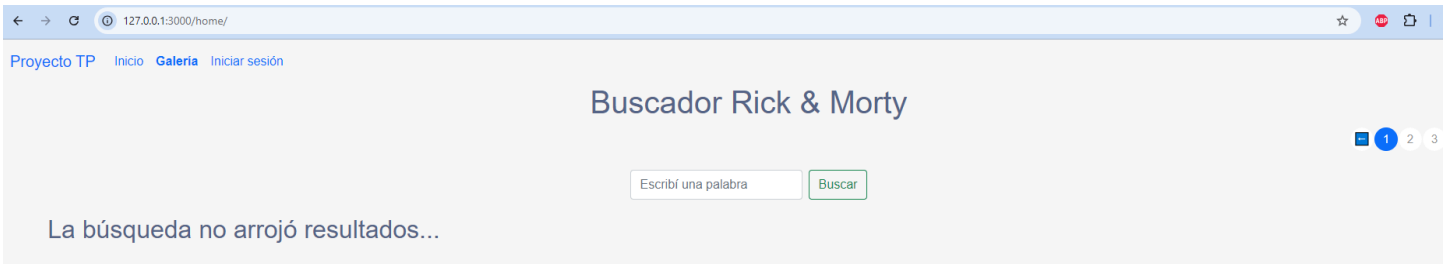
INFORME DEL PROYECTO

Proyecto página web	Rick and Morty	**Periodo de informe**
Profersores	Velcic Fernando y Bagnes Patricia	21/10/2024 - 25/11/2024
Informe de	Fernández Ezequiel y Zabala Damian	

Introduccion

Este proyecto consiste en hacer los arreglos al proyecto de la página web de Rick and Morty para que pueda funcionar correctamente. Inicialmente se ve así:



Se espera que implementemos los conocimientos adquiridos y aprendamos más a medida que avanzamos en el proyecto , finalizando el proyecto se espera que la página web se vea así:

Puntos principales

- En el archivo 'views.py' debemos arreglar la función 'home' para que muestre las imágenes.

```python
def home(request):
    images = []
    favourite_list = []
    return render(request, 'home.html', { 'images': images, 'favourite_list': favourite_list })
```

- En el archivo 'services.py' debemos modificar la función 'getAllImages' que transforma la imágenes al formato(card) que se espera al finalizar el trabajo.

```python
def getAllImages(input=None):
    # obtiene un listado de datos "crudos" desde la API, usando a transport.py.
    json_collection = []

    # recorre cada dato crudo de la colección anterior, lo convierte en una Card y lo agrega a images.
    images = []

    return images
```

- En el archivo 'home.html' se espera que modifiquemos una parte del código y agreguemos los bordes de las cards para concluir el trabajo y se muestre como se espera.

Modificaciones y arreglos

- En el primer punto debemos arreglar la función home del archivo views.py. nos trajo complicaciones para llamar a la función 'getAllImages' porque no sabíamos que había que llamarlo desde el archivo services.py, luego de entender eso la función quedó de la siguiente manera.

```
def home(request):
    images = services.getAllImages()
    if request.user.is_authenticated:
        favourite_list = services.getAllFavourites
    else:
        favourite_list =[]
    return render(request, 'home.html', { 'images': images, 'favourite_list': favourite_list })
```

- Los arreglos del archivo 'services.py' para que la función muestre las imágenes o cards, en la función 'getAllImages' requerimos conocimiento adquirido en el punto anterior y nuevo conocimiento que fue la implementación de un from ... Import... Para poder llamar a la función que necesitamos desde 'transport.py', y crear un bucle for para la variable 'images'. Con una pista del profesor que se necesitaría unas 3 o 4 líneas de codigo llegamos a la conclusión de crear un bucle.

```python
from ..transport import transport
from ..persistence import repositories
from ..utilities import translator
from django.contrib.auth import get_user

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

- En el archivo 'home.html' adquirimos nuevos conocimientos para arreglar el código y que funcione correctamente. Primero arreglar comparaciones que no eran (comparar un booleano con un string) comparaba true == 'alive' lo que no daba error pero siempre adoptaba el estado 'alive' como status de la card. Modificamos esa parte y agregamos los bordes de las cards según su estado.

```
<div class="card mb-3 ms-5
    {% if img.status == 'Alive' %}  border border -1 border-success
    {% elif img.status == 'Dead' %}  border border -1 border-danger
    {% else %}  border border -1 border-warning
    {% endif %}"
    style="max-width: 540px;">
```

```
<strong>
            {% if img.status == 'Alive' %} 🟢 {{ img.status }}
            {% elif img.status == 'Dead' %} 🔴 {{ img.status }}
            {% else %} 🟠 {{ img.status }}
            {% endif %}
        </strong>
```

Puntos extra

Buscador de la pagina

Hacer que el buscador funcione

En el archivo 'views.py'. El buscador está implementado y debemos hacer que al buscar una palabra clave arroje los resultados si es que existen , en caso contrario arroja un mensaje que no encontró resultados. Con los conocimientos que adquirimos hasta este se nos hizo más evidente que debíamos hacer y el único incoveniente que encontramos fue agregar el parámetro 'input' en el archivo 'services.py'.

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

Arreglar el login

Este punto con los conocimientos que ya teníamos solo tuvimos que prestar atención a los 'import', había uno que no estaba siendo utilizado 'logout' , agregamos el logout a la función 'éxit' del archivo 'views.py' y hacemos que nos redireccione a la pantalla de inicio.

```
@login_required
def exit(request):
    logout(request)
    return render( request, 'index.html')
```

Apartado de favoritos

Este punto nos lleva bastante tiempo y fue muy entretenido de hacer ya que requirió no solo conocimiento y lógica, sino también mucho testeo y hallar los errores que iban surgiendo, probar nuevas formas de hacerlo, buscar en otros archivos y probar nuevamente, hasta llegar a una solución que era muy similar a la que se esperaba y que funcione correctamente.

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

def getAllFavourites(request):
    if not request.user.is_authenticated:
        return []
    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
```

```python
@login_required
def getAllFavouritesByUser(request):
    favourite_list = []
    favourite= services.getAllFavourites(request)
    for favorito in favourite:
        favourite_list.append(favorito)
    return render(request, 'favourites.html', {'favourite_list': favourite_list })

@login_required
def saveFavourite(request):
    services.saveFavourite(request)
    favourite_list = services.getAllFavourites(request)
    image= services.getAllImages()
    return render(request,'home.html', {'images': image,'favourite.html': favourite_list})
```

Página terminada

Luego de haber terminado los pasos anteriormente mencionados, la vista de la página web se ve así:

Conclusión

Este proyecto nos hizo explorar nuevos lenguajes, adquirir nuevos conocimientos y expandir nuestra lógica para comprender cómo funcionan ciertas funciones sin ver su código , explorar nuevos lenguajes, practicar todo lo que ya habíamos visto una vez más y también a usar mejor la lógica tanto así como disfrutar de la programación. Este trabajo nos ayudá a abrir la mente y nos impulso a querer aprender más sobre los lenguajes necesarios para hacer páginas web.

Agradecimientos

Agradecemos principalmente a nuestro profesor: Velcic Fernando que con sus explicaciones personales no hizo entender muchas formas diferentes de ver un programa.

Y también a la profesora: Bagnes Patricia quien con sus demostraciones gráficas nos ayudó a entender mejor la lógica de la programación.