**Trabajo practico final / Parte 2**

Herner, Juan Manuel

El trabajo consta de una aplicación web basada en una arquitectura multicapas. Además, integra una API de la NASA para renderizar sus imágenes, en la cual he integrado funcionalidades de login, búsqueda y loading.

**Diseño**

Se unificaron la mayoría de las páginas, quedando el index como una breve presentación del trabajo, y la página home donde se presentan todas las funcionalidades de la aplicación en sí. Además, se ha rediseñado completamente la interfaz, otorgándole una estética más moderna y acorde a la temática espacial. Esto incluyó la creación desde cero de la estructura HTML respectiva y una renovación total del CSS.

**Views.py**

Se creó la función **“login\_view”** que al verificar la solicitud POST obtiene los valores enviados desde el formulario de inicio de sesión para autenticarlos y devolver un objeto en caso de éxito. Posteriormente, al autenticarse, repite la acción pero en la sesión actual y redirecciona a la misma pagina.

*def* login\_view(*request*):

    if request.method == 'POST':

        username = request.POST.get('username')

        password = request.POST.get('password')

        user = authenticate(request, *username*=username, *password*=password)

        if user is not None:

            login(request, user)

            return redirect('home')

    return render(request, 'home.html')

Se modificó la función **“getAllImagesAndFavouriteList”** para que solamente obtenga las imágenes en favoritos si el usuario fue anteriormente autenticado, en caso contrario devuelve una lista vacia.

*def* getAllImagesAndFavouriteList(*request*):

    images = services\_nasa\_image\_gallery.getAllImages()

    if request.user.is\_authenticated:

        favourite\_list = services\_nasa\_image\_gallery.getAllFavouritesByUser(request)

    else:

        favourite\_list = []

    return images, favourite\_list

Se modificó la función **“search”** para que obtenga el contenido ingresado en el input por el usuario y halle imágenes que coincidan con el termino de búsqueda. Posteriormente se renderizaran las imágenes junto a toda su información.

*def* search(*request*):

    images, favourite\_list = getAllImagesAndFavouriteList(request)

    search\_msg = request.POST.get('query', '')

    if search\_msg:

        images = services\_nasa\_image\_gallery.getImagesBySearchInputLike(search\_msg)

    return render(request, 'home.html', {'images': images, 'favourite\_list': favourite\_list, 'search\_msg': search\_msg})

Se modificaron las funciones para que llamen a distintas funciones o métodos ya creados en services\_nasa

@login\_required

*def* saveFavourite(*request*):

    if request.method == 'POST':

        services\_nasa\_image\_gallery.saveFavourite(request)

    return redirect('home')

@login\_required

*def* deleteFavourite(*request*):

    if request.method == 'POST':

        services\_nasa\_image\_gallery.deleteFavourite(request)

    return redirect('getAllFavouritesByUser')

@login\_required

*def* exit(*request*):

    logout(request)

    return redirect('home')

**Interactividad JS**:

Se añadió una función llamada **“spinner”** para que la pagina al estar cargando el contenido o mientras los formularios de búsqueda o inicio de sesión sean ejecutados hagan visible al spinner y al acabar estos eventos desaparezca mediante el uso de clases.

*function* spinner(*params*) {

    document.addEventListener('DOMContentLoaded', *function*() {

        document.getElementById('loading\_\_spinner').style.display = 'flex';

        window.addEventListener('load', *function*() {

            document.getElementById('loading\_\_spinner').style.display = 'none';

        });

*var* form = document.getElementById('buscador');

*var* form2 = document.getElementById('sesion');

        form.addEventListener('submit', *function*() {

            document.getElementById('loading\_\_spinner').style.display = 'flex';

        });

        form2.addEventListener('submit', *function*() {

            document.getElementById('loading\_\_spinner').style.display = 'flex';

        });

        window.onload = *function*() {

            document.getElementById('loading\_\_spinner').style.display = 'none';

        };

    });

}

spinner()

Se añadió la función **“seleccionastronauta”** para que la imagen del astronauta colgando no interfiera en las selecciones o clicks del usuario, además de impedir su selección.

*function* seleccionAstronauta() {

    astronauta.addEventListener("dragstart", *e* *=>* {

        event.preventDefault(); // Previene el arrastre de la imagen

    });

}

seleccionAstronauta()

Se añadió una función llamada **“menuInteractivo”** para que la web tenga un menú hamburguesa y pueda visualizarse la navegación de forma responsive.

*function* menuInteractivo() {

    activado = false

    menu.addEventListener("click", *e* *=>* { //abre y cierra el menu

        if (activado == false) {

            header\_\_nav.classList.add("menu\_\_abierto");

            activado = true

        } else {

            header\_\_nav.classList.remove("menu\_\_abierto");

            activado = false

        }

    });

}

menuInteractivo()

Se añadió una función llamada **“desplazamientoBarra”** que seleccionara el thumb del scrollbar y tomara su “posición” guardando la anterior en otra variable, si la posición disminuye respecto a la anterior rotara la imagen del cohete hacia abajo, en caso contrario seguirá normalmente hacia arriba.

*function* desplazamientoBarra() {

*var* ultimaPosicionDesplazamiento = window.pageYOffset;

    window.addEventListener("scroll", *function* () {

*var* pulgar = document.querySelector("::-webkit-scrollbar-thumb"); // "selecciona" la barra de desplazamiento

*var* posicionActualDesplazamiento = window.pageYOffset; // obtiene la posicion actual

        if (posicionActualDesplazamiento > ultimaPosicionDesplazamiento) {

            document.body.classList.add("scrolling"); //rota 180 grados la imagne

        } else {

            document.body.classList.remove("scrolling"); //vuelve a su posicion original

        }

        ultimaPosicionDesplazamiento = posicionActualDesplazamiento; //actualiza la posicion previa

    });

}

desplazamientoBarra()

Se añadió una función llamada **“menuPerfil”** con la misma finalidad del menú hamburguesa, tiene la utilidad de visibilizar u ocultar el login al hacer click sobre el perfil.

*function* menuPerfil() {

    activado = false

    menu\_\_perfil.addEventListener("click", *e* *=>* {

        if (!activado) {

            activado = true

            header\_\_fondo.classList.add("visible");

            header\_\_caja\_\_perfil.classList.add("visible");

        }

    })

    login\_\_cerrar.addEventListener("click", *e=>*{

        if (activado) {

            activado = false

            header\_\_fondo.classList.remove("visible");

            header\_\_caja\_\_perfil.classList.remove("visible");

        }

    })

}

menuPerfil()

**Services\_nasa\_image\_gallery**

Se modificaron todas las funciones presentes de forma repetitiva para que recorran los json y sean convertidos en objetos dentro de nasacard.

# capa de servicio/lógica de negocio

from ..transport import transport

from ..dao import repositories

from ..generic import mapper

from django.contrib.auth import get\_user

*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 = [mapper.fromRequestIntoNASACard(obj) for obj in json\_collection]

    # 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

*def* getImagesBySearchInputLike(*input*):

    return getAllImages(input)

# añadir favoritos (usado desde el template 'home.html')

*def* saveFavourite(*request*):

    fav = mapper.fromTemplateIntoNASACard(request)

    fav.user = get\_user(request)

    return repositories.saveFavourite(fav)

# usados en el template 'favourites.html'

*def* getAllFavouritesByUser(*request*):

    if not request.user.is\_authenticated:

        return []

    user = get\_user(request)

    favourite\_list = repositories.getAllFavouritesByUser(user)

    mapped\_favourites = [mapper.fromRepositoryIntoNASACard(fav) for fav in favourite\_list]

    return mapped\_favourites

*def* deleteFavourite(*request*):

    favId = request.POST.get('id')

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

También se añadió el sistema de paginación importando paginator

from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger

mostramos hasta un máximo de 6 imágenes

   paginator = Paginator(images, 6)

    page = request.GET.get('page')

    try:

        images = paginator.page(page)

    except PageNotAnInteger:

        # Si la página no es un entero,muestra la primera página

        images = paginator.page(1)

    except EmptyPage:

        # Si la página está fuera de rango muestra la última página

        images = paginator.page(paginator.num\_pages)