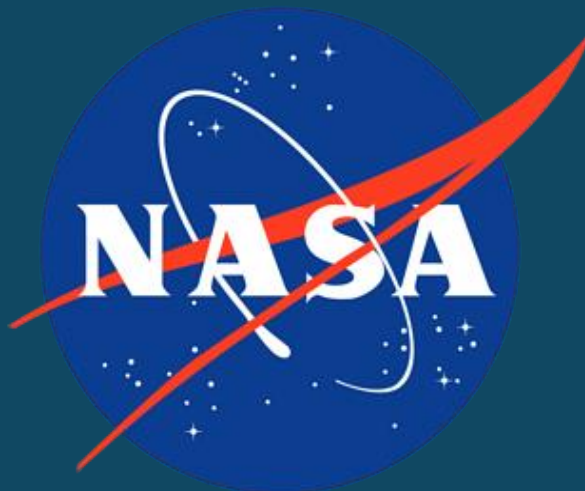




TRABAJO PRACTICO INFORME DE CÓDIGO



Índice:

1. Consiga y generalidades del trabajo.
2. Mapa conceptual de carpetas y funciones.
3. Detalle de funciones implementadas.
4. Conclusiones

Leti Angel, Scopa Juan, Colman Victor
COMISIÓN 10

1. Consignas y generalidades del trabajo.

El trabajo consiste en implementar una aplicación web fullstack usando Django Framework que permita consultar las imágenes de la API pública que proporciona la NASA. La información que provenga de esta API será renderizada por el framework en distintas cards que mostrarán -como mínimo- la imagen en cuestión, un título y una descripción.

Como opcional se puede realizar: Buscador, login, agregar a favoritos, visualizar favoritos y eliminarlos, paginación o scroll, agregar comentarios a imágenes, alta de usuarios y mejorar la visual.

En este trabajo decidimos implementar los adicionales de: Buscador, login, agregar a favoritos, visualizarlos y eliminarlos, paginación y mejora de la visual.

El proyecto contenía funciones que ya estaban implementadas y funcionando y otras que se debían completar.

Template: Se cuenta con 6 HTMLs: header (cabecera de la página), footer (pie de página), home (sección donde se mostrarán las imágenes y el buscador), index (contenedor principal que incluye a los 3 HTMLs anteriores), favourites (Contiene tanto el apartado para la lista de favoritos como para el botón de agregar o quitar los mismos) y login (Contiene el apartado para la página del login). Para el caso del header, se implementó cierta lógica para determinar si un usuario está logueado (o no) y obtener así su nombre; para el caso del home, éste tiene un algoritmo que permite recorrer cada objeto de la API y dibujar su información en pantalla

Views: En el archivo views.py encontrarán algunas funciones semidesarrolladas: `index_page(request)` que renderiza el contenido de 'index.html'; `home(request)` que obtiene todas las imágenes mapeadas de la API -a través de la capa de servicio- y los favoritos del usuario, y muestra el contenido de 'home.html' pasándole dicha información. Esta última hace uso de la función auxiliar `getAllImagesAndFavouriteList(request)` que devuelve 2 listas: una de las imágenes de la API y otra de las imágenes marcadas como favoritos del usuario.

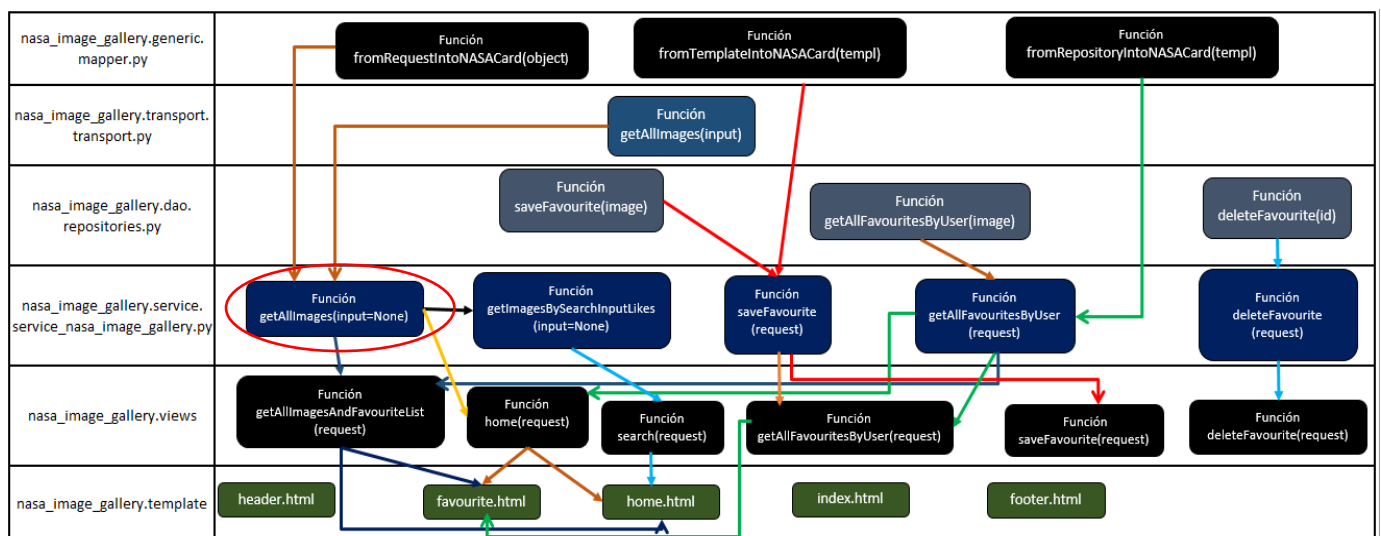
En cuanto a la lógica del se incluye el archivo *transport.py* completo con todo el código necesario para consumir la API. Además, se anexa un *mapper.py* con las funciones necesaria para convertir/mapear los resultados en una NASACard (objeto que finalmente se utilizará en el template para dibujar los resultados).

2. Mapa conceptual de carpetas y funciones:

En la siguiente imagen se observa la división de las carpetas del proyecto con las funciones que se implementaron en cada una, unidas por medio de flechas aquellas que se conectan.

Por ejemplo, la función `getAllImages(input=None)` (marcada en la imagen en un círculo rojo) ubicada en `service_nasa_image_gallery.py` en su composición trae a la función `fromRequestIntoNASACard(object)` de `mapper.py` y a la función `getAllImages(input)` de `transport.py`. y, su vez, es llamada en las funciones `getImagesBySearchInputLikes(input=None)` de su misma carpeta, en `getAllImagesAndFavouriteList(request)` de `views` y en la función `home` también en la `views.py`

Esta imagen es una primera vista a la composición del proyecto, la cual se termina de completar en el siguiente punto del índice.



3. Detalle de funciones implementadas: A continuación, se mostrarán una imagen de cada función desarrollada con una breve descripción.

Funciones de [services_nasa_image_gallery.py](#)

getAllImages(input=None):

```
def getAllImages(input=None):
    json_collection = transport.getAllImages(input)
    images = []
    for json in json_collection: # recorre el listado de objetos JSON, Lo transforma en un
        try: # se agrega un try para que en caso de que falte algun item en la creación de
            imagen=mapper.fromRequestIntoNASACard(json)
            if imagen.title and imagen.description and imagen.image_url and imagen.date:
                images.append(imagen)
            else:
                return redirect('home')
        except Exception as e:
            print(f"Error al procesar La imagen: {e}")
    return images
```

En esta función se agregó inicialmente una definición de variable (json_collection) la cual trae la función `getAllImages` contenida en `transport.py`. Luego se realiza un for por el json_collection en la que se genera la variable imagen que genera la NasaCard utilizando la función `fromRequestIntoNASACard` desde `mapper.py`, esa imagen generada es agregada a la lista previamente definida "images". Con solo eso el código cumplía su función, luego en pruebas el equipo noto que en la búsqueda de la palabra "artemis" se generaba un error debido a que una imagen en esa búsqueda no contiene descripción, por lo cual se agrega un try para que, dado un error en alguno de los elementos necesario para la creación de la NASACard la misma se genere igual y otorgue un mensaje de error con el ítem que falto procesar (e)

SaveFavourite(request):

```
# añadir favoritos (usado desde el template 'home.html')
def saveFavourite(request):
    fav = mapper.fromTemplateIntoNASACard(request) # transformamos un request del template en una NASACard.
    fav.user = get_user(request) # Le seteamos el usuario correspondiente.

    return repositories.saveFavourite(fav) # Lo guardamos en la base.
```

Función utilizada para guardar favoritos, la variable "fav" es un resultado de la función `fromTemplateIntoNASACard(request)`, se agrega el usuario con el "get" y luego se llama a la función `saveFavourite` del repositorio a la cual se le da el argumento "fav"

getAllFavouritesByUser(request):

```
# usados en el template 'favourites.html'
def getAllFavouritesByUser(request):
    if not request.user.is_authenticated: # si el usuario no esta registrado
        return [] # retorna lista vacía
    else:
        user=get_user(request) # se asocia el usuario desde el request con la funcion GET
        favourite_list=repositories.getAllFavouritesByUser(user) # Se genera la lista de usuarios llamando a la funcion desde el repositorio

        mapped_favourites=[] # se abre una lista vacía donde se colocara la nasa_card

        for favourite in favourite_list: #for para recorrer la lista de favoritos
            nasa_card=mapper.fromRepositoryIntoNASACard(favourite) #se genera la nasa_card desde el repositorio
            mapped_favourites.append(nasa_card) # se agrega la nasa_card del favorito en la lista

        return mapped_favourites #La lista de favoritos en formato nasa_card
```

Esta función revisa si el usuario esta autenticado, en caso de no estarlo, retorna una lista vacía, en caso de que si lo este, se genera la variable favourite_list, con la función *getAllFavouritesByUser* traída desde **repositories.py**.

Se abre lista vacía "mapped_favourites" y se realiza un for de cada favorito en la lista generando una nasa_card por cada uno y se agrega a la lista mapped_favourites.

Retorna mapped_favourites.

deleteFavourite(request):

```
def deleteFavourite(request):
    favId = request.POST.get('id')
    return repositories.deleteFavourite(favId) # borramos un favorito por su ID.
```

Función que agrega un ID al favorito para otorgarlo como argumento de la función **deleteFavourite(id)** que se encuentra en el repositorio.

Funciones de [views.py](#)

getAllImagesAndFavouriteList(request):

```
# auxiliar: retorna 2 listados -> uno de las imágenes de la API y otro de los favoritos del usuario.
def getAllImagesAndFavouriteList(request):
    favourite_list = services_nasa_image_gallery.getAllFavouritesByUser(request) # llama a la función solo para traer los favoritos
    images = services_nasa_image_gallery.getAllImages() # llama a la función que trae todas las imágenes

    paginator = Paginator(images, 5) # Mostrar 5 imágenes por página
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    return render(request, 'home.html', {'page_obj': page_obj, 'favourite_list': favourite_list})
```

Función que trae todas las imágenes generadas, utilizando dos funciones realizadas en [services_nasa_image_gallery.py](#) ([getAllFavouritesByUser](#) y [getAllImages](#)). Se agregó el llamado a la función [getAllImages](#) que crea dos listas: “images” que son los resultados de la función [getAllImages](#) y “favourite_list” que toma la lista de favoritos desde [getAllFavouritesByUser](#). Se sumó la función para el paginado, esta función organiza las imágenes en páginas y muestra la página correcta según la solicitud del usuario. Además, maneja los posibles errores, asegurándose de que siempre se muestre una página válida, ya sea la primera o la última, este “try” lo agregamos ya que nos generaba el error que la page no era un int.

home(request):

```
# función principal de la galería.
def home(request):
    images = services_nasa_image_gallery.getAllImages()
    favourite_list = services_nasa_image_gallery.getAllFavouritesByUser(request)

    images_per_page = int(request.GET.get('per_page', 5)) # Default 5 imágenes por página
    paginator = Paginator(images, images_per_page)
    page_number = request.GET.get('page')
    try:
        page_obj = paginator.page(page_number)
    except PageNotAnInteger:
        page_obj = paginator.page(1)
    except EmptyPage:
        page_obj = paginator.page(paginator.num_pages)

    return render(request, 'home.html', {'page_obj': page_obj, 'favourite_list': favourite_list, 'images_per_page': images_per_page})
```

Función que renderiza el template 'home.html', se agregó el llamado a [getAllImages](#) y [getAllImagesAndFavouriteList](#) definiendo las variables, images y favourite_list, respectivamente. Se agregó la función "Paginator" que genera la vista de las páginas totales y permite navegar entre ellas, también permite al usuario elegir la cantidad de imágenes por página.

search(request):

```
def search(request):
    search_msg = request.GET.get('query', '')
    images = []

    if search_msg:
        images = services_nasa_image_gallery.getImagesBySearchInputLike(search_msg)
    else:
        images = services_nasa_image_gallery.getAllImages(None)
        #CODIGO PARA COLOCAR PAGINAS EN LA APP
    images_per_page = int(request.GET.get('per_page', 5)) # avariable per_page que proviene del url

    paginator = Paginator(images, images_per_page) # Funcion paginatos images=total de imagenes, images_per_page= imagenes por pagina
    page_number = request.GET.get('page')

    try:
        page_obj = paginator.page(page_number)
    except PageNotAnInteger: #Error en caso de que no sea un entero
        page_obj = paginator.page(1) #Ir a pagina 1
    except EmptyPage: # En caso de que la pagina este vacía
        page_obj = paginator.page(paginator.num_pages)

    return render(request, 'home.html', {'page_obj': page_obj, 'query': search_msg, 'images_per_page': images_per_page})
```

Se agrega condicionante para el buscador en caso de que contenga o no un input. Si no hay escritura se llama la función `services_nasa_image_gallery.getAllImages()`, caso contrario `services_nasa_image_gallery.getImagesBySearchInputLike(search_msg)`. Se agrega la funcion `paginator`.

getAllFavouritesByUser(request):

```
@login_required
def getAllFavouritesByUser(request):
    favourite_list=services_nasa_image_gallery.getAllFavouritesByUser(request)
    return render(request, 'favourites.html', {'favourite_list': favourite_list})
```

Función que renderiza el template favourites.html llamando a la función `getAllFavouritesByUser` desde `services_nasa_image_gallery`. Mostrando lista de favoritos.

saveFavourite(request):

```
@login_required
def saveFavourite(request):
    services_nasa_image_gallery.saveFavourite(request)
    return redirect('favoritos') #redirecciona a favoritos para verificar que se haya agregado correctamente
```

Trae la función `savefavourite` desde `services_nasa_image_gallery` y le otorga el argumento `request`

deleteFavourite(request):

```
@login_required
def deleteFavourite(request):
    services_nasa_image_gallery.deleteFavourite(request)
    return getAllFavouritesByUser(request)
```

Trae la función `deleteFavourite` desde `services_nasa_image_gallery` y le otorga el argumento `request`.

Modificaciones en TEMPLATES:

- a) Footer: Se modificó el *class* para mejorar la visual y se agregó los nombres de los participantes.

```
<footer class="footer-custom text-center text-lg-start fixed-bottom">
  <div class="footer-content text-center p-3" style="display: flex; justify-content: space-between;">
    <strong>Introducción a la Programación | UNGS: Angel Letti, Juan Scopa, Colman Victor</strong>
    <strong>{{VERSION}}</strong>
  </div>
</footer>
</body>
</html>
```

Introducción a la Programación | UNGS: Angel Letti, Juan Scopa, Colman Victor

Trabajo práctico - 1er cuatrimestre del 2024.

- b) Header: Se agregaron plantillas para trabajar en el formato de la aplicación y el tamaño de las cards. También modificamos cada class para que tome el formato de la plantilla.

Formato de la card:

```
}
.small-card {
  max-width: 300px;
  margin: 10px auto;
}
.small-card img {
  height: auto;
  max-width: 100%;
}
```

Al mismo tiempo, se agregaron las urls necesarias para que funcione el login y logout

```
{% if request.user.is_authenticated %}
  <a class="nav-link" href="{% url 'logout' %}">Salir</a>
{% else %}
  <a class="nav-link" href="{% url 'login' %}">Iniciar sesión</a>
{% endif %}
```

- c) Index: Agregamos una etiqueta "style" para mejorar el formato y también un formato de "nasa-logo" para introducir la imagen de la nasa para decorar el inicio.

```
.nasa-logo {
  display: block;
  margin: 0 auto; /* Centrar horizontalmente */
  max-width: 20%; /* Ajustar al ancho máximo disponible */
  height: auto; /* Mantener la proporción de la imagen */
  margin-bottom: 20px; /* Espacio inferior */
}
</style>

<img src="https://upload.wikimedia.org/wikipedia/commons/thumb/e/e5/NASA_Logo.svg/1200px-NASA_Logo.svg.png"
```

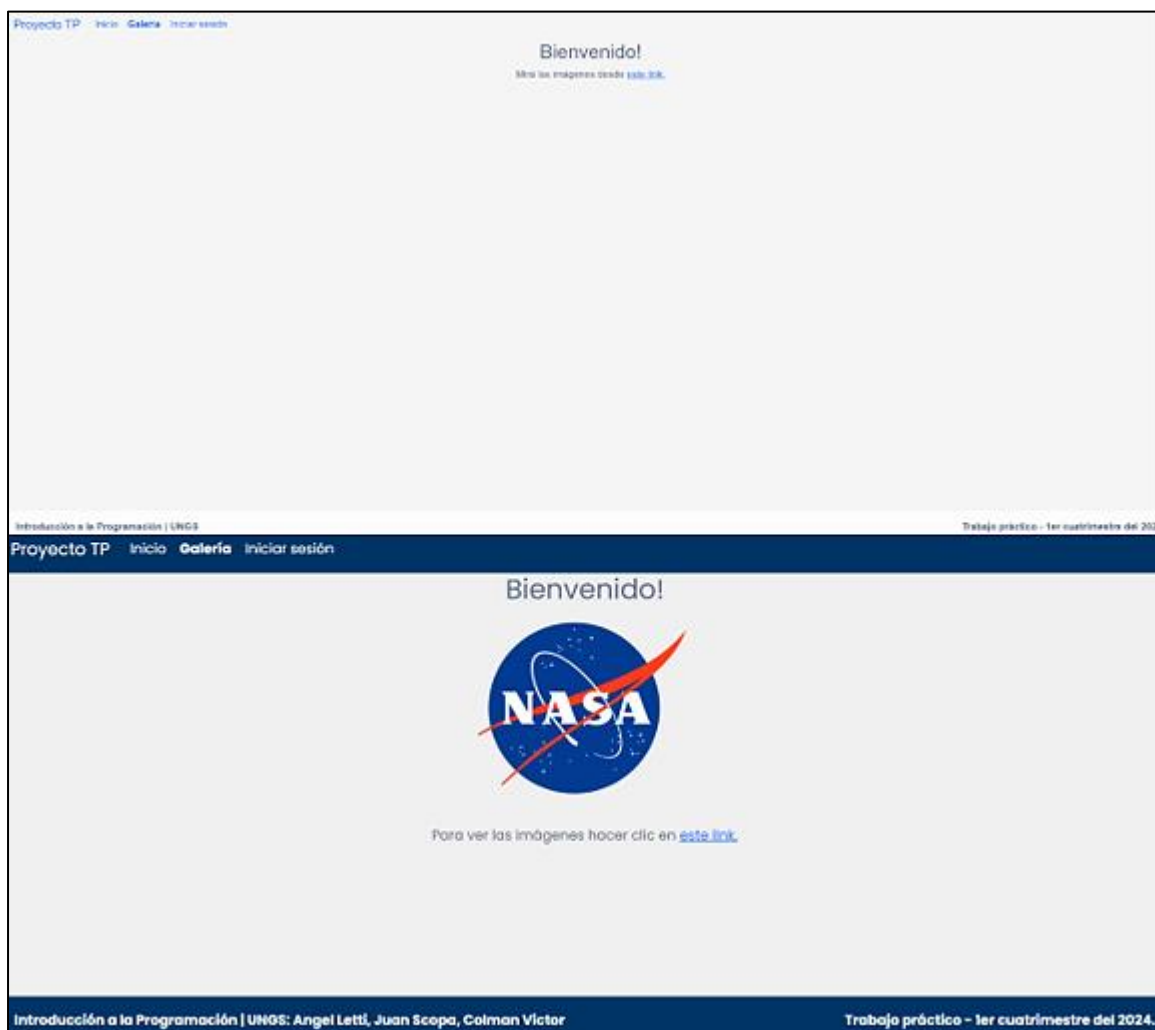
- d) Home: Se modificaron las clases en las etiquetas para asociarlas a las plantillas agregadas, también se agregaron las etiquetas necesarias para visualizar el paginador correctamente con los bucles y condicionantes necesarios.

4. Conclusiones:

Con respecto al código podemos comentar que ha sido un desafío para el equipo poder hacer funcionar el “home”, debido a la falta de conocimientos específicos sobre cómo se programa en django y html. Luego de varias jornadas de trabajo pudimos concretar y gracias a ese progreso, se nos hizo más sencillo avanzar con los adicionales, con excepción del control de páginas, el cual presento un desafío importante, ya que debíamos agregar varias funciones en el código del template, del cual no teníamos experiencia alguna.

Comparativa de visuales:

Home



Galería

[Proyecto TP](#) [Inicio](#) [Galería](#) [Iniciar sesión](#)

Galería de Imágenes de la NASA

EXPLORE MOON & MARS



Space Exploration Video
Space Exploration Video



Space 24/7/365: Space Station Benefits
As a home to humans and a lab for scientific research for more than 20 years, the International Space Station is a one-of-a-kind platform for advancing technologies such as robots, computers, health monitors, life support systems, and more for both space and ground applications. The station also has made it possible for numerous experiments to delve into how space travel affects people physically and psychologically – knowledge that often improves the quality of life for people on Earth, too. Learn more: <https://go.nasa.gov/3ATQOig> Explore other station benefits: <https://www.nasa.gov/stationbenefits> Image Credit: Canadian Space Agency © 17



Space-to-Ground_171_170407
NASA's Space to Ground is your weekly update on what's happening aboard the International Space Station. Got a question or comment? Use #spacetoground to talk to us

FOLLOW THE SPACE STATION! Twitter: https://twitter.com/Space_Station Facebook: <https://www.facebook.com/ISS> Instagram: <https://instagram.com/iss/>

[Proyecto TP](#) [Inicio](#) [Galería](#) [Iniciar sesión](#)

Galería de Imágenes de la NASA

Imágenes por página:

EXPLORE MOON & MARS



Space Exploration Video
Space Exploration Video



Space 24/7/365: Space Station Benefits
As a home to humans and a lab for scientific research for more than 20 years, the International Space Station is a one-of-a-kind platform

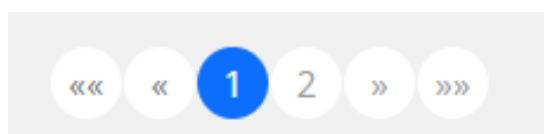


Space-to-Ground_171_170407
NASA's Space to Ground is your weekly update on what's happening aboard the International Space Station. Got a

Introducción a la Programación | UNGS: Angel Letti, Juan Scopa, Colman Victor

Trabajo práctico - 1er cuatrimestre del 2024.


Paginador:



Lista de favoritos

Proyecto TP Inicio **Galería** Favoritos Salir

Listado de FAVORITOS

#	Imagen	Título	Descripción	Fecha	Acciones
-		Dusty Space Cloud	This image shows the Large Magellanic Cloud galaxy in infrared light as seen by ESA Herschel Space Observatory and NASA Spitzer Space Telescope. The brightest center-left region is called 30 Doradus, or the Tarantula Nebula.	Jan. 10, 2012	