

Licenciatura en Sistemas

Trabajo Práctico

Buscador de personajes

Introducción a la Programación

(2do semestre 2024)

INTEGRANTES:

Luca Sanchez sanchezluca604@gmail.com

Jonathan Zarate jonizarate00@gmail.com

Matias Pallero matiaspallero2@gmail.com

Introducción

El siguiente trabajo apunta a la resolución de diversos ejercicios dentro de una aplicación web, el objetivo principal es el funcionamiento de un buscador de personajes sobre la serie Rick y Morty.

Desarrollo

En principio, lo que se necesitó para realizar este proyecto fue instalar una serie de extensiones para que la aplicación web pueda ser modificada y ser puesta a prueba de manera correcta. Empezamos analizando todos los archivos, leyendo los comentarios para averiguar en dónde podríamos empezar, también requirió mucho aprendizaje autodidacta y de investigación de cada uno. El trabajo no hubiese sido llevado a cabo sin la importante presencia de varias reuniones virtuales. Empezamos por el lado de añadir las imágenes, las convertimos en “cards” a las cuales les añadimos el borde con color dependiendo de su status, agregamos la funcionalidad del buscador y por último el log in. En el medio por supuesto hubo una variedad de dificultades, que pudimos sobrellevar gracias a la importante constancia que tuvimos que tener para no atascarnos en ningún problema. En general fue cuestión de ver el problema desde distintos puntos de vista para tener diversas soluciones.




Funcionalidades principales:

```
def getAllImages(input=None):  
  
    json_collection=transport.getAllImages(input)  
    images = []  
    for item in json_collection:  
        images.append(translator.fromRequestIntoCard(item))  
    return images
```

GetAllImages: Esta función está definida en el archivo services.py, obtiene una lista de datos crudos de la API usando transport.py.

```
def fromRequestIntoCard(object):  
    card = Card(  
        url=object['image'],  
        name=object['name'],  
        status=object['status'],  
        last_location = object['location']['name'],  
        first_seen = object['origin']['name']  
    )  
    return card
```

fromRequestIntoCard: Esta función está definida en `translator.py`, transforma las imágenes en “cards”.

```
<div class="col" >
  <div class="card mb-3 ms-5"
    {% if img.status == 'Alive' %}
    style="border-color: green;"
    {% elif img.status == 'Dead' %}
    style="border-color: red;"
    {% else %}
    style="border-color: orange;"
    {% endif %}>
    <div class="row g-0">
      <div class="col-md-4">
        
      </div>
```

border color: Para cambiar el color del borde teniendo en cuenta si el personaje está vivo o muerto (o paradero desconocido) modificamos la línea 40 del archivo `home.html`. En el archivo agregamos un condicional, donde si el status del personaje está vivo, el color del borde será verde, si está muerto será rojo y si no se conoce su paradero, será naranja.

```
def search(request):
    search_msg = request.POST.get('query', '')

    # si el texto ingresado no es vacío, trae las imágenes y favoritos desde services.py,
    # y luego renderiza el template (similar a home).
    if search_msg:
        images = services.getAllImages(search_msg)
        favourite_list = []
        return render(request, 'home.html', {
            'images': images,
            'favourite_list': favourite_list,
        })
    else:
        return redirect('home')
```

search: Utiliza un condicional, en donde si el texto ingresado no es vacío, busca entre la lista de personajes el nombre ingresado en el buscador, si la búsqueda coincide, devuelve una lista de resultados con las cards de los personajes. Si el texto es vacío, se muestra la galería inicial.

```
def saveFavourite(request):
    fav = translator.fromTemplateIntoCard(request)
    fav.user = get_user(request)

    return repositories.saveFavourite(fav)
```

saveFavourite: Esta función transforma la solicitud del usuario en una card que es asignada al usuario correspondiente.

```
def getAllFavourites(request):
    if not request.user.is_authenticated:
        return "usuario no resgistrado"
    else:
        user = get_user(request)

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

        for favourite in favourite_list:
            card = translator.fromRepositoryIntoCard(favourite)
            mapped_favourites.append(card)

        return mapped_favourites
```

getAllFavourites: Esta función busca todos los favoritos del usuario y los almacena en una lista vacía.

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

deleteFavourite: Elimina el favorito de la lista mediante su “id”.

Conclusiones:

Todos los integrantes de este trabajo estamos de acuerdo en lo mucho que nos enseñó, tanto a ser autodidactas como a familiarizarnos mucho más con el lenguaje utilizado y, también, a conocer otros distintos lenguajes, funcionalidades y entre otras cosas. Principalmente, lo que creemos que fue más relevante a la hora de realizar este trabajo fueron las reuniones y la manera eficiente en la que nos organizamos para realizarlo.