Unit Test realizados en el proyecto Ryu Technology : Playground Final project

Se realizaron varios test en las diferentes Apps del proyecto

App users:

En esta app se realizó un test al modelo Profile:

```python
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    avatar = models.ImageField(upload_to='profiles', null = True, blank = True)
    bio = models.TextField(null = True, blank = True)
    link = models.URLField(max_length = 200, null = True, blank = True)
```

Se definieron tres funciones, **test_avatar_field**, **test_bio_field** y **test_link field**. En cada función se crea un usuario y una instancia del modelo **Profile**. En cada prueba se accede a los campos del modelo y se prueba la respuesta con los valores asignados

- **Test_avatar_field**: se usó el comando:

py manage.py test users.tests.ProfileModelTest.test_avatar_field

```python
def test_avatar_field(self):
    user = User.objects.create_user(username='testuser', password='testpassword')
    Profile.objects.create(user=user, avatar='profiles/no-avatar.jpg', bio='Test bio', link='http://www.google.com')
    profile = Profile.objects.get(id=1)
    field_label = profile._meta.get_field('avatar').verbose_name
    self.assertEqual(field_label, 'avatar')
```

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test users.tests.ProfileModelTest.test_avatar_field
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.178s

OK
Destroying test database for alias 'default'...
```

La primera prueba con los datos esperados de user, avatar, bio y link la prueba sale correcta

Para la segunda prueba de avatar, se cambió el avatar por un número (avatar es un campo ImageField), la prueba

```python
def test_avatar_field(self):
    user = User.objects.create_user(username='testuser', password='')
    Profile.objects.create(user=user, avatar=12345, bio='Test bio', link='http://www.google.com')
    profile = Profile.objects.get(id=1)
    field_label = profile._meta.get_field('avatar').verbose_name
    self.assertEqual(field_label, 'avatar')
```

Esto genera un error en la prueba:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Creating test database for alias 'default'...
System check identified no issues (0 silenced).
E
======================================================================
ERROR: test_avatar_field (users.tests.ProfileModelTest.test_avatar_field)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\users\tests.py", line 9, in test_avatar_field
    Profile.objects.create(user=user, avatar=12345, bio='Test bio', link='http://www.google.com')
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\manager.py", line 87, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\query.py", line 658, in create
    obj.save(force_insert=True, using=self.db)
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\base.py", line 814, in save
    self.save_base(
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\base.py", line 877, in save_base
    updated = self._save_table(
              ^^^^^^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\base.py", line 1020, in _save_table
    results = self._do_insert(
              ^^^^^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\base.py", line 1061, in _do_insert
    return manager._insert(
           ^^^^^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\manager.py", line 87, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\query.py", line 1805, in _insert
    return query.get_compiler(using=using).execute_sql(returning_fields)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\sql\compiler.py", line 1819, in execute_sql
    for sql, params in self.as_sql():
                       ^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\sql\compiler.py", line 1743, in as_sql
    value_rows = [
                 ^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\sql\compiler.py", line 1744, in <listcomp>
    [
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\sql\compiler.py", line 1745, in <listcomp>
    self.prepare_value(field, self.pre_save_val(field, obj))
                               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\sql\compiler.py", line 1693, in pre_save_val
    return field.pre_save(obj, add=True)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\venv\Lib\site-packages\django\db\models\fields\files.py", line 315, in pre_save
    if file and not file._committed:
                    ^^^^^^^^^^^^^^^
AttributeError: 'int' object has no attribute '_committed'

----------------------------------------------------------------------
Ran 1 test in 0.209s

FAILED (errors=1)
Destroying test database for alias 'default'...
```

```
AttributeError: 'int' object has no attribute '_committed'

----------------------------------------------------------------------
Ran 1 test in 0.209s

FAILED (errors=1)
Destroying test database for alias 'default'...
```

- **Test_bios_field**: se usó el comando:

py manage.py test users.tests.ProfileModelTest.test_bio_field

Para la primera prueba, con los campos correctos la prueba pasa sin errores:

```python
def test_bio_field(self):
    user = User.objects.create_user(username='testuser', password='testpassword')
    Profile.objects.create(user=user, avatar='profiles/no-avatar.jpg', bio='Test bio', link='http://www.google.com')
    profile = Profile.objects.get(id=1)
    field_label = profile._meta.get_field('bio').verbose_name
    self.assertEqual(field_label, 'bio')
```

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test users.tests.ProfileModelTest.test_bio_field
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.177s

OK
Destroying test database for alias 'default'...
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final>
```

Para la segunda prueba, se deja el campo **bio** vacío, la prueba pasa normal. Esto es debido a que el campo está configurado con Null=True y Blank =True, debido a que es un campo opcional, el usuario se crea sin estos valores

```python
def test_avatar_field(self):
    user = User.objects.create_user(username='testuser', password='testpassword')
    Profile.objects.create(user=user, avatar='profiles/no-avatar.jpg', bio='', link='http://www.google.com')
    profile = Profile
    field_label = pro                                   ).verbose_name
    self.assertEqual(field_label, 'avatar')
```

```
(variable) field_label: str
```

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test users.tests.ProfileModelTest.test_bio_field
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.181s

OK
Destroying test database for alias 'default'...
```

- **Test_link_field**: se usó el comando:

py manage.py test users.tests.ProfileModelTest.test_link_field

```python
def test_link_field(self):
    user = User.objects.create_user(username='testuser', password='testpassword')
    Profile.objects.create(user=user, avatar='profiles/no-avatar.jpg', bio='Test bio', link='http://www.google.com')
    profile = Profile.objects.get(id=1)
    field_label = profile._meta.get_field('link').verbose_name
    self.assertEqual(field_label, 'link')
```

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test users.tests.ProfileModelTest.test_link_field
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.179s

OK
Destroying test database for alias 'default'...
```

Con todos los campos correspondientes la prueba da bien. Igualmente, si se varía, también pasa la prueba, por la misma razón que acepta campos Null y Blank:

```python
def test_link_field(self):
    user = User.objects.create_user(username='testuser', password='testpassword')
    Profile.objects.create(user=user, avatar='profiles/no-avatar.jpg', bio='Test bio', link='www.google.com')
    profile = Profile.objects.get(id=1)
    field_label = profile._meta.get_field('link').verbose_name
    self.assertEqual(field_label, 'link')
```

```python
def test_link_field(self):
    user = User.objects.create_user(username='testuser', password='testpassword')
    Profile.objects.create(user=user, avatar='profiles/no-avatar.jpg', bio='Test bio', link=888)
    profile = Profile.objects.get(id=1)
    field_label = profile._meta.get_field('link').verbose_name
    self.assertEqual(field_label, 'link')
```

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test users.tests.ProfileModelTest.test_link_field
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.180s

OK
Destroying test database for alias 'default'...
```

App blog:

Para los test del app blog, se realizaron sobre la función **createPage:**

```
def createPage(request):


    if(request.user.has_perm('blog.can_edit') or
request.user.has_perm('blog.can_delete') ):

        if(request.method == "POST"):

            form = PostsForm(request.POST,request.FILES)
            print(form)
            if(form.is_valid()):


                info = form.cleaned_data
                print(info)
                usertable= User.objects.get(username=request.user.username)
                image = request.FILES['imageMain']

                blog = Posts.objects.create(user=usertable, title = info['title'],
subtitle=info['subtitle'], imageMain = image, Message = info['Message'])
                blog.save()

                return render(request, 'blog/newpage.html',{
                'form': PostsForm})
            else:
                return render(request, 'blog/newpage.html',{
                'form': PostsForm, 'errors':form.errors})


        return render(request, 'blog/newpage.html',{
            'form': PostsForm
        })

    else:

        messages.error(request, 'No tienes permisos para realizar esta operación')
        return redirect('pages')
```

Se creó una clase llamada **CreatePasgeTest** y se usó el commando:

py manage.py test blog.tests.CreatePageTest.test_create_page_with_permission

- **test_create_page_with_permission:**

```python
class CreatePageTest(TestCase):

    def test_create_page_with_permission(self):

        self.user = User.objects.create_user(username='userprueba', password='password_123')
        self.client = Client()

        self.user.user_permissions.add(35) #De acuerdo al DB , Blog.Can_Edit tiene id=35
        self.client.login(username='userprueba', password='password_123')

        # Petición POST
        form_data = {'title':'Test Titulo','subtitle':'Test subtitulo',
                     'Message': 'Esto es una prueba',
                     'imageMain': SimpleUploadedFile("test_image.jpg", b"file_content", content_type="image/jpeg"),
                     'dateAdded' : '2023-05-18 04:29:48.571511','dateModified': '2023-05-19 05:46:33.153497'}

        Posts.objects.create(title='Test Titulo', subtitle='Test subtitulo',
                             Message='Esto es una prueba', imageMain='tardigrado.webp', user=self.user)

        #Simulación del Post
        response = self.client.post('/pages/create/', form_data)

        # Verificar que se haya creado correctamente
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'blog/newpage.html')

        # Verificar que se haya guardado en la DB
        created_page = Posts.objects.get(title='Test Titulo')
        self.assertEqual(created_page.subtitle, 'Test subtitulo')
```

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test blog.tests.CreatePageTest.test_create_page_with_permission
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
<tr>
    <th><label for="id_title">Título:</label></th>
    <td>

      <input type="text" name="title" value="Test Titulo" class="form-control" maxlength="100" required id="id_title">


    </td>
  </tr>

  <tr>
    <th><label for="id_subtitle">Descripción:</label></th>
    <td>

      <input type="text" name="subtitle" value="Test subtitulo" class="form-control" maxlength="100" required id="id_subtitle">


    </td>
  </tr>

  <tr>
    <th><label for="id_Message">Message:</label></th>
    <td>

      <div class="django-ckeditor-widget" data-field-id="id_Message" style="display: inline-block;">
    <textarea name="Message" cols="40" rows="10" required id="id_Message" data-processed="0" data-config="{&quot;skin&quot;: &quot;moono-lisa&quot;, &quot;toolb
ce&quot;, &quot;-&quot;, &quot;Bold&quot;, &quot;Italic&quot;]], &quot;toolbar_Full&quot;: [[&quot;Styles&quot;, &quot;Format&quot;, &quot;Bold&quot;, &quot;Ita
uot;, &quot;Strike&quot;, &quot;SpellChecker&quot;, &quot;Undo&quot;, &quot;Redo&quot;], [&quot;Link&quot;, &quot;Unlink&quot;, &quot;Anchor&quot;], [&quot;Imag
&quot;Table&quot;, &quot;HorizontalRule&quot;], [&quot;TextColor&quot;, &quot;BGColor&quot;], [&quot;Smiley&quot;, &quot;SpecialChar&quot;], [&quot;Source&quot
ot;full&quot;, &quot;height&quot;: 300, &quot;width&quot;: 900, &quot;filebrowserWindowWidth&quot;: 940, &quot;filebrowserWindowHeight&quot;: 725, &quot;langua
ta-external-plugin-resources="[]" data-id="id_Message" data-type="ckeditortype">Esto es una prueba</textarea>
</div>


    </td>
  </tr>

  <tr>
    <th><label for="id_imageMain">Imagen:</label></th>
    <td>
      <ul class="errorlist"><li>Envíe una imagen válida. El fichero que ha enviado no era una imagen o se trataba de una imagen corrupta.</li></ul>
      <input type="file" name="imageMain" class="form-control" accept="image/*" required id="id_imageMain">
```

```
    </td>
  </tr>
.
----------------------------------------------------------------------
Ran 1 test in 0.462s

OK
Destroying test database for alias 'default'...
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final>
```

Para esta prueba, se emulo agregar un post en la página, por lo que se necesitó crear un usuario, loguearlo y asignarle un permiso de edición/borrado, se crea una instancia de la clase Posts que se verifica con el método **assertEqual** y también se simula un método post con **client.post** . Para esta primera prueba los resultados son buenos.

En la siguiente prueba cambiando el valor del action de post genera error, debido a que se espera una respuesta código 200 y se obtiene una 404.

```python
class CreatePageTest(TestCase):

    def test_create_page_with_permission(self):

        self.user = User.objects.create_user(username='userprueba', password='password_123')
        self.client = Client()

        self.user.user_permissions.add(35) #De acuerdo al DB , Blog.Can_Edit tiene id=35
        self.client.login(username='userprueba', password='password_123')

        # Petición POST
        form_data = {'title':'Test Titulo','subtitle':'Test subtitulo',
                     'Message': 'Esto es una prueba',
                     'imageMain': SimpleUploadedFile("test_image.jpg", b"file_content", content_type="image/jpeg"),
                     'dateAdded' : '2023-05-18 04:29:48.571511','dateModified': '2023-05-19 05:46:33.153497'}

        Posts.objects.create(title='Test Titulo', subtitle='Test subtitulo',
                             Message='Esto es una prueba', imageMain='tardigrado.webp', user=self.user)

        #Simulación del Post
        response = self.client.post('/pages-create/', form_data)

        # Verificar que se haya creado correctamente
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'blog/newpage.html')

        # Verificar que se haya guardado en la DB
        created_page = Posts.objects.get(title='Test Titulo')
        self.assertEqual(created_page.subtitle, 'Test subtitulo')
```

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test blog.tests.CreatePageTest.test_create_page_with_permission
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
F
======================================================================
FAIL: test_create_page_with_permission (blog.tests.CreatePageTest.test_create_page_with_permission)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\blog\tests.py", line 30, in test_create_page_with_permission
    self.assertEqual(response.status_code, 200)
AssertionError: 404 != 200

----------------------------------------------------------------------
Ran 1 test in 0.370s

FAILED (failures=1)
Destroying test database for alias 'default'...
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final>
```

- **Test_create_page_without_permission:** se llevo a cabo con:

py manage.py test blog.tests.CreatePageTest.test_create_page_without_permission

```python
def test_create_page_without_permission(self):

    self.user = User.objects.create_user(username='userprueba', password='password_123')
    self.client = Client()

    #self.user.user_permissions.add(35) #De acuerdo al DB , Blog.Can_Edit tiene id=35
    self.client.login(username='userprueba', password='password_123')

    # Simular una petición POST sin el permiso necesario
    form_data = {'title':'Test Titulo','subtitle':'Test subtitulo',
                 'Message': 'Esto es una prueba','imageMain':'<InMemoryUploadedFile: tardigrado.webp (image/webp)>',
                 'dateAdded' : '2023-05-18 04:29:48.571511','dateModified': '2023-05-19 05:46:33.153497'}

    response = self.client.post('/pages/create/', form_data)

    # Verificar que la redirección y el mensaje de error se hayan realizado correctamente
    self.assertEqual(response.status_code, 302)
    self.assertRedirects(response, '/pages/')

    messages = list(get_messages(response.wsgi_request))
    self.assertEqual(len(messages), 1)
    self.assertEqual(str(messages[0]), 'No tienes permisos para realizar esta operación')

    # Verificar que no se haya creado ningún objeto de página en la base de datos
    self.assertRaises(Posts.DoesNotExist, Posts.objects.get, title='Test Titulo')
```

Esta prueba se realiza sin permisos de usuario (están comentados) y consiste en simular un POST esperando el error (302) y el redireccionamiento. La prueba da correcta porque no se puede ejecutar el post y no se almacena nada en la base de datos.

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test blog.tests.CreatePageTest.test_create_page_without_permission
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.379s

OK
Destroying test database for alias 'default'...
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final>
```

Para el siguiente test y se le quitó el comentario a los permisos, la prueba da error porque el post da respuesta 200 y espera error

```python
def test_create_page_without_permission(self):

    self.user = User.objects.create_user(username='userprueba', password='password_123')
    self.client = Client()

    self.user.user_permissions.add(35) #De acuerdo al DB , Blog.Can_Edit tiene id=35
    self.client.login(username='userprueba', password='password_123')

    # Simular una petición POST sin el permiso necesario
    form_data = {'title':'Test Titulo','subtitle':'Test subtitulo',
                 'Message': 'Esto es una prueba','imageMain':'<InMemoryUploadedFile: tardigrado.webp (image/webp)>',
                 'dateAdded' : '2023-05-18 04:29:48.571511','dateModified': '2023-05-19 05:46:33.153497'}

    response = self.client.post('/pages/create/', form_data)

    # Verificar que la redirección y el mensaje de error se hayan realizado correctamente
    self.assertEqual(response.status_code, 302)
    self.assertRedirects(response, '/pages/')

    messages = list(get_messages(response.wsgi_request))
    self.assertEqual(len(messages), 1)
    self.assertEqual(str(messages[0]), 'No tienes permisos para realizar esta operación')

    # Verificar que no se haya creado ningún objeto de página en la base de datos
    self.assertRaises(Posts.DoesNotExist, Posts.objects.get, title='Test Titulo')
```

```
======================================================================
FAIL: test_create_page_without_permission (blog.tests.CreatePageTest.test_create_page_without_permission)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final\blog\tests.py", line 55, in test_create_page_without_permission
    self.assertEqual(response.status_code, 302)
AssertionError: 200 != 302

----------------------------------------------------------------------
Ran 1 test in 0.416s

FAILED (failures=1)
Destroying test database for alias 'default'...
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final>
```

App Mensajeria

- Test en Mensajeria: se ejecutó sobre la función **sendMessage** de la app **mensajeria**

```
@login_required
def sendMessage(request):
    from django.db.models import Q
    user = request.user

    if (request.method == 'POST'):
        message = request.POST['message']
        receiver_id = request.POST['receiver']
        receiver = User.objects.get(id=receiver_id)
        message = MessagesChat.objects.create(sender=request.user, receiver=receiver,
message=message)

        return redirect('chatroom-getmsgs',  id=receiver_id)
    else:
        return HttpResponse("Invalid request method.")
```

py manage.py test mensajeria.tests.SendMessageTest.test_send_message

```
class SendMessageTest(TestCase):


    def test_send_message(self):

        self.client = Client()
        self.user1 = User.objects.create_user(username='user1', password='testpassword_1')
        self.user2 = User.objects.create_user(username='user2', password='testpassword_2')

        # Iniciar sesión como user1
        self.client.login(username='user1', password='testpassword_1')

        # Simular POST
        postSimulado = {'message': 'mensaje de prueba','receiver': self.user2.pk}
        response = self.client.post('/mensajeria/chatRoom/receive'.format(self.user2.pk), postSimulado)

        # Verificar  redirección a la vista del chat y el código de respuesta
        self.assertRedirects(response, reverse('chatroom-getmsgs', kwargs={'id': self.user2.pk}))
        self.assertEqual(response.status_code, 302)

        # Aqui se chequea si se creo el mensaje en la DB
        messages = MessagesChat.objects.filter(sender=self.user1, receiver=self.user2, message='mensaje de prueba')
        self.assertEqual(messages.count(), 1)
```

En esta prueba se crearon dos usuarios y se emuló el envío de un mensaje de uno a otro, con una simulación de una respuesta post y un redireccionamiento. Se verificó igualmente en el DB que se haya guardado la prueba. Los resultados fueron correctos.

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test mensajeria.tests.SendMessageTest.test_send_message
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.551s

OK
Destroying test database for alias 'default'...
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final>
```

- **Test_invalid_request_method**: La otra función del test consistió en sustituir la simulación de la prueba POST por una prueba GET

```python
def test_invalid_request_method(self):

    self.client = Client()
    self.user1 = User.objects.create_user(username='user1', password='testpassword_1')
    self.user2 = User.objects.create_user(username='user2', password='testpassword_2')

    # Iniciar sesión como user1
    self.client.login(username='user1', password='testpassword_1')

    # Simular una petición GET en lugar de una petición POST
    response = self.client.get(reverse('chatroom-sendmsgs'))

    # Verificar que se reciba una respuesta de "Invalid request method."
    self.assertEqual(response.status_code, 200)
    self.assertEqual(response.content.decode('utf-8'), 'Invalid request method.')
```

Resultados esperados son correctos:

```
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final> py manage.py test mensajeria.tests.SendMessageTest.test_invalid_request_method
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.542s

OK
Destroying test database for alias 'default'...
(venv) PS D:\Users\luiggi_marquez\Desktop\Coder\Python\Desafios\proyecto_final>
```

Cada uno de estos tests se realizaron en el archivo tests.py en la carpeta de cada aplicación (users,blog, mensajeria)