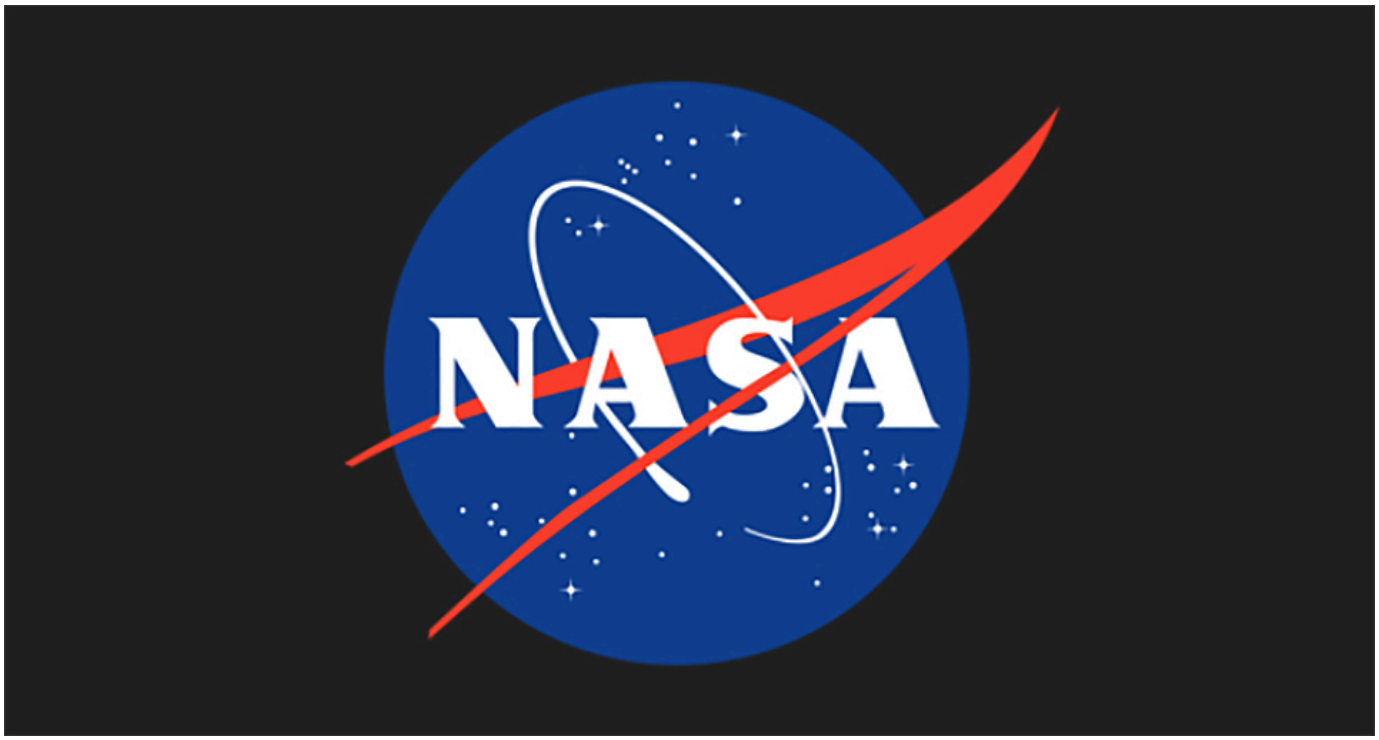
Trabajo API Nasa Introducción A La Programación

Introducción

El trabajo consiste en utilizar la API de la Nasa para visualizar una galería de imágenes paginadas cada cinco, de lo que busque el usuario. El mismo puede añadir las fotos que quiera a “favoritos” con un comentario.



Lauzan, Mateo
Daniluk, Ignacio

Funciones

Carpeta auth

Archivo auth -> views.py

```python
def login_user(request):
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('home')
        messages.error(request, "Credenciales inválidas")
        return redirect('login')
```

Se recibe el nombre de usuario y la contraseña desde el formulario, se intenta autenticar y, en caso, de que las credenciales sean válidas, se autentica al usuario. Si estas son incorrectas, se devuelve un error en la vista.

```python
def create_user(request):
        username = request.POST.get('username')
        password = request.POST.get('password')
        email = request.POST.get('mail')
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')

        if not (username and password and email):
            messages.error(request, "Los campos username, password y email son obligatorios")
            return redirect("create_page")

        if User.objects.filter(username=username).exists():
            messages.error(request, "El usuario ya se encuentra registrado")
            return redirect("create_page")

        try:
            User.objects.create_user(
                username=username,
                password=password,
                email=email,
                first_name=first_name,
                last_name=last_name
            )
            subject = 'Credenciales de acceso'
            message= f'Usuario: {username}, contraseña: {password}'
            recipient = email.strip()

            send_mail(subject, message, EMAIL_HOST_USER, [recipient], fail_silently=False)

            messages.success(request, f"Usuario creado con éxito, enviamos tus credenciales a {email}")
            return redirect("create_page")
        except Exception as e:
            messages.error(request, f"No se pudo crear el usuario: {str(e)}")
            return redirect("create_page")
```

Se reciben los datos del usuario y si, el nombre de usuario, la contraseña o el email no se envían, se devuelve un mensaje de error. Luego se chequea si el nombre de usuario ya se encuentra registrado en la base de datos. Si eso es correcto, se devuelve un error. Por último, se crea el usuario en la base de datos y se envía un correo al email especificado con sus credenciales de acceso.

Carpeta nasa_image_gallery

Archivo nasa_image_gallery -> layers -> dao -> repositories.py

```python
def saveFavourite(image):
    try:
        fav = Favourite.objects.create(title=image.title, description=image.description, image_url=image.image_url, date=imag
        date, user=image.user, comment=image.comment)
        return fav
    except Exception as e:
        print(f"Error al guardar el favorito: {e}")
        return None

def getAllFavouritesByUser(user):
    favouriteList = Favourite.objects.filter(user=user).values('id', 'title', 'description', 'image_url', 'date', 'comment')
    return list(favouriteList)

def deleteFavourite(id):
    try:
        favourite = Favourite.objects.get(id=id)
        favourite.delete()
        return True
    except Favourite.DoesNotExist:
        print(f"El favorito con ID {id} no existe.")
        return False
    except Exception as e:
        print(f"Error al eliminar el favorito: {e}")
        return False
```

La primera función recibe como argumentos los datos que le llegan a la función, crea el objeto en la base de datos y los devuelve.

La segunda recibe el usuario y devuelve una lista con las imágenes que este tiene agregadas como favoritos

La tercera recibe el id del registro y, si existe en la base de datos lo borra, sino devuelve un error

Archivo nasa_image_gallery -> layers -> generic -> mapper.py

```
def fromRequestIntoNASACard(object):
    nasa_card = NASACard(
                        title=object['data'][0]['title'],
                        description=object['data'][0]['description'],
                        image_url=object['links'][0]['href'],
                        date=object['data'][0]['date_created'][:10]
                )

    return nasa_card


def fromTemplateIntoNASACard(templ):
    nasa_card = NASACard(
                        title=templ.POST.get("title"),
                        description=templ.POST.get("description"),
                        image_url=templ.POST.get("image_url"),
                        date=templ.POST.get("date"),
                        comment=templ.POST.get("comment")
                )
    return nasa_card


def fromRepositoryIntoNASACard(repo_dict):
    nasa_card = NASACard(
                        id=repo_dict['id'],
                        title=repo_dict['title'],
                        description=repo_dict['description'],
                        image_url=repo_dict['image_url'],
                        date=repo_dict['date'],
                        comment=repo_dict['comment']
                )
    return nasa_card
```

La primera función convierte la info de la API de la nasa en una NASACard

La segunda, la convierte desde la solicitud para guardarlo en la base de datos

La tercera la transforma desde la base de datos y lo convierte en una NASACard antes de devolvérsela al usuario

Archivo nasa_image_gallery -> layers -> generic -> nasa_card.py

```python
class NASACard:
    def __init__(self, title, description, image_url, date, user=None, id=None, comment=None):
        self.title = title
        self.description = description
        self.image_url = image_url
        self.comment = comment
        self.date = date
        self.user = user
        self.id = id

    def __str__(self):
        return f'Título: {self.title}, Descripción: {self.description}, URL de la imagen: {self.image_url}, Fecha: {self.date},
        Usuario: {self.user}, Id: {self.id}, Comentario: {self.comment}'

    # 2 NASACards son iguales si comparten el mismo title, description e image_url.
    # método equals.
    def __eq__(self, other):
        if not isinstance(other, NASACard):
            return False
        return (self.title, self.description, self.image_url) == \
               (other.title, other.description, other.image_url)

    # método hashCode.
    def __hash__(self):
        return hash((self.title, self.description, self.image_url))
```

Crea la clase de NASACard con sus propios métodos

Archivo nasa_image_gallery -> layers -> generic -> utils.py

```python
def pagesQuantity(imagesQuantity):
    pages = imagesQuantity/ 5
    if pages > 2000:
        pages = 2000

    return math.ceil(pages)

def getPreviousPages(number):
    previousNumbers = []
    for i in range(number - 1, 1, -1):
        if i > 1 and len(previousNumbers) < 5:
            previousNumbers.append(i)
    if not previousNumbers:
        return None
    return previousNumbers[::-1]

def getNextPages(number, limit):
    nextNumbers = []
    for i in range(number + 1, limit):
        nextNumbers.append(i)
        if len(nextNumbers) >= 5:
            break

    return nextNumbers
```

La primera función recibe la cantidad de imágenes que devuelve la API y divide a ese número por 5 ya que es la cantidad que se van a mostrar por página. En caso de que el número de páginas de mayor a 2000, se le asigna ese valor ya que la API no acepta un valor de página mayor. Por último, se retorna el número redondeado para arriba.

La segunda, recorre desde un número **(la página actual)** menor al indicado por parámetro hasta el 1 para obtener los anteriores. La función se corta si ya se llegó al 1 o si ya se obtuvieron los 5 números menores.

La última función del archivo, recibe el número de la página actual y el límite que es la última página de las imágenes. Si ya hay números en la lista, corta el ciclo.

Archivo nasa_image_gallery -> layers -> services -> services_nasa_image_gallery.py

```python
def getAllImages(input=None, page='1'):
    data = transport.getAllImages(input, page)
    images = []
    for obj in data[0]:
        image = mapper.fromRequestIntoNASACard(obj)
        images.append(image)

    return images, data[1]


def getImagesBySearchInputLike(input):
    return getAllImages(input)

def saveFavourite(request):
    fav = mapper.fromTemplateIntoNASACard(request)
    fav.user = get_user(request)

    return repositories.saveFavourite(fav)


def getAllFavouritesByUser(request):
    if not request.user.is_authenticated:
        return []
    else:
        user = get_user(request)

        favourite_list = repositories.getAllFavouritesByUser(user)
        mapped_favourites = []

        for favourite in favourite_list:
            nasa_card = mapper.fromRepositoryIntoNASACard(favourite)
            mapped_favourites.append(nasa_card)

        return mapped_favourites


def deleteFavourite(request):
    favId = request.POST.get('id')
    return repositories.deleteFavourite(favId)
```

La primera función recibe las imágenes desde la API, las recorre para convertirlas en una NASACard y las devuelve con la cantidad de páginas.

La segunda función recibe el valor del término de búsqueda que ingresó el usuario y llama a la función previamente mencionada **getAllImages** para retornar las imágenes

La tercera recibe la imágen desde la solicitud, la convierte en una NASACard, obtiene al usuario de la solicitud y llama a la función previamente mencionada **saveFavourite** para guardar el registro en la base de datos.

La cuarta función llama a la función previamente mencionada **getAllFavouritesByUser** y obtiene las imágenes favoritas del usuario autenticado, si el usuario no está autenticado devuelve una lista vacía, luego convierte estas imágenes en una NASACard y las retorna.

La última obtiene el id del registro enviado a través de la solicitud y llama a la función previamente mencionada **deleteFavourite**

Archivo nasa_image_gallery -> layers -> transport -> transport.py

```
def getAllImages(input=None, page='1'):
    if input is None:
        json_response = requests.get(config.NASA_REST_API_DEFAULT_SEARCH + '&page=' + page +'&page_size=5').json()
    else:
        json_response = requests.get(config.NASA_REST_API + input + '&page=' + page +'&page_size=5').json()

    json_collection = []

    pages = pagesQuantity(json_response['collection']['metadata']['total_hits'])

    for object in json_response['collection']['items']:
        try:
            if 'links' not in object:
                object['links'] = [{'href': '/static/images/Tierra.png'}]

            json_collection.append(object)

        except KeyError:
            pass

    return json_collection, pages
```

Esta función llama a la API de la Nasa, obtiene las imágenes, cuenta la cantidad de páginas con la función previamente mencionada **pagesQuantity** y recorre cada una. En caso de que no tenga imagen se le asigna una por defecto.

Archivo nasa_image_gallery -> models.py

```
class Favourite(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    image_url = models.TextField()
    date = models.DateField()
    comment = models.TextField(default='')

    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)

    class Meta:
        unique_together = ('user', 'title', 'description', 'image_url', 'date', 'comment')
```

Se define la clase del modelo de **Favourite** para utilizarla en distintas funciones.

Archivo nasa_image_gallery -> views.py

```
def index_page(request):
    return render(request, 'index.html')

def login_page(request):
        return render(request, 'registration/login.html')

def getAllImagesAndFavouriteList(request, input, page):
    fetchImages = services_nasa_image_gallery.getAllImages(input, page)
    images = fetchImages[0]
    pages  = fetchImages[1]
    favourites = services_nasa_image_gallery.getAllFavouritesByUser(request)
    favourite_list = favourites

    return images, favourite_list, pages

def home(request):
    page = request.GET.get('page') or '1'
    fetchImagesAndFavourites = getAllImagesAndFavouriteList(request, input=None, page=page)

    images = fetchImagesAndFavourites[0]
    favourite_list = fetchImagesAndFavourites[1]

    pages = int(fetchImagesAndFavourites[2])
    previous_pages= getPreviousPages(int(page))
    next_pages = getNextPages(int(page), pages)

    return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list, 'query': 'space', 'pages': pages,
    'previous_pages' : previous_pages, 'next_pages' : next_pages} )
```

La primeras dos funciones del archivo devuelve los archivos html.

La tercera llama a las funciones previamente mencionadas **getAllImages y getAllFavouritesByUser** y devuelve la lista de imágenes, de favoritos del usuario y las páginas que van.

La cuarta función carga el html de la página principal pero antes obtiene la página por parámetro de la url y luego llama a las funciones previamente mencionadas **getAllImagesAndFavouriteList**, **getPreviousPages** y **getNextPages**.

```
def search(request):
    search_msg = request.GET.get('query') or 'space'
    page = request.GET.get('page') or '1'

    images, favourite_list, pages = getAllImagesAndFavouriteList(request, search_msg, page)

    if not images:
        messages.error(request, "No se encontraron imágenes para: " + search_msg)
        return redirect('home')

    previous_pages= getPreviousPages(int(page))
    next_pages = getNextPages(int(page), pages)

    return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list, 'query': search_msg, 'pages': str
    (pages), 'previous_pages' : previous_pages, 'next_pages' : next_pages} )


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


@login_required
def saveFavourite(request):
    services_nasa_image_gallery.saveFavourite(request)
    return redirect('favoritos')
```

La función search obtiene por parámetro el término que ingresó el usuario, sino se le asigna el valor de “space”, y el número de página que si es nulo se le asigna el 1. Luego obtiene las imágenes, la lista de favoritos y las páginas con la función previamente mencionada **getAllImagesAndFavouriteList**. Si no hay imágenes para este término de búsqueda se retorna un mensaje de error. Después se obtienen las anteriores y siguientes páginas con las funciones previamente mencionadas **getPreviousPages y getNextPages** y se retorna esa info con el html.

La función getAllFavouritesByUser llama a la función previamente mencionada **getAllFavouritesByUser** y devuelve el html (sólo se puede acceder a esta página estando autenticado).

La función saveFavourite llama a la función previamente mencionada **saveFavourite** y te redirige a la página dónde están las imágenes marcadas como favoritas (sólo se puede llamar a la función estando autenticado).

```
@login_required
def deleteFavourite(request):
    services_nasa_image_gallery.deleteFavourite(request)
    return redirect('favoritos')

@login_required
def exit(request):
    logout(request)
    return redirect('home')

def create_page(request):
    return render(request,"registration/register.html")
```

La primera función de la foto llama a la función previamente mencionada **deleteFavourite** para borrar el registro de la imagen guardada como favorita y te recarga la página (sólo se puede llamar a la función estando autenticado).

La función de exit desautentica al usuario y lo redirige a la página principal.

La última función de la foto carga el html de la página de registro.

```python
urlpatterns = [
    path('', views.index_page, name='index-page'),
    path('login/', views.login_page, name='login'),
    path('home/', views.home, name='home'),
    path('buscar/', views.search, name='buscar'),
    path('register/', views.create_page, name='create_page'),
    path('favourites/', views.getAllFavouritesByUser, name='favoritos'),
    path('favourites/add/', views.saveFavourite, name='agregar-favorito'),
    path('favourites/delete/', views.deleteFavourite, name='borrar-favorito'),

    path('exit/', views.exit, name='exit'),
]
```

```python
urlpatterns = [
    path('login_user/', views.login_user, name='login_user'),
    path('create_user/', views.create_user, name='create_user'),

]
```

En ambos archivos (**urls.py**) se definen las urls. En uno las de la carpeta **auth** y en el otro las de la carpeta **nasa_image_gallery**.