Práctica 2

1) Aplicación básica

El archivo *pr2.1,py* es el que se ha implementado para este ejercicio, simplemente muestra un mensaje de "**Hello World**" en la dirección de la home de la página.

2) Sirviendo contenidos estáticos

La implementación de este ejercicio se encuentra en el archivo *pr2-2.py*. Devuelve una página en formato HTML localizado en la carpeta *templates* con el estilo que defina la hoja de estilo *style.css* que se encuentra en la carpeta *static/css*. Se muestra también una imagen estática.

3) Manejo de URLs

El archivo *pr2-3.py* tiene toda la información necesaria para mostrar al usuario una página cuando este escriba una determinada dirección en la URL. Por ejemplo si el usuario escribe *"localhost:puerto/user*" éste verá un mensaje de "**Hola Mundo!**". En cambio se se dirige a la url "*localhost:puerto/user/usuario*" mostrará un mensaje diferente en función de la cadena proporcionada en "*usuario*", es decir, si escribimos "*localhost:puerto/user/antonio*" nos devolverá un mensaje de "**Bienvenido antonio!**".

4) Creando Imágenes Dinámicas [Fractales]

En el archivo *pr2-4.py* se ha implementado una solución para mostrar un fractal de *n* colores dependiendo de una serie de parámetros como son las coordenadas (x1, y1, x2, y2) el ancho de la imagen y el número de iteraciones. Estos datos se introducen en un formulario que se encuentra en la raíz de la web y se generará el fractal. Si el ancho y/o número de iteraciones es alto tardará más tiempo en generarlo. (Datos de ejemplo para la visualización x1=-1 y1=-1 x2=1 y2=1 ancho=800 iteraciones=300. Seleccionar colores diferentes)

También se ha implementado una caché para que cuando se introduzcan unos datos para el fractal que ya han sido introducidos previamente, se use la imagen generada anteriormente sin esperar a procesarla de nuevo.

En el archivo *clean_cache.py* se ha mejorado la caché, con la limitación de aquellas imágenes que se encuentren más de un dia en el disco. Este script se ejecutará como un demonio del sistema operativo que comprobará en segundo plano si las imágenes llevan creadas más de un día y de ser así se borran.

5) Creando Imágenes Dinámicas [Vectoriales]

En el apartado anterior se han generado una serie de imágenes dependiendo de los datos que el usuario introduce en el formulario. En este ejercicio, archivo *pr2-5.py* se ha implementado una aplicación web para la generación de imágenes SVG(Scalable Vector Graphics) aleatorias. La aplicación selecciona de forma automática diferentes tipos de SVG, por ejemplo, círculos, rectángulos, estrellas, etc con una serie de parámetros que también son aleatorios, por tanto por cada vez que se acceda a la página web, ésta mostrará una imagen SVG de diferente tipo, tamaño y color

Práctica 3

Esta aplicación se encuentra organizada de la siguiente forma:

- / (Raíz de la aplicación)
    - pr.py (Programa en Python)
    - static (Contenido estático que utilizará la aplicación)
        - css
        - fonts
        - images
    - templates (Archivos HTML para dar formato a la web)
    - data.db (Archivo de la base de datos con Shelve)
    - data.key (Archivo de correspondencia del usuario con la key de la DB)

Páginas web accesibles

- Home (Raíz)
- Register
- Profile/View (Para ver los datos de usuario registrado)
- Profile/Edit (Para modificar los datos de usuario registrado
- Blog
- Login (Solo se usa para el POST del formulario de login)
- Road (Para ver los últimos links visitados)

Templates y estilo

Se ha usado un template extraido de la web http://www.freewebsitetemplates.com/ a Flask y con el uso de templates. Se ha realizado mediante la adaptación de dicho template con sus correspondientes archivos en HTML puro a los archivos *layout.html* y *template.html*. Layout contiene lo que sería la base de la página web, contiene ciertos elementos que van a ser comunes en todas las páginas como puede ser la barra superior de navegación, los menús y el footer. Mientras, template contendrá todos los bloques o contenido de elementos que se van a extender del layout, dependiendo de la página web en la nos encontremos en el momento de la navegación.

También se ha utilizado la hoja de estilos(*css/style.css*) que traía por defecto con una serie de modificaciones.

Manejo de sesiones

Para poder mantener un usuario activo se ha usado la variable global session donde una vez que un usuario se ha registrado en la web se actualiza esta variable con los siguientes datos:

- username (Cadena que representa el nombre de usuario)
- password (Contraseña del usuario)
- name (El nombre del usuario)
- lastname (El apellido del usuario)
- address (Domicilio del usuario)
- key (Clave única del usuario que se corresponde con la clave de la base de datos)
- url1, url2, url3 (Camino o recorrido de las últimas URLs visitadas)

Cuando un usuario va a identificarse en la web, se hace uso de la variable session para más adelante comprobar si dicho usuario tiene las credenciales necesarias para entrar con su cuenta. De no ser así podrá registrarse rellenando un formulario en la dirección "*localhost:puerto/register*".

Cada vez que un usuario entra en una página web, se registra dicha dirección en la variable session, concretamente como último enlace visitado.

Ciertos tipos de elementos de las páginas están restringidas a usuarios con cuenta, es por ello que para esto la sesión activa de usuario es fundamental. Una vez que el usuario está con su cuenta, recibirá un mensaje de bienvenida junto con su nombre por cada página que visite, además, de que en el menú superior derecho verá más opciones, como por ejemplo, acceso a su perfil (antes aparecia la opción de registro) y la opción de ver sus últimos enlaces visitados.

Persistencia sencilla con Shelve

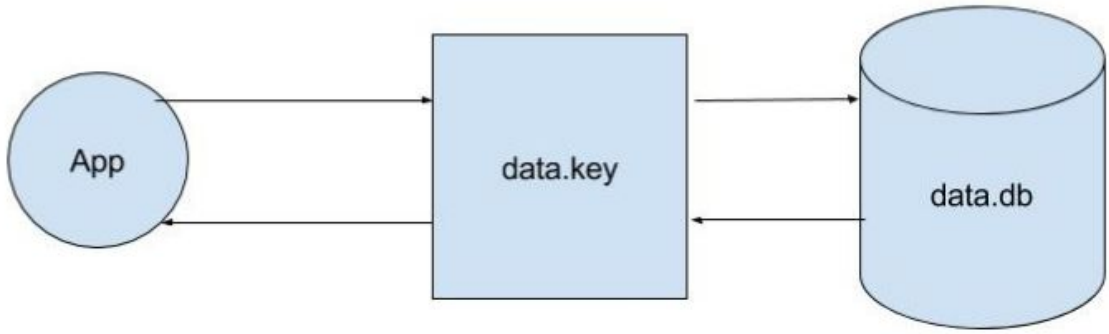
Para poder hacer establecer sesiones con un usuario que ya se ha registrado anteriormente de manera permanente y no temporal se hará uso de una base de datos con Shelve. De esta forma tanto si el usuario se va de la página como si se reinician las sesiones en el lado del servidor, pueda seguir teniendo constancia de que dicho usuario se registró con anterioridad y que posee una cuenta con sus datos.

Habrá tres momentos en los que la aplicación contacte con la base de datos:

- Cuando un usuario se de alta en la web para introducir los nuevos datos en la base de datos.
- Cuando un usuario se identifique, para comprobar que efectivamente sus datos de usuario y contraseña son correctos.

- Cuando un usuario modifique sus datos a través de su perfil.

La aplicación contacta con base de datos de la siguiente forma:



El archivo *data.key* realiza una correspondencia entre el nombre de usuario y la clave única de un usuario para acceder a la base de datos. De esta forma la clave de la DB no es el nombre de usuario, sino una clave numérica que identifica a dicho usuario.

Los archivos *data.key* y *data.db* inicialmente no existirán, pero se generarán tras el primer registro en la aplicación.

Nota: Para la correcta visualización del HTML al cambiar de ejercicios y/o de práctica es recomendable borrar la caché del navegador (Pulsar Ctrl + F5)