En este trabajo practico desarrollamos una página web utilizando Django. El objetivo fue implementar distintas funcionalidades relacionadas al registro de usuarios, visualización de contenido y demás. A lo largo del desarrollo se trabajó en equipo, distribuyendo las tareas y haciendo uso de GitHub. A continuación, se detallan las funciones implementadas y respectivas explicaciones.  
  
**services.getAllImages:** Esta función se encarga de conseguir la información de los pokemones mediante la función getAllImages en transport.py, para luego devolverlos en forma de lista. Primero se intentó desarrollar la función para obtener los datos desde la misma, pero luego se la corrigió para que usara la función presente en el archivo transport.py.

def getAllImages():

json\_collection = transport.getAllImages()

cards = []

for pokemon in json\_collection:

card = translator.fromRequestIntoCard(pokemon)

cards.append(card)

return cards

**forms.subscribeForm:** Define los campos que deben completarse en el formulario al momento de registrarse.

class SubscribeForm(forms.Form):

username = forms.CharField()

name = forms.CharField()

surname = forms.CharField()

password = forms.CharField()

email = forms.EmailField()

**views.subscribe:** Toma los datos del formulario y si son válidos llama a la función register\_user, si hubo algún problema se le muestra al usuario por pantalla, y sino se le envía un mail con las credenciales de inicio de sesión.

def subscribe(request):

form = SubscribeForm()

if request.method == 'POST':

form = SubscribeForm(request.POST)

if form.is\_valid():

usuario = form.cleaned\_data

errores = register\_user(form.cleaned\_data)

if errores:

for error in errores:

messages.error(request,error)

else:

username = usuario['username']

email = usuario['email']

password = usuario['password']

subject = 'Registro exitoso'

message = f'¡Gracias por registrarte {username}!\nEstas son tus credenciales de inicio de sesión:\nUsuario:{username}\nContraseña:{password}'

send\_mail(subject, message, settings.EMAIL\_HOST\_USER, [email], fail\_silently=False)

return redirect('loading\_home')

return render(request, 'registration/register.html', {'form': form})

**services.register\_user:** Esta función es la encargada de la creación de nuevas cuentas, si tanto el nombre de usuario como la dirección de correo electrónico están disponibles guarda los datos, y sino devuelve los errores.

def register\_user(usuario):

username = usuario['username']

email = usuario['email']

password = usuario['password']

name = usuario['name']

surname = usuario['surname']

errores = []

if User.objects.filter(username = username).exists():

errores.append("Ese nombre de usuario ya está en uso.")

if User.objects.filter(email = email).exists():

errores.append("Esa dirección de correo electrónico ya está asociada a otra cuenta.")

if errores:

return errores

User.objects.create\_user(

username = username,

email = email,

password = password,

first\_name = name,

last\_name = surname,

)

return errores

Además, se agregó un template utilizado para el registro de usuarios y se hicieron cambios en el archivo main/settings.py para implementar el envío de mails al momento de registrar la cuenta.