Universidad Nacional de General Sarmiento

Trabajo grupal de

Introducción a la programación

Trabajo práctico: galería de imágenes de la NASA

Profesoras:

Bottino Flavia

Winograd Natalia

Alumnos/as:

Fester Hein Valentina Ivone

Figueroa Lucas Gabriel

Roldán Marianela Sabrina

Ruiz Gonzalo Nicolas

Comisión:

COM-02

Fecha de entrega:

Jueves 27 de junio de 2024

Trabajo práctico: galería de imágenes de la NASA

introducción

El desarrollo web ha evolucionado significativamente en las últimas décadas, transformándose en un elemento clave de la estructura tecnológica moderna. Desde la creación de las primeras páginas estáticas hasta el despliegue de aplicaciones online, la programación ha permitido a las empresas y desarrolladores crear experiencias digitales sofisticadas y accesibles. Este informe tiene como objetivo explorar los componentes fundamentales del desarrollo web, examinar las tecnologías y lenguajes de programación utilizados en la materia. El informe proporcionará una visión integral del proceso de programación, desde la concepción de una idea hasta la implementación de aplicaciones en línea, para lograr nuestro objetivo abordaremos diversos aspectos como, en qué consistió el trabajo, las funciones implementadas en él, las dificultades de implementación y las decisiones tomadas, para poder llegar a la conclusión final. Por último, se espera que quien está empezando a programar tenga una comprensión clara de cómo se construyen este tipo sitios web.

¿En qué consistió el trabajo?

El trabajo consistió en desarrollar tres funciones, ya creadas pero incompletas, que permitirían visualizar la galería de imágenes de una página en línea dedicada a imágenes de la NASA, galería conformada por objetos llamados NASAcard, los cuales estaban compuestos por una imagen, un título y una descripción.
Para esto, utilizamos distintas herramientas como: Django Framework que permite consultar las imágenes de la API pública que proporciona la NASA; Visual Studio Code y Python, para desarrollar las funciones; y GitHub, para poder compartir el repositorio donde trabajamos.
Para ello debimos aplicar todo lo aprendido durante la cursada de la materia, haciendo mayor hincapié en el tema de listas y ciclos.

Funciones implementadas

services_nasa_image_gallery.py:

- *getAllImages(input=None):*

```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
```

Esta función invoca al servicio correspondiente que traía la API, por ello nos dimos cuenta que era la primera función que había que realizar para después mandarla al home(request). Para hacer que funcione se transportaron las imágenes, se recorrieron y se utilizó la aplicación del mapper. Este último, se refiere a un componente o conjunto de funciones que se utiliza para convertir o "mapear" datos de un formato o estructura a otro.

Código modificado:

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

    for json in json_collection:
     images.append(mapper.fromRequestIntoNASACard(json))

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

    return images
```

views.py:

- *getAllImagesAndFavouriteList(request):*

```python
# auxiliar: retorna 2 listados -> uno de las imágenes de la API y otro de los favoritos del usuario.
def getAllImagesAndFavouriteList(request):
    images = []
    favourite_list = []

    return images, favourite_list
```

La segunda función utilizada fue getAllImagesAndFavouriteList. Como obtenía dos listados, decidimos que, en la lista vacía de imágenes, se juntaran las imágenes de services_nasa_image_gallery con la función de getAllImages. Luego, para la lista vacía de favourite_list, pasaba algo similar, juntamos las mismas imágenes de services_nasa_image_gallery, pero con getAllFavouritesByUser(request), para así para implementar la sección de favoritos: traer los favoritos de un usuario, guardarlos, eliminarlos.

Código modificado:

```python
# auxiliar: retorna 2 listados -> uno de las imágenes de la API y otro de los favoritos del usuario.
def getAllImagesAndFavouriteList(request):
    images = services_nasa_image_gallery.getAllImages()
    favourite_list =services_nasa_image_gallery.getAllFavouritesByUser(request)
```

- *home(request):*

```python
def home(request):
    # llama a la función auxiliar getAllImagesAndFavouriteList() y obtiene 2 listados: uno de imágenes de la API y otro de favoritos por usuario*
    # (*) este último, solo si se desarrolló el opcional de favoritos; caso contrario, será un listado vacío [].
    images, favourite_list = []
    return render(request, 'home.html', {'images': images, 'favourite_list': favourite_list} )
```

Finalmente, notamos que la función home(request), llamaba a la función hecha anteriormente(getAllImagesAndFavouriteList(request). Por ello, completamos la lista vacía de images, favourite_list con getAllImagesAndFavouriteList(request).

Código modificado:

```python
# función principal de la galería.
def home(request):
    # llama a la función auxiliar getAllImagesAndFavouriteList() y obtiene 2 listados: uno de las imágenes de la API y otro de favoritos
    # (*) este último, solo si se desarrolló el opcional de favoritos; caso contrario, será un listado vacío [].
    images,favourite_list =getAllImagesAndFavouriteList(request)

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

Extras:

views.py

- *search(request):*

```
# función utilizada en el buscador.
def search(request):
    images, favourite_list = getAllImagesAndFavouriteList(request)
    search_msg = request.POST.get('query', '')

    # si el usuario no ingresó texto alguno, debe refrescar la página; caso contrario, debe filtrar aquellas imágenes que posean el texto
    pass
```

Como extra, elegimos completar la función seach(request), para que el buscador filtre adecuadamente las imágenes por el dato ingresado por el usuario o que muestre las imágenes del valor predeterminado si el usuario no ingresa nada. Por si sola no hacía nada; notamos que request.POST.get('query', '') obtenía el dato ingresado por el usuario, por lo que se aplicó un condicional que, si el valor obtenido era distinto de '', una nueva variable llamada images_filtered, que contenía la función getImagesBySearchInputLike(search_msg) de la capeta services , devolvería lo pedido por el usuario y la lista de favoritos; en caso contrario, redireccionaría al "home" y mostraría las imágenes por defecto.

Código modificado:

```
# función utilizada en el buscador
def search(request):
    images, favourite_list = getAllImagesAndFavouriteList(request)

    search_msg = request.POST.get('query', '')
    if (search_msg !=''):
        images_filtered = services_nasa_image_gallery.getImagesBySearchInputLike(search_msg)
        return render(request, 'home.html', {'images': images_filtered, 'favourite_list': favourite_list} )
    else:
        return redirect ('home')
```

Dificultades de implementación y decisiones tomadas

Los obstáculos se presentaron desde el inicio del trabajo, fue un poco complejo el momento de la instalación de las aplicaciones, como aún no teníamos mucho conocimiento de GitHub o sobre los repositorios, costó un poco más. Además, se nos presentó la dificultad de ver el Pyscripter que ya lo teníamos familiarizado al ver el VSCode con toda la página que tenía miles de términos que no se entendían.
En cuanto a las funciones, con leer lo que te proporcionaban como consejos no bastaba, por lo que decidimos investigar por cuenta propia las otras funciones que se tenían que llamar como el mapper o el transport o las funciones NASAcard, que resultaron muy importantes, ya que sin ellas no se podía llevar a cabo la finalidad del trabajo práctico.
En el caso del buscador, se nos complicó entender que traía la variable search_msg, pero finalmente nos dimos cuenta de que traía lo ingresado por el usuario, así que pudimos completar la función.

Conclusión

Si bien la actividad se nos dificultó al principio, pudimos resolver con éxito los requerimientos mínimos para la entrega del trabajo. El desarrollo de las funciones para la galería de imágenes de la NASA nos permitió aplicar los conocimientos adquiridos durante la cursada, además obtuvimos un mejor entendimiento de herramientas clave como Django Framework, Visual Studio Code, Python y GitHub.
Con respecto a las dificultades encontradas, como la instalación de aplicaciones y el desarrollo de las funciones, nos vimos obliga dos a investigar y aprender de manera autónoma, lo que resultó en un crecimiento significativo en nuestras capacidades.
En conclusión, este trabajo no solo cumplió con los objetivos mínimos propuestos, sino que también proporcionó una valiosa experiencia práctica en el desarrollo de aplicaciones web complejas. Los conocimientos y habilidades adquiridos durante este proyecto serán de gran utilidad para futuros desarrollos.

Referencias

divcode. (2023). *como INSTALAR DJANGO en visual studio code.* Obtenido de YouTube: https://www.youtube.com/watch?v=_MrsNyNJJoA

Intro. a la Programación UNGS. (2024). *GIT - Ramas (branches).* Obtenido de Youtube: https://www.youtube.com/watch?v=BRY9gamL9PE&t=1s

Intro. a la Programación UNGS. (2024). *Introducción a GIT.* Obtenido de YouTube: https://www.youtube.com/watch?v=mzHWafbVRyU&t=1s

Intro. a la Programación UNGS. (2024). *Presentación del TP: galería de imágenes de la NASA | IP | 1er semestre 2024.* Obtenido de Youtube: https://www.youtube.com/watch?v=bKcTNzzjzCA&t=2s

ungs-ip. (2024). *ip-public-repo.* Obtenido de GitHub: https://github.com/ungs-ip/ip-public-repo