Manual de Estilo de Programación SAGE

SAFE Software DeVeloPment V1.0

Conocimientos previos:

Se asume que el lector tiene conocimiento de los siguientes tópicos:

- Lenguaje de programación Python.
- Orientación a objetos.
- Código HTML5 y CSS.
- Web framework Django.
- Suite de pruebas PyUnit.

Distribución de aplicaciones:

Las aplicaciones de SAGE deben estar almacenadas en directorios identificados con nombres relacionados con su funcionalidad. Por ejemplo si los componentes de una aplicación son todos relacionados con los usuarios del sistema, el nombre del directorio de la aplicación debería llamarse usuarios.

Los *urls*, *models*, views, *forms*, *controllers*, *templates* y demás archivos pertinentes deben almacenarse en la carpeta de la aplicación a la que están relacionados.

Nomenclatura de componentes:

Directorios de aplicaciones:

Deben estar identificados con la notación *snake_case*, identificada apropiadamente con la aplicación relacionada.

Ejemplos:

- pagina_principal – Aplicación que agrupa dentro de sí los componentes necesarios para mostrar la página que se muestra apenas se abre SAGE.
- reservas – Aplicación que contiene los archivos necesarios para manejar las reservas en los estacionamientos.

Templates:

Nombradas bajo la notación *snake_case*. Deben empezar por el nombre de la aplicación a la que están relacionadas, seguido de la función que cumplen en la interfaz del usuario.

Ejemplos:

- billetera_crear – Este template está destinado a mostrar la interfaz de creación de una billetera electrónica.
- estacionamiento_detalle – La utilidad de este template es mostrar en pantalla los detalles relacionados con un estacionamiento dado.

Controllers:

Se debe utilizar notación UpperCamelCase y deben estar debidamente nombrados de forma que su función resulte fácil de interpretar.

Ejemplos:

- def ConsultarIngresos() – Controller para calcular los ingresos de un estacionamiento dado un RIF.
- def SeleccionarFeriados() – Controller que almacena en la base de datos los feriados especificados para cada estacionamiento.

Forms:

Usar notación UpperCamelCase adjuntado "Form" al final. Deben tener el nombre del modelo al que están adjuntados o su utilidad especifica

Ejemplos:

- def EstacionamientoExtendedForm() – Formulario utilizado para pedir al usuario detalles adicionales para la parametrización del estacionamiento.
- def PropietarioForm() – Formulario que recibe los datos de los propietarios de estacionamientos.

Views:

Las views deben estar nombradas en notación Snake_Case con las primeras letras de cada palabra en mayúscula. Adicionalmente debe describir brevemente la función del view en la aplicación.

Ejemplos:

- Estacionamiento_Cancelar_Reserva
- Consultar_Saldo
- Billetera_Pagar
- Billetera_Recargar
- Modo_Pago

URLs:

Los nombres deben concatenar las palabras utilizadas con todas las letras en minúscula.

Adicionalmente como regla general, los url hijos deben estar relacionados con su directorio padre.

Ejemplos:

- https://safesdv.sage.ve/propietario
- https://safesdv.sage.ve/propietario/editar
- https://safesdv.sage.ve/estacionamiento/consultaingresos
- https://safesdv.sage.ve/estacionamiento/feriados

- https://safesdv.sage.ve/usuario/reservas

Models:

Los nombres de modelos deben ser breves y explicativos utilizando la convención UpperCamelCase.

Los campos del modelo utilizan la notación snake_case y no deben ser redundantes, es decir que si la clase se llama Propietario no deben existir campos llamados nombre_propietario o cedula_propietario.

Ejemplos:

- ```
  class Reserva(models.Model):
      tipo_cedula     = models.CharField(max_length = 1)
      cedula          = models.CharField(max_length = 10)
      nombre          = models.CharField(max_length = 64)
      apellido        = models.CharField(max_length = 64)
      estacionamiento = models.ForeignKey(Estacionamiento)
      inicio          = models.DateTimeField()
      final           = models.DateTimeField()
      estado          = models.CharField(max_length = 25)
      tipo_vehiculo   = models.CharField(max_length = 7)
  ```

Estilo:

Templates:

Cada bloque del template que esté anidado dentro de otra etiqueta debe estar identada con 4 espacios de forma que sea fácilmente identificable cada uno de los componentes de la página.

Ejemplos:

```

    <ul>
        <li><a href="/index">
            Link 1
        </a></li>
        <li><a href="/index">
            Link 2
        </a></li>
    </ul>

```

Los templates deben evitar el uso de tablas a excepción de aquellos lugares en que sea estrictamente necesario (Ej. Inventarios).

No implementar directamente en ningún template componentes de estilo visual de la página web, en su lugar utilizar etiquetas de clase e IDs de archivos css.
```

Si varios archivos html comparten características similares considerar utilizar una plantilla base.

Imports:

Independientemente del archivo, los import deben estar agrupados por tipo (controllers, views, models).

Ejemplo:

```
from estacionamientos.controller import (
    HorarioEstacionamiento,
    get_client_ip,
    tasa_reservaciones,
    calcular_porcentaje_de_tasa,
    consultar_ingresos,
    seleccionar_feriados,
    seleccionar_feriado_extra,
    limpiarEsquemasTarifarios
    )
from reservas.controller import (
    validarHorarioReserva,
    marzullo,
    calcular_Precio_Reserva,
    )

from billetera.forms import (
    BilleteraElectronicaForm
    )
from estacionamientos.forms import (
    EstacionamientoExtendedForm,
    EstacionamientoForm,
    EditarEstacionamientoForm,
    RifForm,
    CedulaForm,
    ElegirFechaForm,
    AgregarFeriadoForm,
    )
from reservas.forms import (
    ReservaForm,
    PagoForm,
    )

from estacionamientos.models import (
    Estacionamiento,
    EsquemaTarifario,
    EsquemaTarifarioM2M,
    TarifaHora,
```

```
TarifaMinuto,
TarifaHorayFraccion,
TarifaFinDeSemana,
TarifaHoraPico,
DiasFeriadosEscogidos,
)
```