# Cátedra: Programación III / Diseño e software III – Segundo Parcial Práctico

**Tomando el caso de estudio del primer parcial, donde se debe desarrollar una aplicación web para gestión de correos electrónicos, donde en una primera etapa solo se implementarán las funciones de enviar correos, visualizar los correos enviados, y asignar uno o más correos a una categoría, se pide:**

1. Generar los modelos y relaciones necesarias para registrar información de emails enviados por el usuario logueado y las distintas categorías a las que luego se puede asignar un mail:
   1. Generar un modelo para registrar los datos de una “categoria”, que consta solo de un “nombre”.
   2. Generar otro modelo para registrar información de un “mail” enviado por el usuario logueado en el sistema, con los siguientes datos:
   * **usuario\_origen**, correspondiente al Usuario que está logueado en el sistema
   * **mail\_origen**, que corresponde al mail del Usuario que está logueado en el sistema
   * **mail\_destinatario,** correspondiente al mail destino de otra persona.
   * **asunto**, que sería el título del mail.
   * **cuerpo**, que corresponde al texto dentro del cuerpo del mail.
   * **fecha\_envio**, correspondiente a la fecha en que se envía el mail (debe ser por defecto la fecha de creación del registro).
   * **categoria,** que corresponde a un objeto del modelo anterior, en principio debe ser vacía y los registros de mail no se deben eliminar si se elimina una categoría (atributo **on\_delete**)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Modelos:

class Categoria(models.Model):  
 nombre = models.CharField(max\_length=255)  
  
 def \_\_str\_\_(self):  
 return self.nombre  
 pass

class Mail(models.Model):  
 usuario\_origen = models.ForeignKey(User, on\_delete=models.CASCADE)  
 mail\_origen = models.EmailField()  
 mail\_destinatario = models.EmailField()  
 asunto = models.CharField(max\_length=255)  
 cuerpo = models.TextField()  
 fecha\_envio = models.DateTimeField(auto\_now\_add=True)  
 categoria = models.ForeignKey(Categoria, on\_delete=models.SET\_NULL, null=True, blank=True)  
  
 def \_\_str\_\_(self):  
 return self.asunto  
 pass

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Completar la definición del método **“clean”** de la clase “MailForm” dentro del archivo “forms.py”. Se debe controlar aquí, que el campo “mail\_destinatario” no sea igual al “mail\_origen”

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class MailForm(forms.ModelForm):

class Meta:

model = Mail

fields= ('mail\_origen', 'mail\_destinatario', 'asunto', 'cuerpo')

def clean(self):  
 datos\_limpios = super().clean()  
 mail\_origen = datos\_limpios.get('mail\_origen')  
 mail\_destinatario = datos\_limpios.get('mail\_destinatario')  
  
 if mail\_origen and mail\_destinatario and mail\_origen == mail\_destinatario:  
 raise forms.ValidationError("El correo del destinatario no puede ser igual al correo del remitente.")  
  
 return datos\_limpios

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

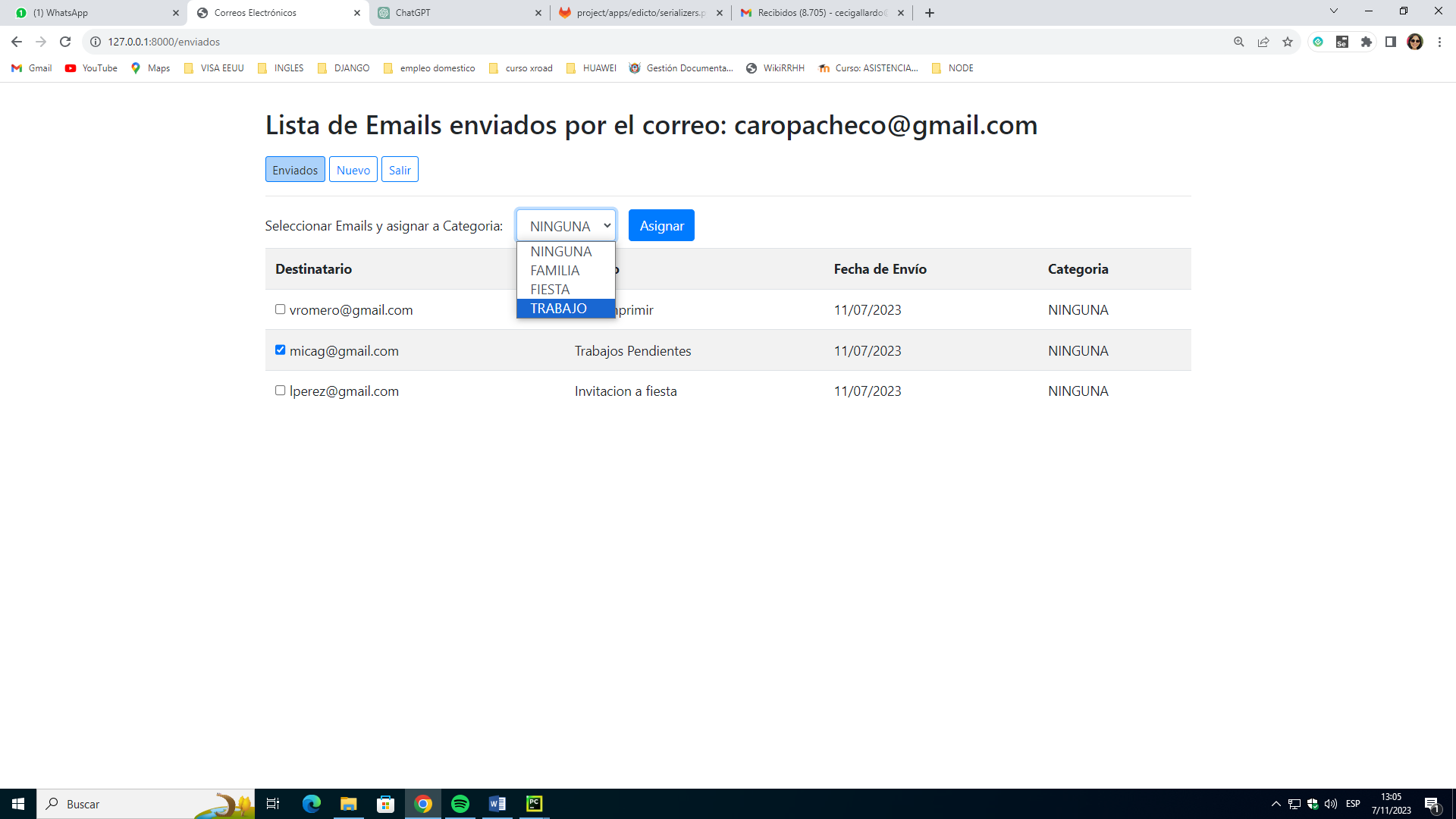
1. Completar la definición de la vista **“enviados”** dentro del archivo “views.py”, según lo indicado en el archivo mediante los comentarios seguidos a la leyenda “# Completar aquí”.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@login\_required  
def enviados(request):  
 mails\_enviados = Mail.objects.filter(usuario\_origen=request.user).order\_by('fecha\_envio')  
  
 categorias = Categoria.objects.all().order\_by('nombre')  
  
 return render(request, 'mail/enviados.html', {'mails\_enviados': mails\_enviados, 'categorias': categorias})  
 pass

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Completar la definición del template **“mail/enviados.html”** según lo indicado en el archivo mediante los comentarios seguidos a la leyenda “Completar aquí”. El resultado debe ser como el de la Figura 1:



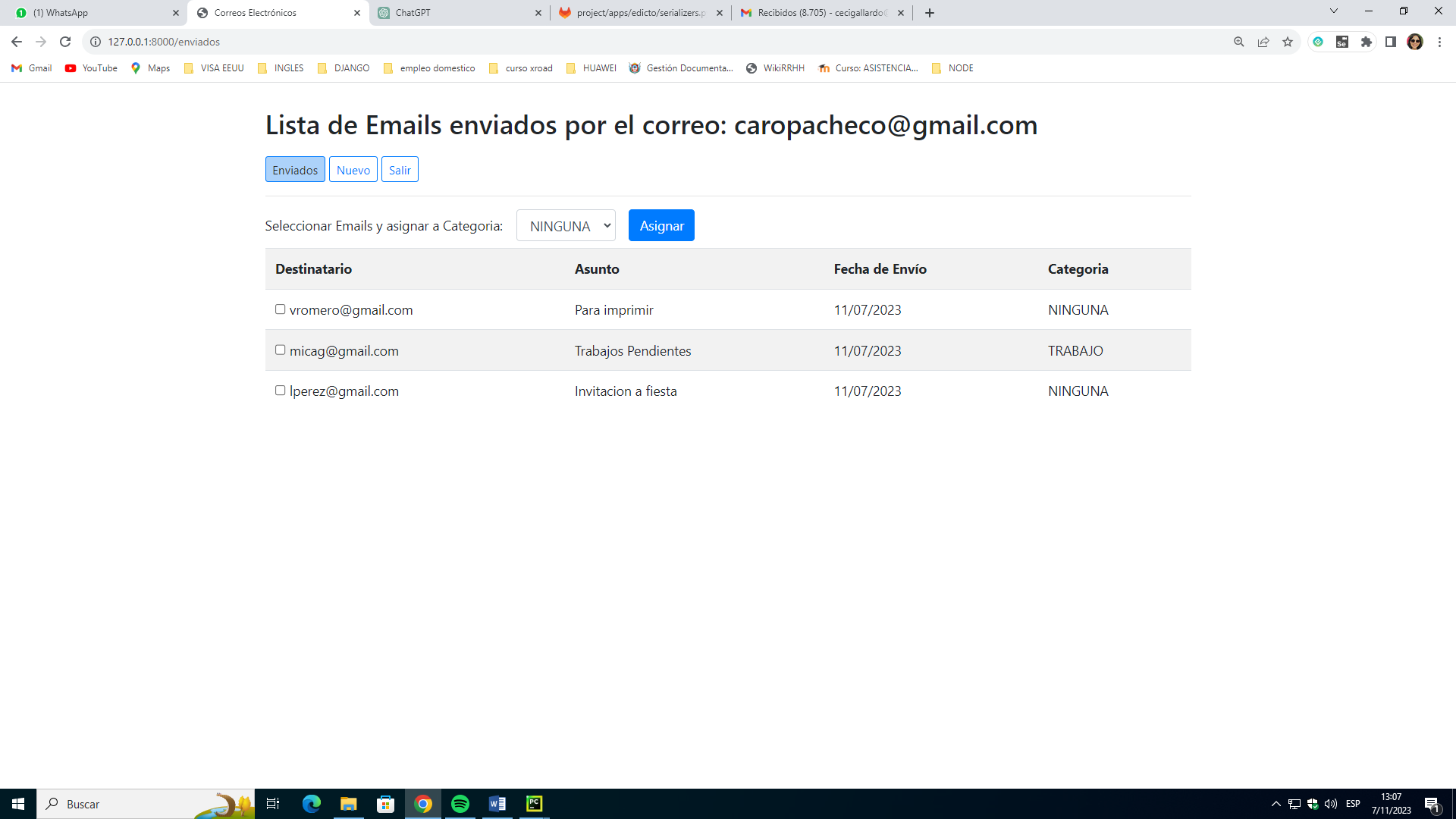
*Figura 1. Template que muestra lista de mails enviados por el usuario logueado*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<table class="table table-striped">  
 <tr>  
 <th>Destinatario</th>  
 <th>Asunto</th>  
 <th>Fecha de Envío</th>  
 <th>Categoria</th>  
 <th>Seleccionar</th>  
 </tr>  
 {% for mail in mails\_enviados %}  
 <tr>  
 <td><input type="checkbox" name="mail" value="{{ mail.id }}"></td>  
 <td>{{ mail.mail\_destinatario }}</td>  
 <td>{{ mail.asunto }}</td>  
 <td>{{ mail.fecha\_envio }}</td>  
 <td>{{ mail.categoria.nombre }}</td>  
 </tr>  
 {% empty %}  
 <tr>  
 <td colspan="5">No hay correos enviados.</td>  
 </tr>  
 {% endfor %}  
</table>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Completar la definición de la vista **“asignar\_categoria”** dentro del archivo “views.py”, según lo indicado en el archivo mediante los comentarios seguidos a la leyenda “# Completar aquí”. Una vez que se ejecute la vista mencionada, se debería volver a ver el template de la Figura 1, con el valor de categoría actualizado para los mails seleccionados, tal como la Figura 2:



*Figura 2. Lista de mails enviados donde se refresca el valor del campo “Categoría”*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@login\_required  
def asignar\_categoria(request):  
 if request.method == 'POST':  
 ids\_emails\_seleccionados = request.POST.getlist('mail')  
  
 categoria\_id = request.POST.get('categoria')  
  
 if categoria\_id == '0':   
 categoria = None  
 else:  
 categoria = Categoria.objects.get(pk=categoria\_id)  
  
 mails\_actualizados = Mail.objects.filter(pk\_\_in=ids\_emails\_seleccionados).update(categoria=categoria)  
  
 return redirect(' mail/enviados.html ')