PROYECTO DE SOFTWARE

Cursada 2021

TEMAS DE HOY

- Repaso Entornos virtuales, Flask y Sesiones
- Acceso a Bases de Datos.
- ORM

REPASO ENTORNOS VIRTUALES

- Multiples versiones del lenguajes.
- Multiples versiones de librerías.
- Requerimientos mínimos.

REPASO ENTORNOS VIRTUALES

- Comando útiles:

```
virtualenv entornoProyecto
source entornoProyecto/bin/activate
deactivate
pip freeze > requirements.txt
```

REPASO FLASK

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


@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

- 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

- 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 %}
```

REPASANDO: MANEJO DE SESIONES

- ¿Qué es una sesión?
- ¿Porqué son necesarias?
- ¿Qué son las cookies?
- ¿Para qué sirven las cookies?
- ¿Dónde se alojan las cookies?
- ¿Dónde se aloja la sesión?

DE NUEVO: ¿DÓNDE SE ALOJA LA SESIÓN EN FLASK?

- Depende...
- Tradicionalmente (en lenguajes como PHP) la sesión es almacenada en un archivo en el servidor, el cliente guarda una cookie que SÓLO posee el sessionID para identificarla.

SESIONES EN FLASK

- Por defecto Flask usa sesiones basadas en cookies (**session cookie**).
- La información de sesión se almacena **en el cliente** en una cookie firmada con una **secret key**.
- Cualquier modificación a la cookie queda invalidada por su firma. Pero es **visible en todo momento** en el cliente.
- No es aconsejable guardar información sensible en una session cookie.

Veamos una de estas sesiones en **http://localhost:5000/iniciar_sesion** decodificadas con **https://github.com/noraj/flask-session-cookie-manager**

SESIONES EN FLASK - FLASK-SESSION

- Flask posee extrensiones como **Flask-Session** que permiten un mejor manejo de las sesiones.
- Con Flask-Session podemos elegir diferentes lugares donde almacenar la sesión en el servidor:
    - redis
    - memcached
    - **filesystem**
    - mongodb
    - sqlalchemy
- Se instala con pip: **pip3 install Flask-Session**.

USO DE FLASK-SESSION

```
from flask_session import Session
# Configuración inicial de la app
app = Flask(__name__)
app.config.from_object(Config)
#Server Side session
app.config['SESSION_TYPE'] = 'filesystem'
Session(app)
```

- Modifiquenos la app y veamos de nuevo... **http://localhost:5000/iniciar_sesion**

Referencias:

- Sesiones en flask: **https://overiq.com/flask-101/sessions-in-flask/**
- Flask-Session: **https://flask-session.readthedocs.io/en/latest/**

ACCEDIENDO A BASES DE DATOS

LENGUAJE SQL (STRUCTURED QUERY LANGUAGE)

- Sentencias insert, update, select, etc....
- Ejemplos:
    - select * from tabla where condición
    - insert into tabla (campos) values (valores)
    - update tabla set campo1='valor1' where condición

IMPORTANTE

MySQL: motor de base de datos

SQL: lenguaje de consulta

MYSQL - MARIADB

- El archivo más importante de MySQL es su configuración que la encontramos en /etc/mysql/my.cnf
- Por defecto el log esta deshabilitado por cuestiones de performance.
- Ejemplos de configuración avanzados: /usr/share/doc/mysql-server-5.1/examples

Consejos

- Habilitar los logs de la manera más descriptiva posible
- Instalar alguna aplicación que nos permita acceder de manera menos complicada a la base de datos como **PhpMyAdmin**

MYSQL

Habilitar el Log de todas las transacciones:

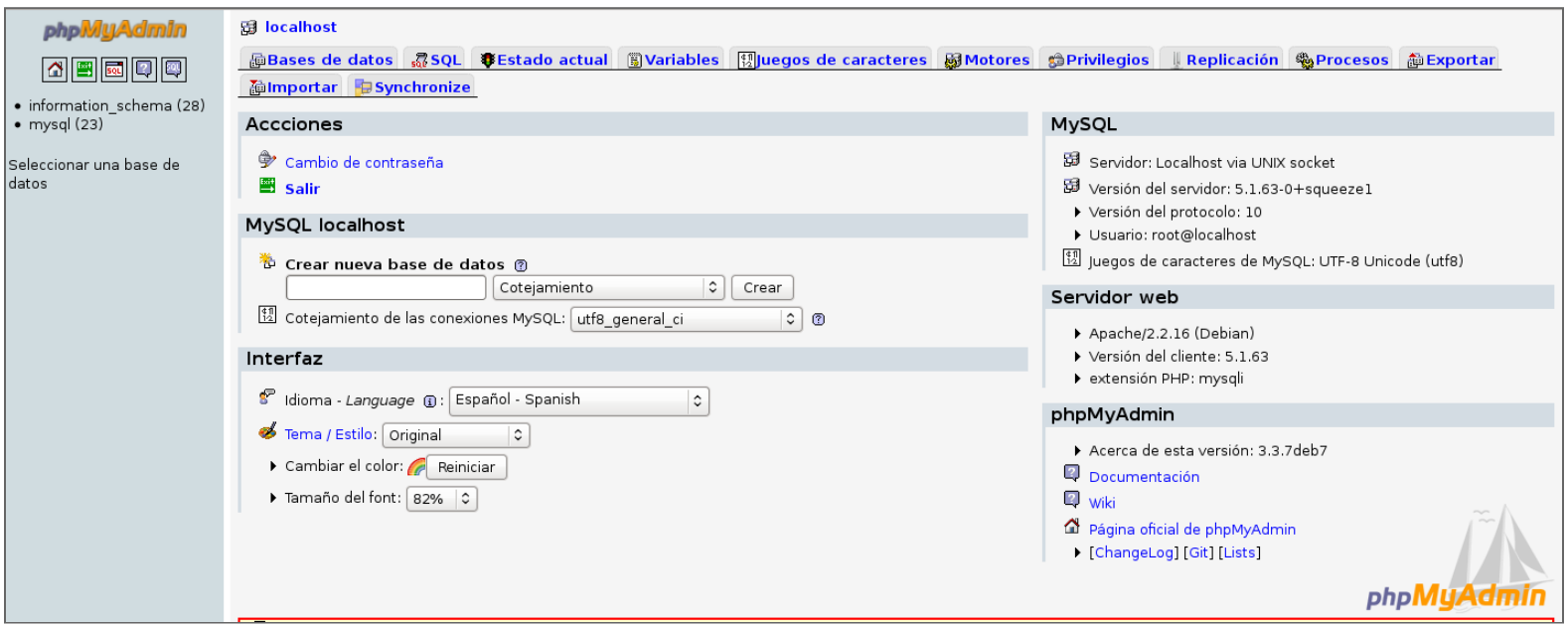
- general_log_file = /var/log/mysql/mysql.log
- general_log = 1

Loguear consultas lentas y sin índices:

- log_slow_queries = /var/log/mysql/mysql-slow.log
- long_query_time = 2
- log-queries-not-using-indexes

PHPMYADMIN

- Interfaz de Administración de la Base de Datos MySQL
- Podemos exportar e importar a varios formatos



- Sitio oficial: **http://www.phpmyadmin.net**
- PhpMyAdmin de la cátedra: **https://sql.proyecto2021.linti.unlp.edu.ar/**

ACCESO A BBDD – MYSQL

- Vamos a acceder a través de pymysql.

PYMYSQL

```
from flask import Flask, render_template, request, session
from flask_session import Session
import pymysql

app = Flask(__name__)

app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"

Session(app)

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


def connection():
    db_conn = pymysql.connect(
        host="localhost",
        user="proyecto2021",
        password="proyecto2021",
        db="proyecto2021",
        cursorclass=pymysql.cursors.DictCursor,
    )
    return db_conn
```

PYMYSQL

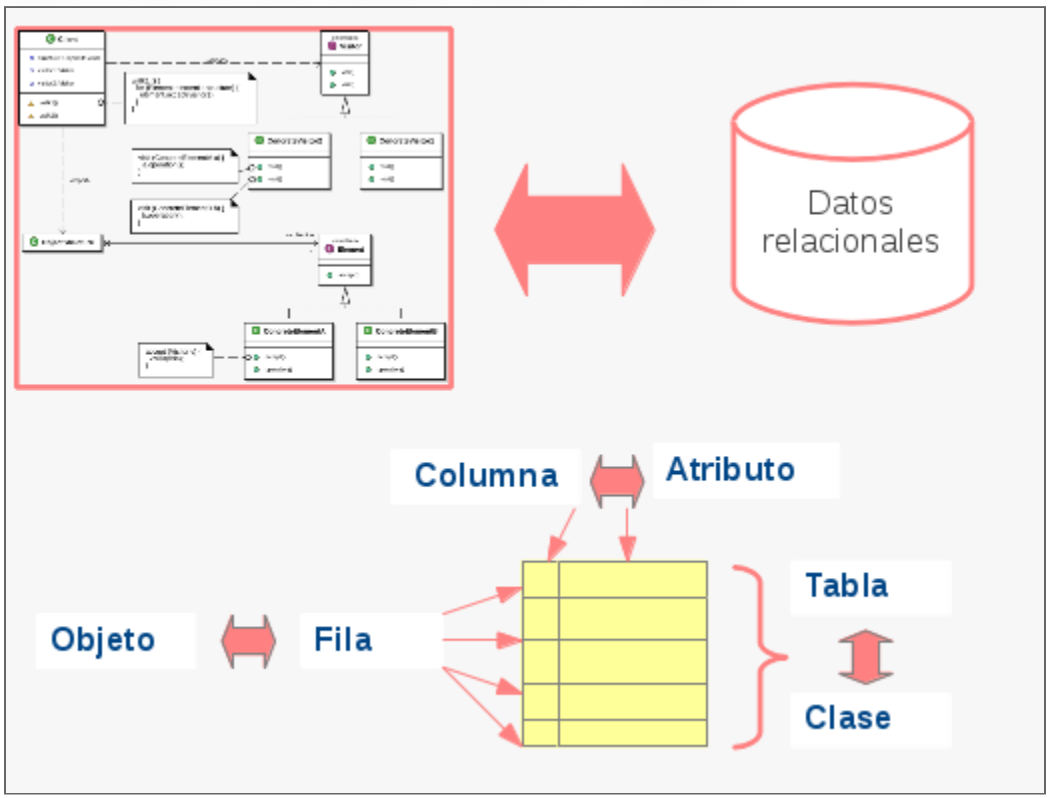
```
@app.route("/equipos")
def equipos():
    db = connection()

    with db.cursor() as cursor:
        res = cursor.execute("select * from equipos")

    if res:
        equipos = cursor.fetchall()
    else:
        equipos = []

    return render_template("equipos.html",
contenido=equipos)

@app.route("/agregar", methods=["POST"])
def agregar():
    equipo = request.form.get("equipo")
    posicion = request.form.get("posicion")

    db = connection()

    with db.cursor() as cursor:
        res = cursor.execute("insert into equipos
(descripción, posicion) values (%s, %s)",
                             (equipo, posicion))
        db.commit()
```

```
    with db.cursor() as cursor:
        res = cursor.execute("select * from equipos")

    if res:
        equipos = cursor.fetchall()
    else:
        equipos = []


    session["pos_equipos"][equipo] = posicion
    return render_template("equipos.html",
contenido=equipos)
```

AHORA BIEN ...

- ¿Qué pasa si queremos migrar de motor de BDD?
- ¿Qué pasa si queremos tener múltiples BDD conectadas?

UNA VUELTA DE ROSCA MÁS -> ORM



ORM - OBJECT-RELATIONAL MAPPING

- Mapeo de objetos a base de datos relacionales.
- Permite acceder a una base de datos relacional como si fuera orientada a objetos.
- Transforma las llamadas a los objetos en consultas SQL, que permiten independizar el motor de BD utilizado en la aplicación.
- De acuerdo a la implementación se utiliza una sintaxis diferente.

¿POR QUÉ?

- BBDD relacionales
    - Datos escalares: números, cadenas de texto, etc…
- Aplicaciones orientadas a objetos.
    - Objetos con propiedades y métodos.
- ¿Problema?
    - Convertir los objetos en datos escalares para grabar los datos en la base de datos y volver a convertirlos en objetos al recuperarlos.

ORM

- Ventajas
    - Abstracción a la BBDD.
    - Reutilización de código.
- Desventajas
    - Otra capa intermedia.
    - Otro lenguaje de consulta.
- Algunos ORMs
    - Sqlalchemy
    - SQLObject
    - Django ORM

SQLALCHEMY: AGREGAMOS UNA CAPA DE ABSTRACCIÓN

- - Ver: **SQLAlchemy**
- Herramientas de Abstracción y ORM (mapea objeto a relacional)
- Múltiples motores de bases de datos: **https://docs.sqlalchemy.org/en/14/dialects/index.html**

COMPARANDO

- A mayor abstracción menos control .
- Al abstraernos es posible que se compliquen algunas funciones propias del motor que queremos usar.
- ¿La abstracción es necesaria?

EJEMPLO SQLALCHEMY

- index.py

```
from flask import Flask, render_template, request, session
from flask_session import Session
from flask_sqlalchemy import SQLAlchemy
import os
from equipos import *



app = Flask(__name__)

app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
app.config["SQLALCHEMY_DATABASE_URI"] =
os.environ.get("BBDD")
db = SQLAlchemy(app)
Session(app)
```

EJEMPLO SQLALCHEMY

- index.py

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


@app.route("/equipos")
def equipos():

    equipos = Equipo.query.all()
    return render_template("equipos.html",
contenido=equipos)

@app.route("/agregar", methods=["POST"])
def agregar():
    equipo = request.form.get("equipo")
    posicion = request.form.get("posicion")

    nuevo = Equipo(descripcion=equipo, posicion=posicion)
    db.session.add(nuevo)
    db.session.commit()

    equipos = Equipo.query.all()

    return render_template("equipos.html",
contenido=equipos)
```

EJEMPLO SQLALCHEMY

- equipos.py

```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class Equipo(db.Model):

    __tablename__ = "equipos"
    id = db.Column(db.Integer, primary_key=True)
    descripcion = db.Column(db.String(45), unique=True,
nullable=False)
    posicion = db.Column(db.String(45), unique=False,
nullable=False)

    def __repr__(self):
        return '<Equipo %r>' % self.descripcion
```

TAREA PARA EL HOGAR

- ¿Qué es SQL injection?

EJEMPLOS

```
https://proyecto-de-
software.github.io/2021/01_teorias/clase6_1.zip
```

FIN