
TRABAJO PRÁCTICO INTRODUCCIÓN A LA PROGRAMACIÓN - COMISIÓN VIRTUAL 2024

Buscador Rick & Morty

**Universidad Nacional de General
Sarmiento**

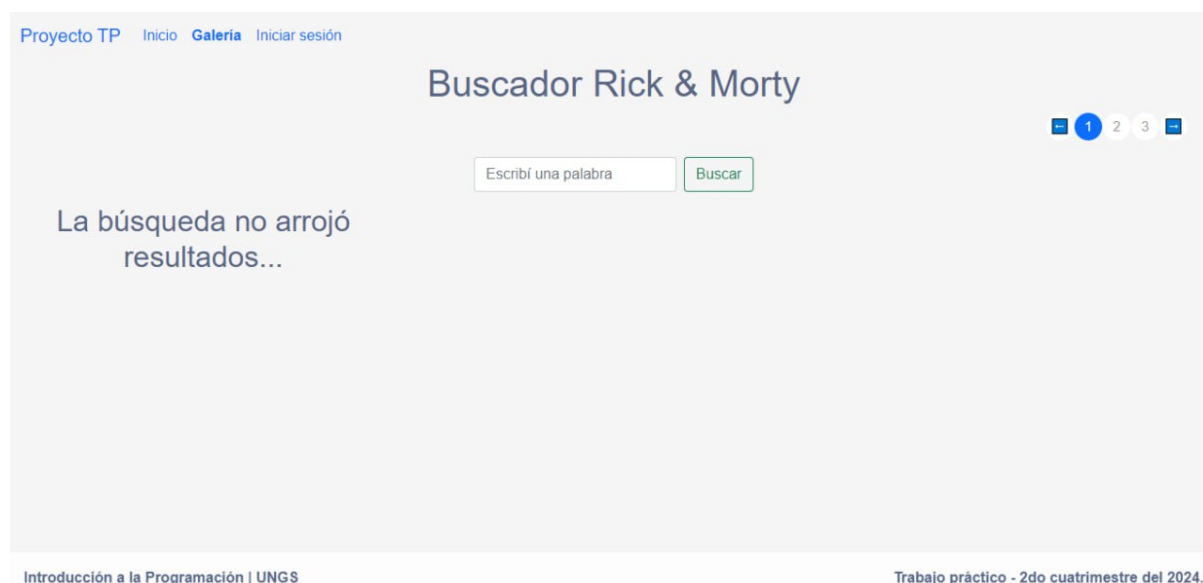
Mazzetto – Zaidan - Gomez

2do cuatrimestre 2024



El trabajo práctico consiste en el desarrollo de una página web centrada en el dibujo animado Rick & Morty, donde se encuentran tarjetas que mostrarán imágenes con información de todos personajes como el estado (vivo, muerto o desconocido), la última ubicación y el episodio inicial.

En primer lugar, al comenzar con el trabajo nos encontramos con lo siguiente:



Comenzamos con el trabajo "dibujando" las tarjetas. Para ello dentro de los archivos buscamos aquellas funciones encargadas de cumplir ese propósito. Allí teníamos que seguir desarrollando las funciones semicompletas con la información necesaria.

◦ (1) views.py:

- **home(request):** obtiene 2 listados que corresponden a las imágenes de la API y los favoritos del usuario, y los usa para dibujar el correspondiente template. Si el opcional de favoritos no está desarrollado, devuelve un listado vacío.

esta función obtiene 2 listados que corresponden a las imágenes de la API y los favoritos del usuario, y los usa para dibujar el correspondiente template.
si el opcional de favoritos no está desarrollado, devuelve un listado vacío.

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

Esta función dibuja las cartas en la Galería, pero necesita una lista de imágenes con la información correspondiente. Para eso usamos una función que nos brinda esa información y la almacenamos en esa lista. Una vez hecho esto, al abrir el servidor, se pueden ver las cartas.

◦ (2) services.py:

- `getAllImages(input=None)`: obtiene un listado, con formato de Card, de imágenes de la API. El parámetro `input`, si está presente, indica sobre qué imágenes/personajes debe filtrar/traer.

```
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
```

Esta función trae las imágenes y la información que se va a mostrar en la Galería de los personajes

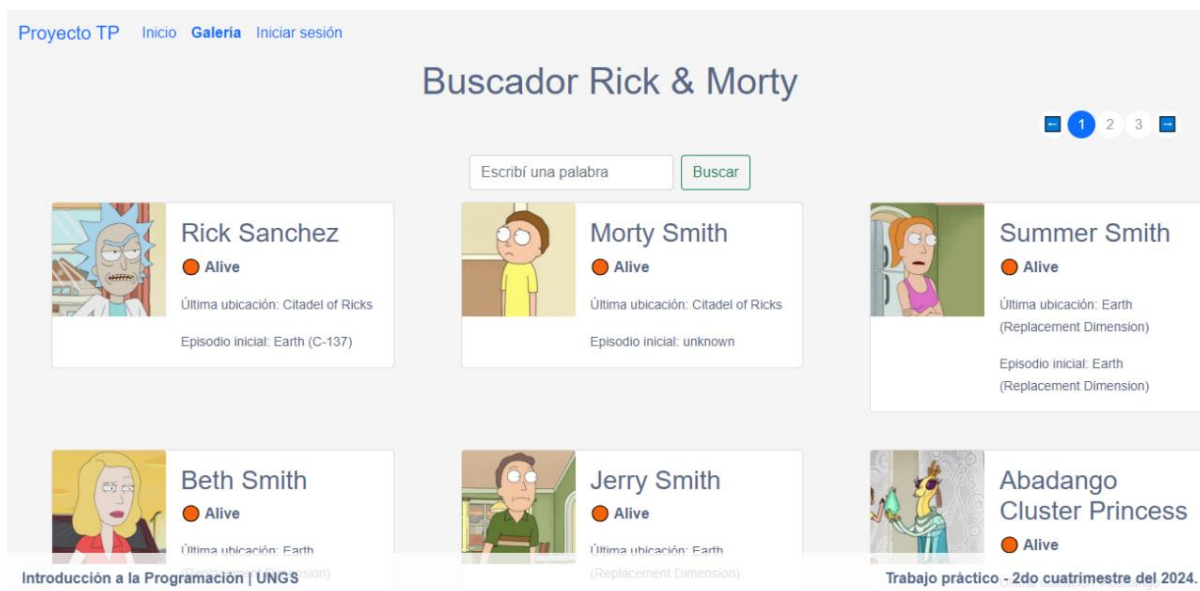
```
# comunicación con la REST API.  
# este método se encarga de "pegarle" a la API y traer una lista de objetos JSON crudos (raw).  
def getAllImages(input=None):  
>     if input is None: ...  
>     else: ...  
  
    json_collection = []  
  
    # si la búsqueda no arroja resultados, entonces retornamos una lista vacía de elementos.  
>     if 'error' in json_response: ...  
>     for object in json_response['results']: ...  
  
    return json_collection
```

```
from app.layers.utilities.card import Card
```

```
# usado cuando la info. viene de la API, para transformarla en una Card.  
def fromRequestIntoCard(object):  
    card = Card(...  
    return card
```

Se utilizaron estas dos funciones para completar la función anterior. Así trabaja correctamente.

Guardamos cada cambio e iniciamos el servidor, al hacerlo, veremos lo siguiente:



Luego continuamos con los bordes de las cartas, para ello tuvimos que identificar que parte del código era la responsable de esa parte de la estructura, para así poder modificarla. A su vez, también notamos el uso de condicionales que, al determinar el estado del personaje le otorgaba un círculo de color.

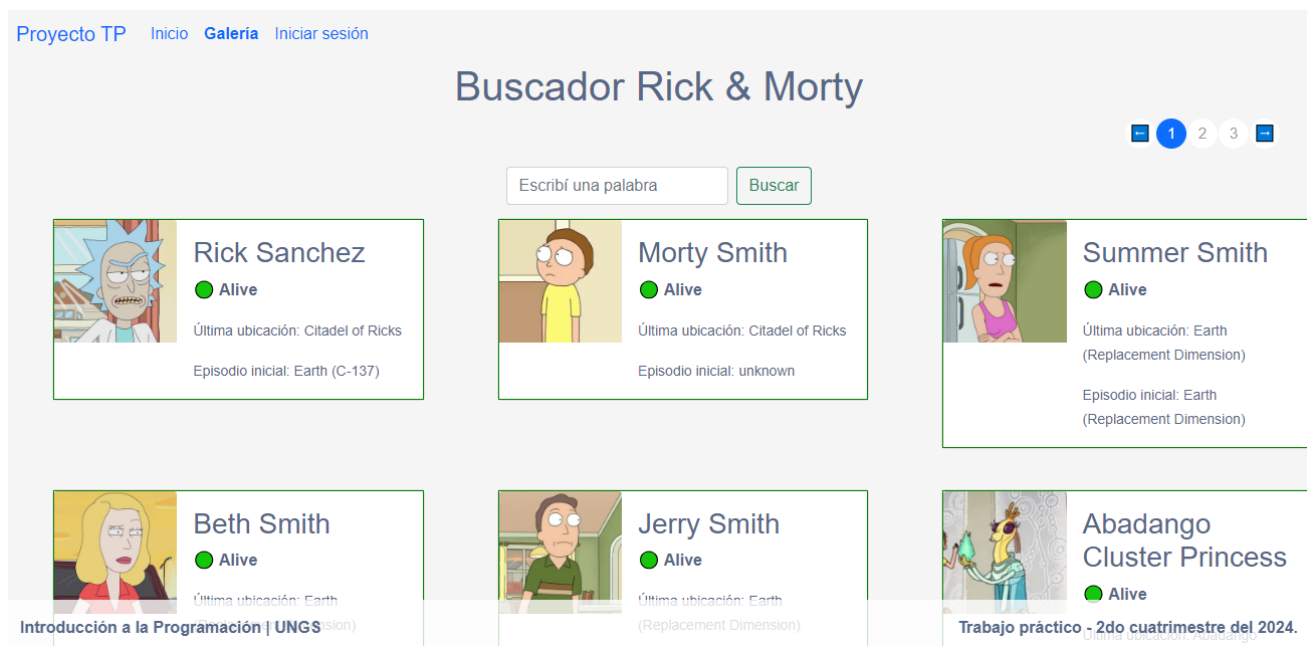
```
<div class="col-md-8">
  <div class="card-body">
    <h3 class="card-title">{{ img.name }}</h3>
    <p class="card-text">
      <strong>
        {% if img.status == 'Alive' %} ● {{ img.status }}
        {% elif img.status == 'Dead' %} ● {{ img.status }}
        {% else %} ● {{ img.status }}
        {% endif %}
      </strong>
    </p>
  </div>
</div>
```

```

<div class="row row-cols-1 row-cols-md-3 g-4">
  {% if images|length == 0 %}
  <h2 class="text-center">La búsqueda no arrojó resultados...</h2>
  {% else %} {% for img in images %}
  <div class="col">
    <div class="card mb-3 ms-5" style="max-width: 540px;">
      {% if img.status == 'Alive' %}
      <div class="card mb-3 ms-5" style="border: solid green 1px;">
        {% elif img.status == 'Dead' %}
        <div class="card mb-3 ms-5" style="border: solid red 1px;">
        {% else %}
        <div class="card mb-3 ms-5" style="border: solid orange 1px;">
        {% endif %}
        <div class="row g-0">
          <div class="col-md-4">
            
          </div>

```

Siguiendo esa lógica reutilizamos el código para modificar el color de borde de las cartas.



Al finalizar con la parte visual de las cartas, continuamos con el funcionamiento del buscador.

```

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

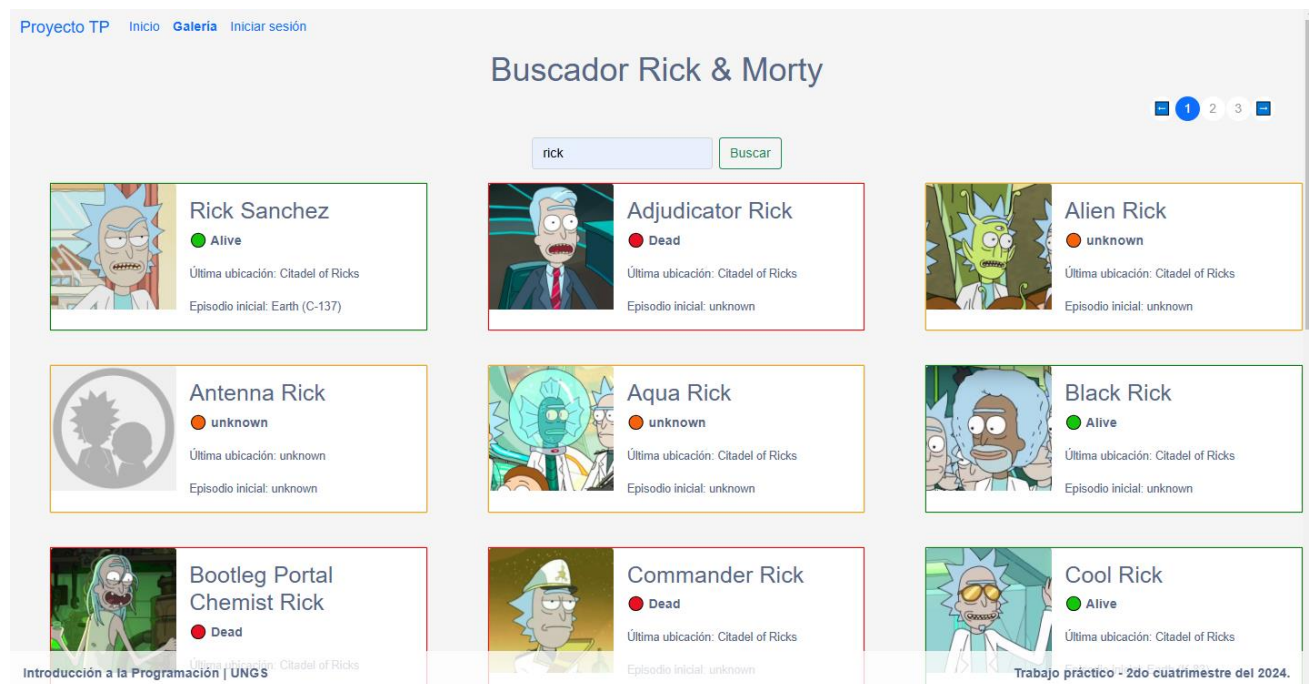
    # si el texto ingresado no es vacío, trae las imágenes y favoritos desde services.py,
    # y luego renderiza el template (similar a home).
    if (search_msg != ''):
        images = services.getAllImages(search_msg)
        favourite_list = services.getAllFavourites(request)

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

```

Para el buscador se reutilizó funciones que me dan las imágenes y los favoritos, la función

dibuja nuevamente las cartas, pero filtrando lo que el usuario ingresa en la barra de búsqueda, en este caso buscamos rick.



En cuanto al login / logout. Al ver las funciones importadas, había una sin usar.

```
from django.shortcuts import redirect, render
from .layers.services import services
from django.contrib.auth.decorators import login_required
from django.contrib.auth import logout
```

Como tiene un nombre descriptivo, lo usamos dentro de la función exit para cerrar sesión antes de dibujar de nuevo la página inicial.

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

Por último, trabajamos con la lista de favoritos. Nos encontramos con más funciones semicompletas en las cuales les otorgamos el código necesario para que trabaje correctamente.

```
# añadir favoritos (usado desde el template 'home.html')
def saveFavourite(request):
    fav = translator.fromTemplateIntoCard(request) # transformamos un request del template en una Card.
    fav.user = get_user(request) # le asignamos el usuario correspondiente.

    return repositories.saveFavourite(fav) # lo guardamos en la base.
```

Esta función se encarga de guardar una carta seleccionada como favorito y se lo asigna al usuario que corresponda.


```
# usados desde el template 'favourites.html'
def getAllFavourites(request):
    if not request.user.is_authenticated:
        return []
    else:
        user = get_user(request)

        favourite_list = repositories.getAllFavourites(user) # buscamos desde el repositories.py TODOS los favoritos del usuario (variable
        mapped_favourites = []

        for favourite in favourite_list:
            card = translator.fromRepositoryIntoCard(favourite) # transformamos cada favorito en una Card, y lo almacenamos en card.
            mapped_favourites.append(card)

        return mapped_favourites
```

Esta función se encarga de traer la lista de favoritos agregadas por el usuario

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

Esta función elimina por id a una carta favorita agregada por el usuario. Con estas funciones implementadas, se pueden usar en las siguientes funciones que son las encargadas de dibujar lo solicitado.

```
# Estas funciones se usan cuando el usuario está logueado en la aplicación.
@login_required
def getAllFavouritesByUser(request):
    favourite_list = services.getAllFavourites(request)
    return render(request, 'favourites.html', { 'favourite_list': favourite_list })
```

Esta función dibuja el apartado de favoritos después de traer la información de la lista de favoritos.

```
@login_required
def saveFavourite(request):
    services.saveFavourite(request)
    return redirect('home')
```

Esta función dibuja el apartado de galería después guardar una carta en favoritos y actualizar la lista de favoritos.


```
@login_required
def deleteFavourite(request):
    services.deleteFavourite(request)
    favourite_list = services.getAllFavourites(request)
    return render(request, 'favourites.html', { 'favourite_list': favourite_list })
```

Esta función dibuja el apartado de favoritos actualizando la lista de favoritos después de eliminar una carta de favoritos.

Una vez finalizado el desarrollo, ya que se haya iniciado sesión, el usuario podrá agregar y eliminar cartas de su lista de favoritos. Viéndose de la siguiente manera:

Escribí una palabra

Buscar




Rick Sanchez

● Alive

Última ubicación: Citadel of Ricks

Episodio inicial: Earth (C-137)

✓ Ya está en favoritos




Morty Smith

● Alive

Última ubicación: Citadel of Ricks

Episodio inicial: unknown

✓ Ya está en favoritos




Summer Smith

● Alive

Última ubicación: Earth (Replacement Dimension)

Episodio inicial: Earth (Replacement Dimension)

✓ Ya está en favoritos




Beth Smith

● Alive

Última ubicación: Earth (Replacement Dimension)

Episodio inicial: Earth (Replacement Dimension)

♥ Añadir a favoritos




Jerry Smith

● Alive

Última ubicación: Earth (Replacement Dimension)

Episodio inicial: Earth (Replacement Dimension)

♥ Añadir a favoritos



Abadango Cluster Princess

● Alive

Última ubicación: Abadango

Episodio inicial: Abadango




♥ Añadir a favoritos

Introducción a la Programación | UNGS

Adjudicator Rick

Trabajo práctico - 2do cuatrimestre del 2024.

Listado de FAVORITOS

#	Imagen	Nombre	Status	Última ubicación	Episodio inicial	Acciones
-		Rick Sanchez	Alive	Citadel of Ricks	Earth (C-137)	<div></div>
-		Morty Smith	Alive	Citadel of Ricks	unknown	<div></div>
-		Summer Smith	Alive	Earth (Replacement Dimension)	Earth (Replacement Dimension)	<div></div>