Contenido

Práctica Cafetería Barista

Parte 1 – Configuración y vistas básicas

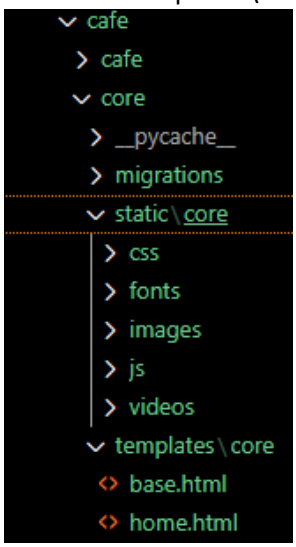
Descarga de archivos

1. Visita el sitio https://www.tooplate.com/
2. Descarga el template: https://www.tooplate.com/download/2137_barista_cafe
3. Descomprime el archivo .zip

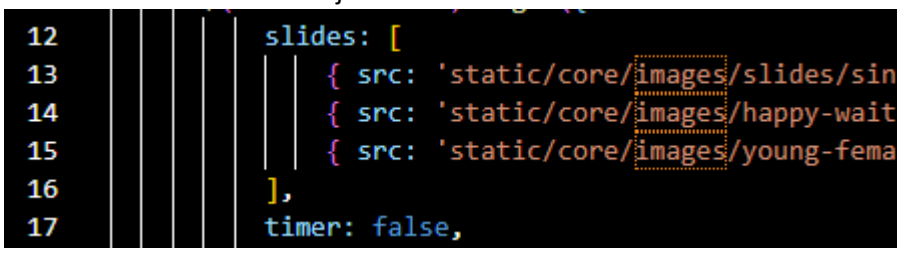
Ambiente de trabajo

1. Opcional – Crea entorno virtual y área de trabajo
2. Crea el proyecto “cafe”
    a. Puede ser la misma carpeta donde has estado trabajando (Diplomado)

Estructura del proyecto

1. Crea la aplicación “core”
2. Crea la estructura “static” y copiar las carpetas css, fonts, images, js y videos
3. Crea la estructura templates\core
    a. 

4. Actualiza el archivo custom.js



Plantillas base y home

1. Crea el archivo base.html
    a. Tip: Revisa el archivo base.html del proyecto webRestaurante
    b. Copia el contenido del “index.html”
        i. Asegurate de solo quedar con el Menu de opción y el footer

    c. Actualiza las referencias hacia los archivos estáticos
        i. Carga el template tag “static”
        ii. Dentro del header, actualiza las referencias hacia los archivos css
        iii. En la referencia a la imagen coffee-beans.png, actualiza la referencia
        iv. En la sección del footer, actualiza las referencias de los archivos js
    d. Recuerda colocar el bloque para en contenido mediante el template tag “block”
2. Crea el archivo home.html
    a. Tip: Revisa el archivo home.html del proyecto webRestaurante
    b. Debes heredar el código de base.html (extends..)
    c. Incluir la “sección 1” dentro del bloque “content”

Vista y urls

1. Crea el método “home” dentro de las vistas de la aplicación “core”
    a. En el método home, debes renderizar la salida hacia el template home.html
2. Crea el archivo urls.py para la aplicación “core”
    a. Tip: Utiliza como guía el archivo urls.py de la aplicación “core” del proyecto webRestaurante
3. Modificar el archivo urls.py principal para incluir las urls de “core”

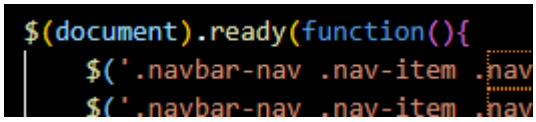
Pruebas

1. Agrega la aplicación “core” en el archivo settings
2. Realiza las migraciones de base de datos
3. Crea el super usuario “admin”
4. Ejecuta la aplicación y verifica su correcto funcionamiento

Parte 2 – Template About

1. Crear el template about.html
    a. Copiar la sección 2 y la sección “barista-team”
    b. Cambia la referencia a estatic del video y de las imágenes que se muestran
2. Crear el método about dentro del archivo views.py
3. Agregar el url about
4. Modificar el template base.html
    a. En el Menu, colocar el url correspondiente a home y about
    b. Colocar la clase active correspondiente

    c. En el archivo click-scroll.js, eliminar la función ready



Parte 3 - Menú

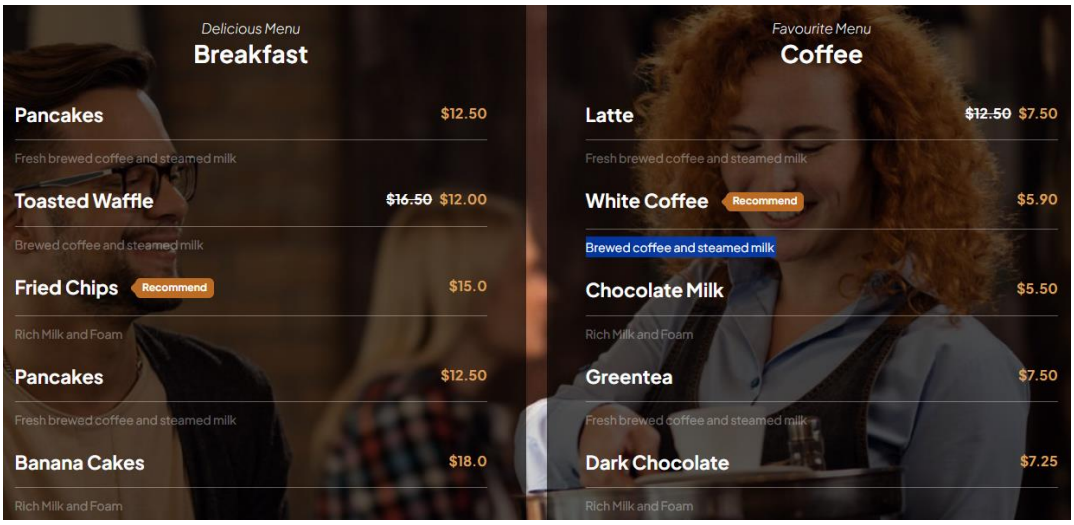
1. Crear la aplicación “menu”
2. Crear el modelo Menu
    a. title, description, order, active, created, updated

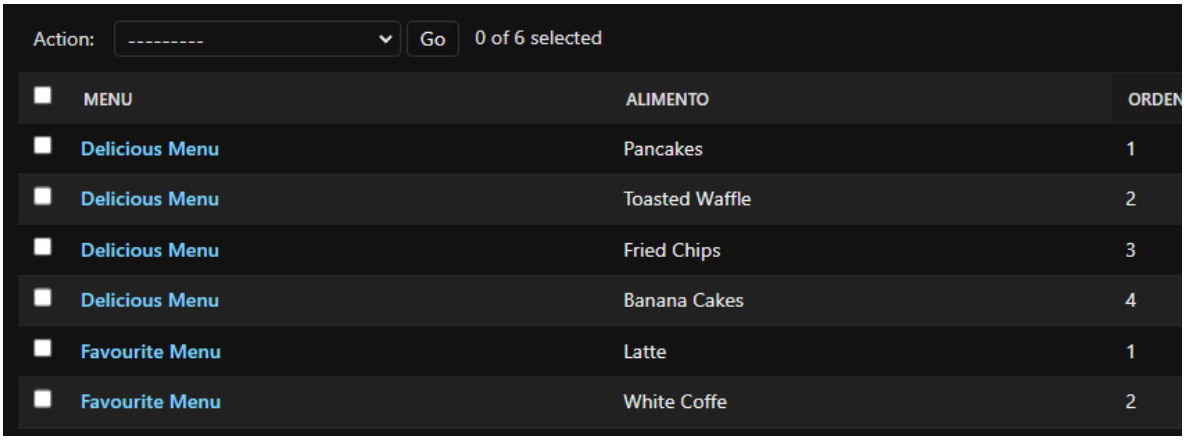
```python
class Menu(models.Model):
    title           = models.CharField(max_length=100, verbose_name='Título')
    description     = models.CharField(max_length=100, verbose_name='Descripción')
    order           = models.IntegerField(verbose_name="Orden de presentación")
    active          = models.BooleanField(verbose_name='Activo', default=True)
    created         = models.DateTimeField(auto_now_add=True, verbose_name="Fecha de creación")
    updated         = models.DateTimeField(auto_now=True, verbose_name="Fecha de actualización")

    You, 1 second ago | 1 author (You)
    class Meta:
        verbose_name        = 'Menu'
        verbose_name_plural = 'Menus'
        ordering            = ['order']

    def __str__(self):
        return self.title
```

3. Crear el modelo Food
    a. menu (FK), name, description, cost, previous_cost, recommended, created, updated

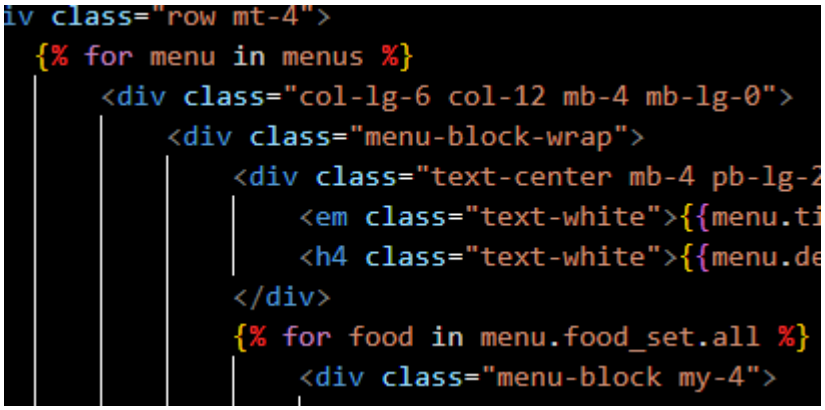
```python
class Food(models.Model):
    menu            = models.ForeignKey(Menu, verbose_name='Menu', on_delete=models.CASCADE)
    name            = models.CharField(max_length=100, verbose_name='Alimento')
    description     = models.CharField(max_length=100, verbose_name='Descripción')
    cost            = models.DecimalField(max_digits=4, decimal_places=2, verbose_name='Costo')
    previous_cost   = models.DecimalField(max_digits=4, decimal_places=2, verbose_name='Costo previo', null=True, blank=True)
    recommended     = models.BooleanField(verbose_name='Recomendado', default=False)
    order           = models.IntegerField(verbose_name="Orden de presentación")
    created         = models.DateTimeField(auto_now_add=True, verbose_name="Fecha de creación")
    updated         = models.DateTimeField(auto_now=True, verbose_name="Fecha de actualización")

    Francisco Vazquez, 2 hours ago | 1 author (Francisco Vazquez)
    class Meta:
        verbose_name        = 'Alimento'
        verbose_name_plural = 'Alimentos'
        ordering            = ['order']

    def __str__(self):
        return self.name
```

4. Crear la clase Admin y registrarlos para que el administrador de Django realice el CRUD de ambos modelos
5. Incluir la aplicación menu en el archivo settings
6. Realizar las migraciones correspondientes
7. Registrar al menos 2 menus y sus respectivos alimentos





8. Crear el método menu_list el cual debe regresar la lista de menus activos en el archivo views.py
9. Crear la estructura de templates\menu
10. Crear el template menu_list.html
    a. Se debe crear un ciclo para poder recorrer los menus y un ciclo interno para recorrer cada uno de los alimentos de cada menú.
    b. Para obtener los alimentos asociados a cada menú, puedes utilizar el método "_set.all" (como se muestra a continuación) o puedes utilizar un "related name" en el modelo correspondiente (revisa la práctica de webResturante)



```
iv class="row mt-4">
  {% for menu in menus %}
      <div class="col-lg-6 col-12 mb-4 mb-lg-0">
          <div class="menu-block-wrap">
              <div class="text-center mb-4 pb-lg-2
                  <em class="text-white">{{menu.ti
                  <h4 class="text-white">{{menu.de
              </div>
              {% for food in menu.food_set.all %}
                  <div class="menu-block my-4">
```

11. Crea el archivo urls.py para incluir el path de menu_list
12. Actualiza el archivo urls.py principal
13. Actualiza el template base.html para que acceda al menu
14. Verifica su funcionamiento