

# PROYECTO DE SOFTWARE

Cursada 2024

# TEMARIO

- MVC

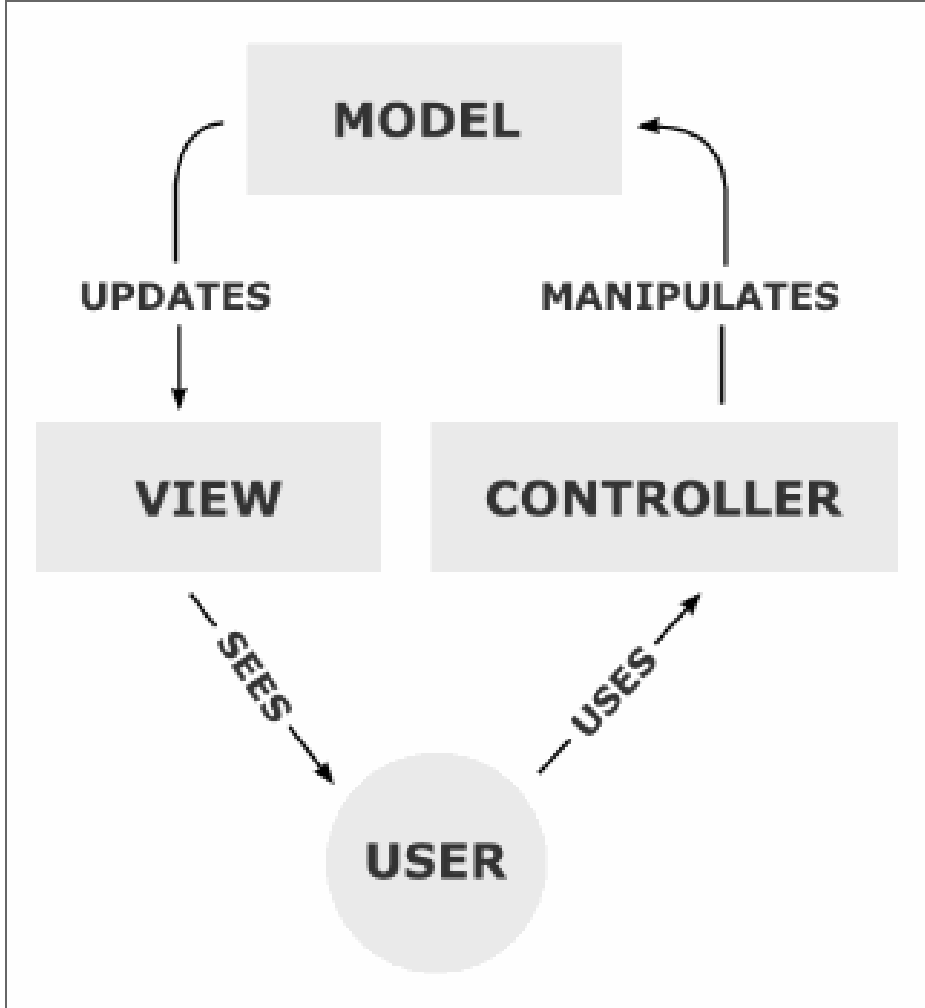
# PATRÓN MVC

# 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.



<https://realpython.com/the-model-view-controller-mvc-paradigm-summarized-with-legos/>

# 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.

# 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
```



# **FLASK**

**¿Problemas? ¿Inconvenientes?**

# MVC - CONFIGURACIÓN

src/web/config.py

```
from os import environ

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

    SECRET_KEY = "secret"
    DEBUG = False
    TESTING = False

class ProductionConfig(Config):
    """Production configuration."""

    pass

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

    DEBUG = True

class TestingConfig(Config):
    """Testing configuration."""

    TESTING = True
```



```
config = {  
  "development": DevelopmentConfig,  
  "test": TestingConfig,  
  "production": ProductionConfig,  
}
```

# MVC – SEPARANDO EL MODELO

src/core/issue.py

```
issues = [  
    {  
        "id": 1,  
        "user": "José",  
        "title": "Mi computadora no funciona.",  
        "description": "Mi departamento me compró una nueva  
computadora y necesito configurarla con todos mis emails y  
documentos de mi vieja computadora.",  
        "status": "new",  
    },  
    {  
        "id": 2,  
        "user": "María",  
        "title": "No puedo obtener mis emails.",  
        "description": "Estoy tratando de acceder a mi correo  
desde casa, pero no puedo obtenerlos. Estoy tratando con Outlook  
en mi casa pero en la oficina tengo Thunderbird.",  
        "status": "in_progress",  
    },  
    {  
        "id": 3,  
        "user": "Rubén",  
        "title": "No puedo imprimir",  
        "description": "Cada vez que trato de imprimir mi  
presentación el programa se cierra. Esto sólo me pasa con  
PowerPoint en Word puedo imprimir. Ya me aseguré que la impresora  
está prendida. Tengo una HP LaserJet 5.",  
        "status": "done",  
    },  
]
```

```
]
```

```
} ,
```

# MVC – SEPARANDO EL MODELO

src/web/\_\_init\_\_.py

```
from flask import Flask, render_template, request
from src.core.issue import issues
.....
```

# MVC - EL CONTROLADOR

src/web/\_\_init\_\_.py

```
def create_app(static_folder="static"):
    app = Flask(__name__, static_folder=static_folder)

    ....

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

    @app.route("/issues/add", methods=["POST"])
    def issues_add():
        issue = {
            "id": request.form.get("id"),
            "user": request.form.get("user"),
            "title": request.form.get("title"),
            "description": request.form.get("description"),
            "status": request.form.get("status"),
        }
        issues.append(issue)
        return render_template("issues/index.html",
```

```
issues=issues)
```

# MVC - EL CONTROLADOR - BLUEPRINT

src/web/controllers/issue.py

```
from flask import Blueprint, render_template, request
from src.core.issue import issues

issue_blueprint = Blueprint("issues", __name__,
url_prefix="/issues")

@issue_blueprint.route("/")
def issues_index():
    return render_template("issues/index.html", issues=issues)

@issue_blueprint.route("/issues/add", methods=["POST"])
def issues_add():
    issue = {
        "id": request.form.get("id"),
        "user": request.form.get("user"),
        "title": request.form.get("title"),
        "description": request.form.get("description"),
        "status": request.form.get("status"),
    }
    issues.append(issue)
```

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



# MVC - EL CONTROLADOR - BLUEPRINT

src/web/\_\_init\_\_.py

```
from src.web.controllers.issue import issue_blueprint
def create_app(static_folder="static"):
    app = Flask(__name__, static_folder=static_folder)
    ....
    app.register_blueprint(issue_blueprint)
    return app
```

# MVC – SEPARANDO LA VISTA

src/web/templates/index.html

```
{% extends "layout.html" %}
{% block title %}Ejemplo Proyecto{% endblock %}
{% block head %}
    {{ super() }}
{% endblock %}
{% block content %}
    {% if contenido == "Argentina" %}
        <h1>La Ciudad Autónoma de Buenos Aires es la
capital de {{contenido}} <h1>
    {% else %}
        <h1>Hola {{contenido}} !!<h1>
    {% endif %}
    <a href="{{url_for('issues.issues_index')}}"> Issues </a>
{% endblock %}
```

# MVC - VISTA

src/web/templates/issues/index.html

```
<form action="{{ url_for('issues.issues_add')}}" method="post">
  <input type="text" name="id" placeholder="Ingrese el id">
  <input type="text" name="user" placeholder="Ingrese el
autor">
  <input type="text" name="title" placeholder="Ingrese el
título">
  <input type="text" name="description" placeholder="Ingrese la
descripción">
  <input type="text" name="status" placeholder="Ingrese el
estado">
  <button>Enviar</button>
</form>
<a href="{{url_for('hello_world')}}"> Volver al inicio </a>
```

# MVC - EXTRAS

src/web/templates/error.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    {% block head %}
      <link rel="stylesheet" href="{{ url_for('static',
filename='error.css') }}" />
    {% endblock %}
    <title>Error!</title>
  </head>
  <body>
    <h1>{{ error_name }}</h1>
    <p class="zoom-area"><b>Error!</b> {{ error_description }}
  </p>
    <div class="link-container">
      <a href="{{ url_for('hello_world') }}" class="more-
link">Go to the homepage</a>
    </div>
  </body>
</html>
```

# MVC - EXTRAS

src/web/helpers/handler.py

```
from flask import render_template

def not_found_error(e):
    kwargs = {
        "error_name": "404 Not Found Error",
        "error_description": "La url a la que quiere acceder no
existe",
    }
    return render_template("error.html", **kwargs), 404

def generic_error(e):
    kwargs = {
        "error_name": "500 Internal Server Error",
        "error_description": "Error interno del servidor",
    }
    return render_template("error.html", **kwargs), 500
```

# MVC - EXTRAS

src/web/\_\_init\_\_.py

```
from src.web.helpers import handler

def create_app(static_folder="static"):
    app = Flask(__name__, static_folder=static_folder)

    ....

    app.register_blueprint(issue_blueprint)

    # Error handlers
    app.register_error_handler(404, handler.not_found_error)
    app.register_error_handler(500, handler.generic_error)

    return app
```

## **PARA SEGUIR VIENDO:**

- MVC ->

[http://es.wikipedia.org/wiki/Modelo\\_Vista\\_Controlador#Frameworks MVC](http://es.wikipedia.org/wiki/Modelo_Vista_Controlador#Frameworks_MVC)

# SEGUIMOS EN LA PRÓXIMA ...

<https://flask.palletsprojects.com/en/3.0.x/>

Speaker notes