVC

REPASO PYTHON

REPASANDO

- ¿Por qué usamos y promovemos el uso de software libre?
- ¿Características de Python?
- ¿Desarrollo en el servidor?

PROGRAMACIÓN EN EL SERVIDOR



NOSOTROS USAREMOS PYTHON

¿Por qué?

- https://www.tiobe.com/tiobe-index/
- https://insights.stackoverflow.com/trends?tags=python
- https://githut.info/
- https://madnight.github.io/githut/#/pull_requests/2021/2

PYTHON

- Fue desarrollado por **Guido Van Rossum** a finales de los años 80.
- El nombre proviene del grupo de cómicos ingleses **Monty Python**.
- Sitio Oficial: **https://python.org/**
- Python Argentina: **https://python.org.ar/**

CARACTERÍSTICAS GENERALES

- Es un lenguaje de alto nivel, fácil de aprender. Muy expresivo y legible.
- Es interpretado.
- Multiplataforma y multiparadigma.
- Posee una sintáxis elegante y tipado dinámico y fuerte.
- Tiene un manejo eficiente de estructuras de datos de alto nivel.

SINTAXIS BÁSICA

- **NO** hay declaración de variables: las variables se crean **dinámicamente** cuando se les asigna un valor.
- Tiene **tipado dinámico**: las variables pueden cambiar de tipo a lo largo del programa.
- Bloques, identación.

FRAMEWORK

- Esquema de trabajo que nos permite:
- Evitar la escritura de código de manera repetitiva
- Buenas prácticas
- Estructura del código relativamente homogeneo.

FLASK

- Instalamos un nuevo paquete al entorno, Flask.

```
(entornoProyecto) diego@diego-personal:~/Documentos/Git/miProyecto$ pip install flask
Collecting flask
  Downloading https://files.pythonhosted.org/packages/9b/93/628509b8d5dc749656a9641f4caf13540e2cdec85276964ff8f43bbb1d3b/Flask-1.1.1-py2.py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 857kB/s
Collecting click>=5.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl (81kB)
    100% |████████████████████████████████| 81kB 2.2MB/s
Collecting Werkzeug>=0.15 (from flask)
  Downloading https://files.pythonhosted.org/packages/ce/42/3aeda98f96e85fd26180534d36570e4d18108d62ae36f87694b476b83d6f/Werkzeug-0.16.0-py2.py3-none-any.whl (327kB)
    100% |████████████████████████████████| 327kB 934kB/s
Collecting itsdangerous>=0.24 (from flask)
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting Jinja2>=2.10.1 (from flask)
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/1d/e7/fd8b501e7a6dfe492a433deb7b9d833d39ca74916fa8bc63dd1a4947a671/Jinja2-2.10.1-py2.py3-none-any.whl (124kB)
    100% |████████████████████████████████| 133kB 1.6MB/s
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->flask)
  Cache entry deserialization failed, entry ignored
  Using cached https://files.pythonhosted.org/packages/fb/40/f3adb7cf24a8012813c5edb20329eb22d5d8e2a0ecf73d21d6b85865da11/MarkupSafe-1.1.1-cp27-cp27mu-manylinux1_x86_64.whl
Installing collected packages: click, Werkzeug, itsdangerous, MarkupSafe, Jinja2, flask
Successfully installed Jinja2-2.10.1 MarkupSafe-1.1.1 Werkzeug-0.16.0 click-7.0 flask-1.1.1 itsdangerous-1.1.0
(entornoProyecto) diego@diego-personal:~/Documentos/Git/miProyecto$
```

FLASK

- Hola Mundo

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

FLASK

- Miramos nuestro código en el browser:

```
export FLASK_APP=index.py
flask run
```

```
(entornoProyecto) diego@diego-personal:~/Documentos/Git/miProyecto$ flask run
 * Serving Flask app "index.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   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 - - [22/Sep/2019 19:23:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [22/Sep/2019 19:23:35] "GET /favicon.ico HTTP/1.1" 404 -
```

FLASK

- Modificamos nuestro código:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    mensaje = 'Hello, World!'
    return mensaje + 2
```

Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

FLASK

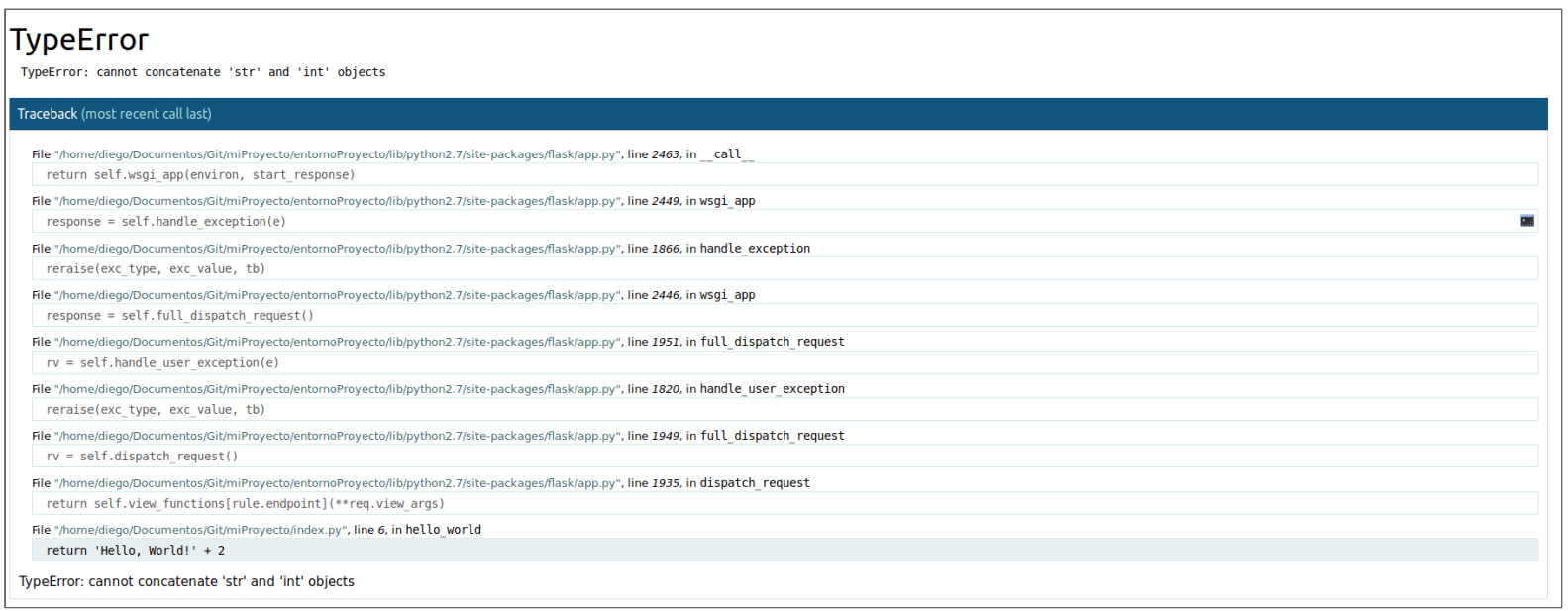
- Activamos el entorno de desarrollo:

```
export FLASK_ENV=development
```

```
^C(entornoProyecto) diego@diego-personal:~/Documentos/Git/miProyecto$ export FLASK_ENV=development
(entornoProyecto) diego@diego-personal:~/Documentos/Git/miProyecto$ flask run
 * Serving Flask app "index.py" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 177-792-228
```

FLASK

- Ahora podemos ver un poco más:



TypeError

TypeError: cannot concatenate 'str' and 'int' objects

Traceback (most recent call last)

File "/home/diego/Documentos/Git/miProyecto/entornoProyecto/lib/python2.7/site-packages/flask/app.py", line *2463*, in __call__

```
return self.wsgi_app(environ, start_response)
```

File "/home/diego/Documentos/Git/miProyecto/entornoProyecto/lib/python2.7/site-packages/flask/app.py", line *2449*, in wsgi_app

```
response = self.handle_exception(e)
```

File "/home/diego/Documentos/Git/miProyecto/entornoProyecto/lib/python2.7/site-packages/flask/app.py", line *1866*, in handle_exception

```
reraise(exc_type, exc_value, tb)
```

File "/home/diego/Documentos/Git/miProyecto/entornoProyecto/lib/python2.7/site-packages/flask/app.py", line *2446*, in wsgi_app

```
response = self.full_dispatch_request()
```

File "/home/diego/Documentos/Git/miProyecto/entornoProyecto/lib/python2.7/site-packages/flask/app.py", line *1951*, in full_dispatch_request

```
rv = self.handle_user_exception(e)
```

File "/home/diego/Documentos/Git/miProyecto/entornoProyecto/lib/python2.7/site-packages/flask/app.py", line *1820*, in handle_user_exception

```
reraise(exc_type, exc_value, tb)
```

File "/home/diego/Documentos/Git/miProyecto/entornoProyecto/lib/python2.7/site-packages/flask/app.py", line *1949*, in full_dispatch_request

```
rv = self.dispatch_request()
```

File "/home/diego/Documentos/Git/miProyecto/entornoProyecto/lib/python2.7/site-packages/flask/app.py", line *1935*, in dispatch_request

```
return self.view_functions[rule.endpoint](**req.view_args)
```

File "/home/diego/Documentos/Git/miProyecto/index.py", line *6*, in hello_world

```
return 'Hello, World!' + 2
```

TypeError: cannot concatenate 'str' and 'int' objects

FLASK

- Vamos recibir argumentos

```
from flask import Flask

app = Flask(__name__)


@app.route("/")
def index():
    return "Hola mundo"

@app.route("/personas")
def personas():
    return "personas"

@app.route("/personas/<string:nombre>")
def persona(nombre):
    nombre = nombre.capitalize()
    return f"Hola {nombre}"
```

FLASK

- Armamos un html:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hola_mundo():
    mensaje = '<!DOCTYPE html>'
    mensaje +=  '<html lang="es"> '
    mensaje +=  '<head>'
    mensaje += '</head>'
    mensaje += '<body>'
    mensaje += 'Hola mundo!'
    mensaje += '</body>'
    mensaje += '</html>'
    return mensaje
```

FLASK

- y otro...:

```
from flask import Flask
app = Flask(__name__)


@app.route("/personas")
def personas():
    mensaje = '<!DOCTYPE html>'
    mensaje +=  '<html lang="es"> '
    mensaje +=  '<head>'
    mensaje += '</head>'
    mensaje += '<body>'
    mensaje += 'Hola personas!'
    mensaje += '</body>'
    mensaje += '</html>'
    return mensaje
```

FLASK

- y otro...:

```
from flask import Flask
app = Flask(__name__)


@app.route("/personas/<string:nombre>")
def persona(nombre):
    nombre = nombre.capitalize()
    mensaje = '<!DOCTYPE html>'
    mensaje +=  '<html lang="es"> '
    mensaje +=  '<head>'
    mensaje += '</head>'
    mensaje += '<body>'
    mensaje += f"Hola {nombre}"
    mensaje += '</body>'
    mensaje += '</html>'
    return mensaje
```

FLASK

¿Problemas? ¿Inconvenientes?

FLASK

- Una solución

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def hola_mundo():
    return render_template ("index.html")
```

FLASK

- index.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo Proyecto</title>
    </head>
    <body>
        <h1>Hola mundo!<h1>
    </body>
</html>
```

FLASK

- Con variables

```
@app.route("/personas")
def personas():
    contenido = "Hola personas!"
    return render_template("index1.html",
contenido=contenido)

@app.route("/personas/<string:nombre>")
def persona(nombre):
    nombre = nombre.capitalize()
    contenido = f"Hola {nombre}"
    return render_template("index1.html",
contenido=contenido)
```

FLASK

- index.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo Proyecto</title>
    </head>
    <body>
        <h1>{{contenido}}<h1>
    </body>
</html>
```

FLASK

- Condicionales

```
@app.route("/")
def hola_mundo():
    return render_template ("index.html", contenido="mundo")


@app.route("/personas")
def personas():
    contenido = "personas!"
    return render_template("index.html", contenido=contenido)

@app.route("/personas/<string:nombre>")
def persona(nombre):
    nombre = nombre.capitalize()
    return render_template("index.html", contenido=nombre)
```

FLASK

- index.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo Proyecto</title>
    </head>
    <body>
            {% if contenido == "River" %}
                <h1>El campeón es {{contenido}}<h1>
            {% else %}
                <h1>Hola {{contenido}}<h1>
            {% endif %}
    </body>
</html>
```

FLASK

- Iteraciones

```
from flask import Flask, render_template

app = Flask(__name__)


pos_equipos = {"River" : "Campeón",
               "Boca": "Sub Campeón",
               "Gremio": "Semi finalista",
               "Palmeiras": "Semi finalista"}

@app.route("/equipos")
def personas():
    return render_template("index.html",
contenido=pos_equipos)
```

FLASK

- index.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo Proyecto</title>
    </head>
    <body>
            <ul>
               {% for equipo in contenido %}
                    <li>{{equipo}}</li>
               {% endfor %}
            </ul>
    </body>
</html>
```

FLASK

- Url

```
app = Flask(__name__)


pos_equipos = {"River" : "Campeón",
               "Boca": "Sub Campeón",
               "Gremio": "Semi finalista",
               "Palmeiras": "Semi finalista"}

@app.route("/")
def index():
    return render_template ("index.html", contenido="mundo")


@app.route("/equipos")
def equipos():
    return render_template("equipos.html",
contenido=pos_equipos)
```

FLASK

- index.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo Proyecto</title>
    </head>
    <body>
       <h1> Libertadores 2018 </h1>
       <a href="{{url_for('equipos')}}"> Equipos </a>
    </body>
</html>
```

FLASK

- equipos.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo Proyecto</title>
    </head>
    <body>
            <ul>
               {% for equipo in contenido %}
                    <li>{{equipo}}</li>
               {% endfor %}
            </ul>
            <a href="{{url_for('index')}}"> Volver al inicio
</a>
    </body>
</html>
```

FLASK

- Herencia
- layout.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo Proyecto</title>
    </head>
    <body>
       <h1> {% block heading %} {% endblock %}</h1>
       {% block contenido %}
       {% endblock %}
    </body>
</html>
```

FLASK

- Herencia
- index.html

```
{% extends "layout.html"%}
{% block heading%}
    <h1> Libertadores 2018 </h1>
{% endblock %}
{% block contenido %}
    <a href="{{url_for('equipos')}}"> Equipos </a>
{% endblock %}
```

FLASK

- Herencia
- equipos.html

```
{% extends "layout.html"%}
{% block heading%}
    <h1> Posiciones 2018 </h1>
{% endblock %}
{% block contenido %}
        <ul>
        {% for equipo in contenido %}
            <li>{{equipo}}</li>
        {% endfor %}
        </ul>
        <a href="{{url_for('index')}}"> Volver al inicio </a>
{% endblock %}
```

FLASK

- Formularios

```
from flask import Flask, render_template, request

app = Flask(__name__)


pos_equipos = {"River" : "Campeón",
               "Boca": "Sub Campeón",
               "Gremio": "Semi finalista",
               "Palmeiras": "Semi finalista"}

@app.route("/equipos")
def equipos():
    return render_template("equipos.html",
contenido=pos_equipos)

@app.route("/agregar", methods=["POST"])
def agregar():
    equipo = request.form.get("equipo")
    posicion = request.form.get("posicion")
    pos_equipos[equipo] = posicion
    return render_template("equipos.html",
contenido=pos_equipos)
```

FLASK

- Formularios
- equipos.html

```
{% extends "layout.html"%}
{% block heading%}
    <h1> Posiciones 2018 </h1>
{% endblock %}
{% block contenido %}
        <ul>
        {% for equipo in contenido %}
            <li>{{equipo}}</li>
        {% endfor %}
        </ul>
        <form action="{{ url_for('agregar')}}" method="post">
        <input type="text" name="equipo" placeholder="Ingrese
un equipo">
        <input type="text" name="posicion"
placeholder="Ingrese la posicion">
        <button>Enviar</button>
        </form>
        <a href="{{url_for('index')}}"> Volver al inicio </a>
{% endblock %}
```

EJEMPLOS

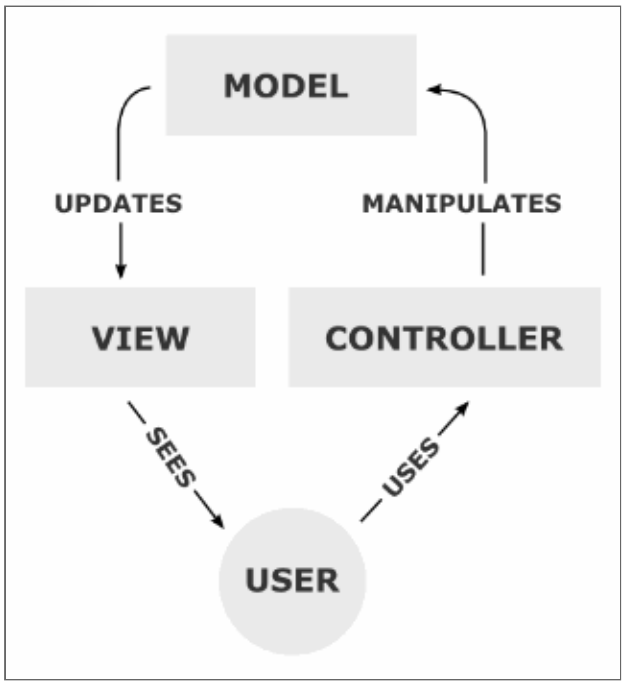
```
https://proyecto-de-
software.github.io/2021/01_teorias/clase4_ejemplos.zip
```

PATRÓN MVC

MODEL – VIEW - CONTROLLER

- Tres componentes:
  - Modelo
  - Vista
  - Controlador
- El principio más importante de la arquitectura MVC es la **separación del código del programa en tres capas**, dependiendo de su naturaleza.
- La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador.

MVC



- Reduce la complejidad, facilita la reutilización y acelera el proceso de comunicación entre capas.

APLICACIÓN TÍPICA SIN MVC

- Aplicación típica que no sigue MVC tiene todo el código en el mismo lugar.

```
#encoding: utf-8
import pymysql

from flask import Flask, g



app = Flask(__name__)

@app.route('/')
def hello_world():
    #CREATE USER 'proyecto'@'localhost' IDENTIFIED BY
'password1';
    #GRANT ALL PRIVILEGES ON *.* TO 'proyecto'@'localhost';
    SECRET_KEY = "dev"
    DEBUG = True
    DB_HOST = 'localhost'
    DB_USER = 'proyecto'
    DB_PASS = 'password1'
    DB_NAME = 'proyecto'
    #Creamos la conexión
    g.db = pymysql.connect(
            host=DB_HOST,
            user=DB_USER,
            password=DB_PASS,
            db=DB_NAME,
```

```
            cursorclass=pymysql.cursors.DictCursor
            )

    mensaje = '<!DOCTYPE html>'
    mensaje +=  '<html lang="en"> '
    mensaje +=  '<head>'
    mensaje += '</head>'
    mensaje += '<body>'

    sql = "SELECT * FROM issues"

    cursor = g.db.cursor()
    cursor.execute(sql)
    issues = cursor.fetchall()
    mensaje += '<table>'

    mensaje += '<tr>'
    for field in issues[0].keys():
        mensaje += '<th>'
        mensaje += field
        mensaje += '</th>'
    mensaje += '</tr>'

    for issue in issues:
        mensaje += '<tr>'
        for field in issue.values():
            mensaje += '<td>'
            mensaje += str(field)
            mensaje += '</td>'
        mensaje += '</tr>'
    mensaje += '</table>'
    mensaje += '</body>'
    mensaje += '</html>'

    #Cerramos conexión a la BBDD
    db = g.pop('db', None)
    if db is not None:
        db.close()
```

```
    return mensaje
```

MVC - CONFIGURACIÓN

app_template/config.py

```
from os import environ

class BaseConfig(object):
    """Base configuration."""

    DEBUG = None
    DB_HOST = "bd_name"
    DB_USER = "db_user"
    DB_PASS = "db_pass"
    DB_NAME = "db_name"
    SECRET_KEY = "secret"

    @staticmethod
    def configure(app):
        # Implement this method to do further configuration
on your app.
        pass


class DevelopmentConfig(BaseConfig):
    """Development configuration."""

    ENV = "development"
    DEBUG = environ.get("DEBUG", True)
    DB_HOST = environ.get("DB_HOST", "localhost")
    DB_USER = environ.get("DB_USER", "MY_DB_USER")
```

```
    DB_PASS = environ.get("DB_PASS", "MY_DB_PASS")
    DB_NAME = environ.get("DB_NAME", "MY_DB_NAME")
```

MVC – SEPARANDO EL MODELO

app_template/app/db.py

```
import pymysql

from flask import current_app
from flask import g
from flask import cli


def connection():
    if "db_conn" not in g:
        conf = current_app.config
        g.db_conn = pymysql.connect(
            host=conf["DB_HOST"],
            user=conf["DB_USER"],
            password=conf["DB_PASS"],
            db=conf["DB_NAME"],
            cursorclass=pymysql.cursors.DictCursor,
        )

    return g.db_conn


def close(e=None):
    conn = g.pop("db_conn", None)

    if conn is not None:
        conn.close()
```

```
def init_app(app):
    app.teardown_appcontext(close)
```

MVC – SEPARANDO EL MODELO

app_template/app/models/issue.py

```
class Issue(object):
    @classmethod
    def all(cls, conn):
        sql = "SELECT * FROM issues"

        cursor = conn.cursor()
        cursor.execute(sql)

        return cursor.fetchall()

    @classmethod
    def create(cls, conn, data):
        sql = """
            INSERT INTO issues (email, description,
category_id, status_id)
            VALUES (%s, %s, %s, %s)
        """

        cursor = conn.cursor()
        cursor.execute(sql, list(data.values()))
        conn.commit()

        return True
```

MVC - EL CONTROLADOR

app_template/app/resources/issue.py

```
from flask import redirect, render_template, request, url_for
from app.db import connection
from app.models.issue import Issue

# Public resources
def index():
    conn = connection()
    issues = Issue.all(conn)

    return render_template("issue/index.html", issues=issues)


def new():
    return render_template("issue/new.html")


def create():
    conn = connection()
    Issue.create(conn, request.form)

    return redirect(url_for("issue_index"))
```

MVC – SEPARANDO LA VISTA

app_template/app/templates/layout.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    {% block head %}
    <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}" />
    <title>{% block title %}{% endblock %}</title>
    {% endblock %}
  </head>
  <body>
    <div id="navbar">
      {% block navbar %}
      {% endblock %}
    </div>
    <div id="content">
      {% with messages = get_flashed_messages() %}
        {% if messages %}
          <ul class=flashes>
          {% for message in messages %}
            <li>{{ message }}</li>
          {% endfor %}
          </ul>
        {% endif %}
      {% endwith %}

      {% block content %}
      {% endblock %}
```

```
    </div>
    <div id="footer">
      {% block footer %}
      {% endblock %}
    </div>
  </body>
</html>
```

MVC – SEPARANDO LA VISTA - VISTA

app_template/app/templates/issue/index.html

```
{% extends "layout.html" %}
{% block title %}Consultas{% endblock %}
{% block head %}
  {{ super() }}
{% endblock %}
{% block content %}
  <h1>Consultas</h1>
  {% for issue in issues %}
    <li>{{ issue.email }} - {{ issue.description }} - {{
issue.category_id }} - {{ issue.status_id }}</li>
  {% endfor %}
  <a href="{{ url_for('home') }}" class="link">Volver</a>
  <a href="{{ url_for('issue_new') }}" class="link">Nuevo</a>
{% endblock %}
```

MVC - SEPARANDO LA VISTA - TEMPLATES

- ¿Qué es?
- Lo vemos más adelante.

VARIACIONES DEL MVC ORIGINAL

- MTV en Django - Particularidades
- Model
- Template
- View

BUENAS PRÁCTICAS MVC

La idea central detrás de MVC es la **reutilización de código** y la **separación de intereses**.

BUENAS PRÁCTICAS MVC - MODELO

- Puede contener:
    - La lógica necesaria para asegurar que los datos cumplen los requerimientos (validaciones).
    - Código de manipulación de datos.
- NO puede contener:
    - En general, nada que se relacione con el usuario final directamente.
    - Se debe evitar HTML embebido o cualquier código de presentación.

BUENAS PRÁCTICAS MVC - VISTA

- Puede contener:
    - Código de presentación, formatear y dibujar los datos.
- NO puede contener:
    - Código que realice consultas a la BD.

BUENAS PRÁCTICAS MVC - CONTROLADOR

- Puede contener:
    - Creación de instancias del Modelo para pasarle los datos necesarios.
- NO puede contener:
    - Código que realice consultas a la BD (SQL embebido).
    - Se debe evitar HTML embebido o cualquier código de presentación.

PARA SEGUIR VIENDO:

- Lenguaje SQL -> **http://www.w3schools.com/sql/default.asp**
- MVC -> **http://es.wikipedia.org/wiki/Modelo_Vista_Controlador#Frameworks_MVC**

SEGUIMOS EN LA PRÓXIMA ...

```
https://flask.palletsprojects.com/en/2.0.x/quickstart/
```