Exposición sobre Python
Lenguajes del back-end

Indice:

Realizado por:

- Sixto Coca Cruz
- Jose Miguel Pelegrina Pelegrina

¿Qué es el back-end?

El backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la comunicación con el servidor.

Algunas de las funciones que se gestionan en la parte del back-end son:

- El Desarrollo de funciones que simplifiquen el proceso de desarrollo.
- Acciones de lógica.
- Conexión con bases de datos.
- Uso de librerías del servidor web (por ejemplo para implementar temas de caché o para comprimir las imágenes de la web).

Además, tiene que velar por la seguridad de los sitios web que gestiona y optimizar al máximo los recursos para que las páginas sean ligeras.

Uno de los lenguajes de programación más utilizados para el desorrollo de back-end es Python.

¿Qué es python?

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Los expertos promueven el Python como el lenguaje ideal para el aprendizaje automático y la creación de redes neuronales. Además, Python es también la elección estándar para realizar análisis de conjuntos de datos a gran escala para averiguar las tendencias del mercado, los precios medios y predecir patrones futuros. Sin embargo, Python para el desarrollo web es también una sólida opción, atrayendo a la gente con su sintaxis simple, una variedad de marcos de trabajo, y un fuerte ecosistema general.

Ventajas de desarrollo web en Python

- **Es un lenguaje sencillo y rápido** de aprender
- Su sintaxis es parecida a **escribir cualquier texto** en inglés
- Posee su **propio gestor de paquetes** (sin necesidad de instalar aplicaciones externas).
- **No necesita un ecosistema para ejecutarse**, como puede ser Xampp, Vangrant, Docker…
- Es el **segundo lenguajes que mejor esta pagado** por las empresas. Por detrás de Ruby.

Variedad de Frameworks

Una de las características más notorias de Python es su ecosistema. Específicamente, el número de farmeworks que están disponibles. Dado que el uso del desarrollo web en Python se ha incrementado significativamente, hay muchos módulos orientados a la web diseñados para ayudar a la gente a producir sitios web funcionales.

Cuando se trata de frameworks, no hay un mejor framework web en Python que todo el mundo necesite aprender para hacer posible el desarrollo web con Python. Puedes separarte de los módulos existentes y construir el lado del servidor de forma individual. Sin embargo, los frameworks son muy útiles debido a que aceleran el proceso de desarrollo.

Otro aspecto importante es que el framework más rico en características puede no ser adecuado para todo el mundo. Si seleccionas los frameworks de acuerdo a su popularidad y comunidades de usuarios, podrías perder tiempo en aprender un módulo que ni siquiera provee las características que necesitas para tu proyecto.

En el ámbito del desarrollo web en Python, hay varios frameworks dominantes: Django, Web2py, Pyramid, Flask, TurboGears, Bottle y CherryPy.
Vamos a comentar brevemente algunos de ellos.

Django

Django es un framework de alto nivel escrito en python que fomenta el desarrollo rápido y el diseño limpio y pragmático. Se ocupa de gran parte de la molestia del desarrollo web, por lo que puede concentrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto.

Ventajas:

- Es muy rápido: podéis construir una aplicación muy buena en poco tiempo.
- Viene bien cargado : Cualquier cosa que necesitéis realizar, ya estará implementada, sólo hay que adaptarla a vuestras necesidades.
- Es bastante seguro : Ya que implementa por defecto algunas medidas de seguridad, las más clásicas, para que no haya SQL Injection, no haya Cross site request forgery (CSRF) o no haya Clickjacking por JavaScript.
- Es muy escalable
- Es increíblemente versátil

Ruby on Rails

Ruby on Rails, también conocido como RoR o Rails, es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma del patrón Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración.

El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

Meteor

Meteor, o MeteorJS, es un framework para aplicaciones web con JavaScript libre y de código abierto escrito usando Node.js. Meteor facilita la creación rápida de prototipos y produce código multiplataforma (web, Android, iOS y escritorio).

Se integra con MongoDB y usa Distributed Data Protocol y un patrón publish-subscribe para propagar automáticamente al cliente cambios en los datos sin requerir que el desarrollador escriba algún código de sincronización. En el cliente, Meteor depende de jQuery y puede ser usado con cualquier librería de UI para JavaScript.

Flask

Flask es un “micro” Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC.

La palabra “micro” no designa a que sea un proyecto pequeño o que nos permita hacer páginas web pequeñas sino que al instalar Flask tenemos las herramientas necesarias para crear una aplicación web funcional pero si se necesita en algún momento una nueva funcionalidad hay un conjunto muy grande extensiones (plugins) que se pueden instalar con Flask que le van dotando de funcionalidad.

Ventajas de usar Flask:

1. Flask es un “micro” Framework: Para desarrollar una App básica o que se quiera desarrollar de una forma ágil y rápida Flask puede ser muy conveniente, para determinadas aplicaciones no se necesitan muchas extensiones y es suficiente.
2. Incluye un servidor web de desarrollo: No se necesita una infraestructura con un servidor web para probar las aplicaciones.
3. Tiene un depurador y soporte integrado para pruebas unitarias
4. Buen manejo de rutas
5. Soporta de manera nativa el uso de cookies seguras.
6. Sirve para construir servicios web (como APIs REST) o aplicaciones de contenido estático.
7. Es Open Source y está amparado bajo una licencia BSD.
8. Buena documentación , código de GitHub y lista de correos.

Caso práctico: Uso de Python con Flask

Vamos a ver un ejemplo simple de cómo servir una página web, conectarla a una base de datos y coger valores introducidos en esta página web para modificar la base de datos.

En primer lugar para poder trabajar con Flask tenemos que importarlo. Utilizando las siguientes línea de código empezamos a servir una página web simple.

```
app = Flask(__name__)

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

if __name__ == "__main__":
    app.run(port=5002)
```

Se declara la variable app con Flask(), definimos la ruta por defecto (‘/’), renderizamos el fichero index.html con render_template y le decimos que ejecute la aplicación en el puerto 5002. Mientras esto se esté ejecutando se servirá la página web.(Muchas de estas funciones tienen que estar importadas en nuestro código estan importadas todas al principio del fichero)

```
sixto@sixtoubuntu:~/Escritorio/PythonTW/FlaskApp$ python app.py
 Serving Flask app "app" (lazy loading)
 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://127.0.0.1:5002/ (Press CTRL+C to quit)
```

Creamos nuestra base de datos y creamos un procedimiento para crear un usuario en esta tabla y que nos asegure que los valores introducidos son correctos, si no nos mandara un error.El Procedimiento comprueba si el nombre de ese usuario ya existe y si no existe crea un usuario.

Para poder utilizar nuestros datos de registro introducidos en la pagina web utilizaremos jquery ajax, así que en la funcion tendremos que indicarlo.

```
@app.route('/signUp',methods=['POST','GET'])
def signUp():
    try:
        _name = request.form['inputName']
        _email = request.form['inputEmail']
        _password = request.form['inputPassword']
```

añadiendo en route estos methods ya podemos utilizar los valores enviados por jquery Ajax.Usando request podemos leer los valores escritos en la página web.(tienen que ser importada los métodos request).

Para crear una solicitud a nuestro método signUp vamos a añadir el siguiente evento al botón de registro,Una solicitud Post de jQuery cuando se pulse el botón de registro.

```
$(function(){
    $('#btnSignUp').click(function(){

        $.ajax({
            url: '/signUp',
            data: $('form').serialize(),
            type: 'POST',
            success: function(response){
                console.log(response);
            },
            error: function(error){
                console.log(error);
            }
        });
    });
});
```

Lo siguiente que tenemos que hacer es almacenar estos datos en nuestra base de datos utilizando mysql, lo primero es añadir las configuraciones de mysql(usuario,contraseña,base de datos,host)

```
# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'sixto'
app.config['MYSQL_DATABASE_PASSWORD'] = 'sixto'
app.config['MYSQL_DATABASE_DB'] = 'BucketList'
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
mysql.init_app(app)
```

Después de tener estos datos creamos la conexión con conn = mysql.connect() como nosotros queremos utilizar un procedimiento almacenado tenemos que crear un cursor para que este pueda llamarlo con cursor = conn.cursor(todo esto viene importado de un paquete).En nuestro ejemplo se cifra la contraseña y llamamos al procedimiento de la siguiente manera.

```
            _hashed_password = generate_password_hash(_password)
            cursor.callproc('sp_createUser',(_name, _email, _hashed_password))
```

Una vez llamado el proceso nos devuelve mensajes de control. Al final de todo cerramos la conexión a la base de datos y el cursor con conn.close() y cursor.close().

En resumen es una aplicación que sirve la página web recoge datos introducidos en esta página y luego comprueba estos datos y los añade, si son correctos, a nuestra tabla.