Universidad Nacional de General Sarmiento

Trabajo práctico:

Introducción a la programación

COMISIÓN: 10

DOCENTES: Jose Luis Mieres, Sergio Santa Cruz, Miguel Rodriguez

TURNO: Miércoles: 18:00 - 22:00

ALUMNOS: Alejandro Karamanin, Juan Rodriguez

Introducción:

En este informe nos explayaremos sobre cómo hemos realizado el trabajo práctico para la materia de **introducción a la programación.** En primer lugar corresponde aclarar el objetivo del trabajo práctico: El mismo consiste en implementar ciertas funciones a una aplicación web fullstack. *(vale aclarar que el grueso de la aplicación ya nos había sido entregado)*

La función principal a implementar es que se pueda acceder desde la página a una galería con las imágenes de una API pública de la NASA. Además de dicha función, se plantearon objetivos adicionales como: hacer funcionar el buscador central, implementar un módulo de autenticación básica (usuario/contraseña), lograr que el usuario pueda guardar resultados como favoritos y los mismos se puedan visualizar y borrar del listado.

Funciones implementadas y su dificultad en el desarrollo:

a) Galería principal

Para lograr que la página muestre modificamos tres de las funciones que ya estaban incluidas en el trabajo, en primer lugar modificamos la función “getAllImages” de “services_nasa_gallery” siguiendo las anotaciones que había incluido el trabajo:

```python
def getAllImages(input=None):
    # obtiene un listado de imágenes desde transport.py y lo guarda en json_collection.
    # ¡OJO! el parámetro 'input' indica si se debe buscar por un valor introducido en el buscador.
    json_collection = []

    images = []

    # recorre el listado de objetos JSON, lo transforma en una NASACard y lo agrega en el listado de images. Ayuda: ver mapper.py.

    return images
```

Para obtener los Json de todas la imágenes de la API utilizamos la función “getAllImages” de “transport” que conecta directamente con la API, posteriormente los Json son convertidos en “NasaCard” utilizando la función “fromRequestIntoNASACard” de “mapper”. Estas “NasaCard” son guardadas en un listado llamado “images” que posteriormente es llamado por la capa “view.py” para ser utilizadas.

(Función modificada)

```
def getAllImages(input=None):
    images = []
    json_collection = transport.getAllImages(input)
        for object in json_collection:
        imageToConvertIntoNasaCard =
mapper.fromRequestIntoNASACard(object)
        images.append(imageToConvertIntoNasaCard)
    return images
```

En segundo lugar modificamos la función “getAllImagesAndFavouriteList” la misma invoca la funcion “getAllImages” de “services_nasa_gallery” para obtener 2 listados, uno de las imágenes de la API y otro -si corresponde- de los favoritos del usuario.

(función modificada)

```
def getAllImagesAndFavouriteList(request):
    images = services_nasa_image_gallery.getAllImages()
    favourite_list = []

    return images, favourite_list
```

La última modificación fue a la función “home”, la misma invoca a “getAllImagesAndFavouriteList” para obtener 2 listados que utiliza para renderizar el template.

(función modificada)

```
def home(request):

    images = []

    favourite_list = []

    images, favourite_list = getAllImagesAndFavouriteList(request)

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

El principal desafío fue entender qué era exactamente lo que debía devolver la función de “services_nasa_gallery” a “view.py”. Una vez entendimos que debía llamar las funciones de “transport” para conectarse con la API y posteriormente transformar lo devuelto en “NasaCards” pudimos avanzar con la función.

Otra parte que nos retrasó bastante es entender cómo importar las funciones de una capa a la otra, pero tras algunos tutoriales pudimos solucionarlo.

b) Buscador

Para implementar la función del buscador desarrollamos tres funciones, dos en la capa “view.py” y una en “services_nasa_gallery”. Originalmente lo habiamos hecho en una sola función en “view.py” pero siendo que la parte de la logica deberia estar en “services_nasa_gallery” optamos por dividir dicha funcion

(esta fue la función original que se termino descartando por las 2 funciones que explicaremos a continuación)

```
def search(request):

    selectedImages = []

    images, favourite_list = getAllImagesAndFavouriteList(request)

    search_msg = request.POST.get('query', '')

    if search_msg == "":

        return refresh(request,images,favourite_list)

    for NasaCard in images:

        if search_msg.lower() in NasaCard.title.lower() or
search_msg.lower() in NasaCard.description.lower():

            selectedImages.append(NasaCard)

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

La funcion “getImagesBySearchInputLike” de “services_nasa_image_gallery.py” toma un listado de “NasaCards” revisa si hay coincidencias en la misma con la palabra introducida en el buscador (search_msg), si es así lo agrega a un listado

llamado "selectedImages" que posteriormente se devuelve. Además se pone la palabra del buscador como la del título y la de la descripción de la “NasaCard” en minúscula para evitar que alguna imagen no entre en el listado por tener diferencias entre mayúsculas y minúsculas.

```
def getImagesBySearchInputLike(request, search_msg, images):

    selectedImages = []

    for NasaCard in images:

                if   search_msg.lower()   in   NasaCard.title.lower()   or
search_msg.lower() in NasaCard.description.lower():

            selectedImages.append(NasaCard)

    return selectedImages
```

La función “search” es parecida a la función “home” previamente descripta. Toma un listado de imágenes, luego las filtra utilizando la función “getImagesBySearchInputLike” y finalmente la renderiza.

```
def search(request):

    images, favourite_list = getAllImagesAndFavouriteList(request)

    search_msg = request.POST.get('query', '')

    if search_msg != "":

      selectedImages =
services_nasa_image_gallery.getImagesBySearchInputLike(request,
search_msg, images)

       return  render(request,  'home.html',  {'images':  selectedImages,
'favourite_list': favourite_list} )

     else:

        return redirect('home')
```

c) Login

Si bien tuvimos que dedicar bastante tiempo para entender el funcionamiento del framework, la modificación para hacerlo funcionar en si fue corta. En el archivo de la “Urls.py” hay una llamada al framework de django que redirige directamente al repositorio donde se encuentra el http del “login” lo que hicimos fue “arreglar” los botones el header para que llame a los Url. Además se modificó el header para que muestre el “éxit” y el “login.”

d) favoritos

Para que la página pudiera mostrar favoritos se necesitaron varias funciones en la capa de “services_nasa_image_gallery.py” y de “view.py” , en primer lugar completamos las funciones “saveFavourite”, “getAllFavouritesByUser” y “deleteFavourite” que ya nos habían sido entregadas con el trabajo en “services_nasa_image_gallery.py”, ademas de utilizar las funciones de “repositories.py” que ya estaban terminadas

En la función “saveFavourite” agregamos la función “fromTemplateIntoNASACard” desde mapper, para transformar el template en una NasaCard y posteriormente se invoca la función “saveFavourite” desde “repository” para guardarlo en la base de datos

```
def saveFavourite(request):
    if request.method == 'POST':
            fav = mapper.fromTemplateIntoNASACard(request)
            if fav:
                fav.user = request.user
                return repositories.saveFavourite(fav)
    return None
```

Por otro lado en la función “getAllFavouritesByUser” en primer lugar verifica si el usuario está ingresado, de ser así busca los favoritos del usuario almacenados usando la función “getAllFavouritesByUser” que ya venía en “repositories”. Luego

transforma el listado de favoritos en “NasaCard” invocando la función desde mapper y devuelve el listado. (posteriormente es llamada por “getAllFavouritesByUser” desde view)

```
def getAllFavouritesByUser(request):

    if request.user.is_authenticated:

        user = get_user(request)

           favourite_list = repositories.getAllFavouritesByUser(user) # buscamos desde el repositorio TODOS los favoritos del usuario (variable 'user').

        mapped_favourites = []



        for favourite in favourite_list:

             nasa_card = mapper.fromRepositoryIntoNASACard(favourite) # transformamos cada favorito en una NASACard, y lo almacenamos en nasa_card.

            mapped_favourites.append(nasa_card)



        return mapped_favourites

    else:

        return None
```

Por último la función “deleteFavourite” toma el pedido del usuario y usando la función “deleteFavourite” borra el favorito de la base de datos

```
def deleteFavourite(request):

    if request.method == 'POST':

        favId = request.POST.get('id')

        if favId:

               return repositories.deleteFavourite(favId) # borramos un favorito por su ID.)

        return None
```

Desde “view.py” las funciones son las siguientes **“getAllFavouritesByUser”, “saveFavourite” y “deleteFavourite”**

En primer lugar **“getAllFavouritesByUser”** crea el listado usando la función del mismo nombre desde “services_nasa_image_gallery” y renderiza dicho listado.

```
def getAllFavouritesByUser(request):

   favourite_list =
services_nasa_image_gallery.getAllFavouritesByUser(request)

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

La función **“saveFavourite”** aplica la función del mismo nombre y carga la página

```
def saveFavourite(request):

    success = services_nasa_image_gallery.saveFavourite(request)

    if success:

        return redirect('home')

    else:

            return render(request, 'home.html', {'error': 'No se pudo
guardar el favorito.'})
```

La función **“deleteFavourite”** llama a la función del mismo nombre para borrar al favorito del listado y recarga la página que muestra los favoritos. En el caso de que no pueda, muestra el mensaje de error

```
def deleteFavourite(request):

    success = services_nasa_image_gallery.deleteFavourite(request)

    if success:

        return getAllFavouritesByUser(request)

    else:

            return render(request, 'home.html', {'error': 'Error al
eliminar el favorito.'})
```

3) Conclusión

Consideramos que el trabajo nos sirvió para aprender a trabajar en funciones específicas de un entorno ya hecho y sobre todo a investigar, ya que el funcionamiento de Django y los repositorios nos era casi totalmente ajeno durante la cursada