



# Proyecto: GIFOS

---

## Descripción general

Este proyecto tiene como objetivo integrar los conocimientos previos de maquetado con los fundamentos de la programación aprendidos en este módulo. El desafío será desarrollar un catálogo de GIFs que contará con funcionalidad de búsqueda, interacción con una API externa y captura de video.

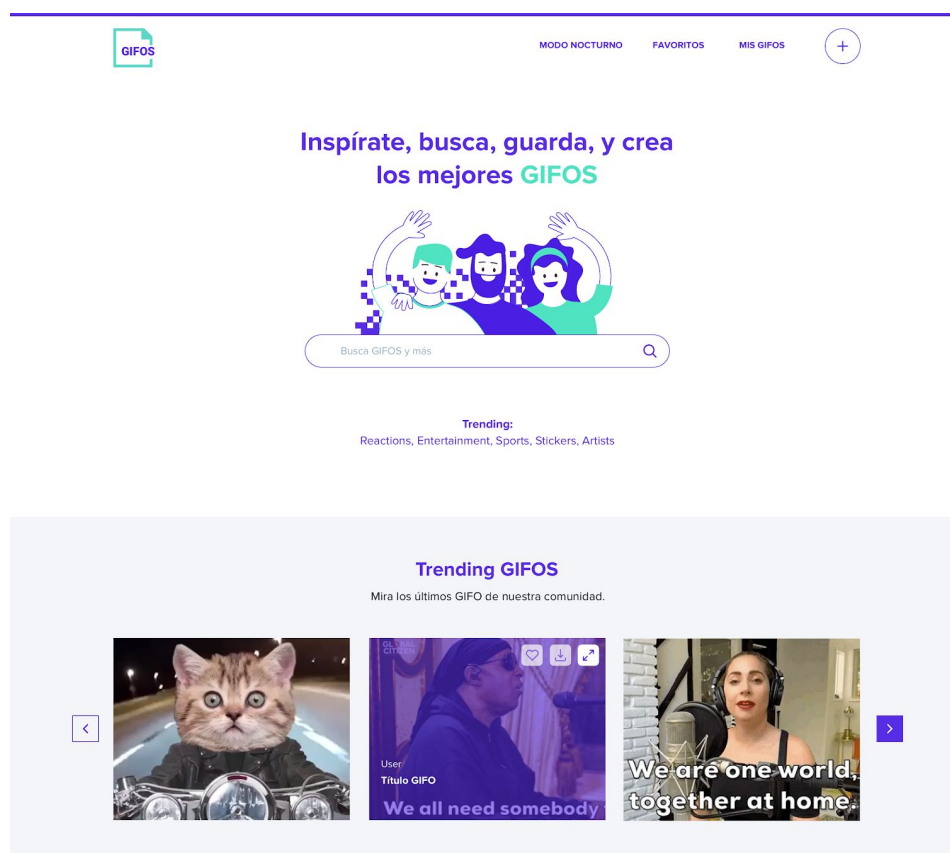
El objetivo del proyecto es replicar un esquema laboral de un Desarrollador, por lo que el diseño debe tomar las imágenes de referencia (verás que tendrás más de una opción en algunas partes y elementos sobre los que podrás decidir cuál realizar).

¡Recuerda que la realización y entrega del Proyecto es individual!



# Entregable

Aplicación web que funcione como un catálogo de GIFs, que permita funciones de búsqueda, interacción con una API externa y captura de video.



## Recursos para desarrollar el proyecto

- [UI Kit:](#) utiliza los assets de este kit para armar tu sitio web. Verás que en las condiciones para aprobar tanto el diseño diurno como el nocturno deben ser iguales a los de referencia (cuando aplique la posibilidad de elegir entre opciones, deberán elegir alguna de las provistas).
- [Prototipo funcional Desktop:](#) podrás navegar las pantallas para entender mejor lo que tienes que lograr mediante tu código (¡las consignas de más abajo te ayudarán a ir paso a paso!).
- [Prototipo funcional Mobile:](#) navega la aplicación Mobile desde tu computadora con este prototipo.





---

## Uso de librerías

No está permitido el uso de librerías, plugins o cualquier otro recurso que no esté especificado en esta guía, ya que el objetivo del proyecto es validar los conocimientos de base.

## Consignas

A continuación, te ofrecemos una guía de pasos sugeridos para construir tu proyecto (¡no es obligatorio que los sigas estrictamente en este orden!).

### a) Estructura de proyecto y themes

Comenzarás creando una estructura básica de carpetas para mantener todos tus archivos ordenados. A continuación crea tus archivos HTML en el raíz de tu proyecto, sin olvidar de vincular todos los recursos importantes, incluyendo scripts, fuentes y hojas de estilo.

Trabaja en el maquetado de tu sitio, utiliza material de test para simular el contenido que obtendrás luego. Puedes utilizar cualquier GIFs o una imagen de placeholder y también inventa tus textos, ya que aquello que pongas luego será reemplazado a través de las llamadas con `fetch` a la API de Giphy.

#### Checkpoint

Comprueba el correcto funcionamiento de todas las páginas en los diferentes viewports.

🧐 *Tip: ocúpate primero de construir una base sólida. ¡La funcionalidad la irás agregando eventualmente!*

### b) Integración con una API externa

Antes de continuar con el desarrollo, Giphy pide como requisito para utilizar su API que crees una “aplicación” y generes tu key.

Para generar esta API Key:

1. Visita la página de Giphy developers <https://developers.giphy.com>
2. Crea una cuenta





### 3. Sigue los pasos para generar la key

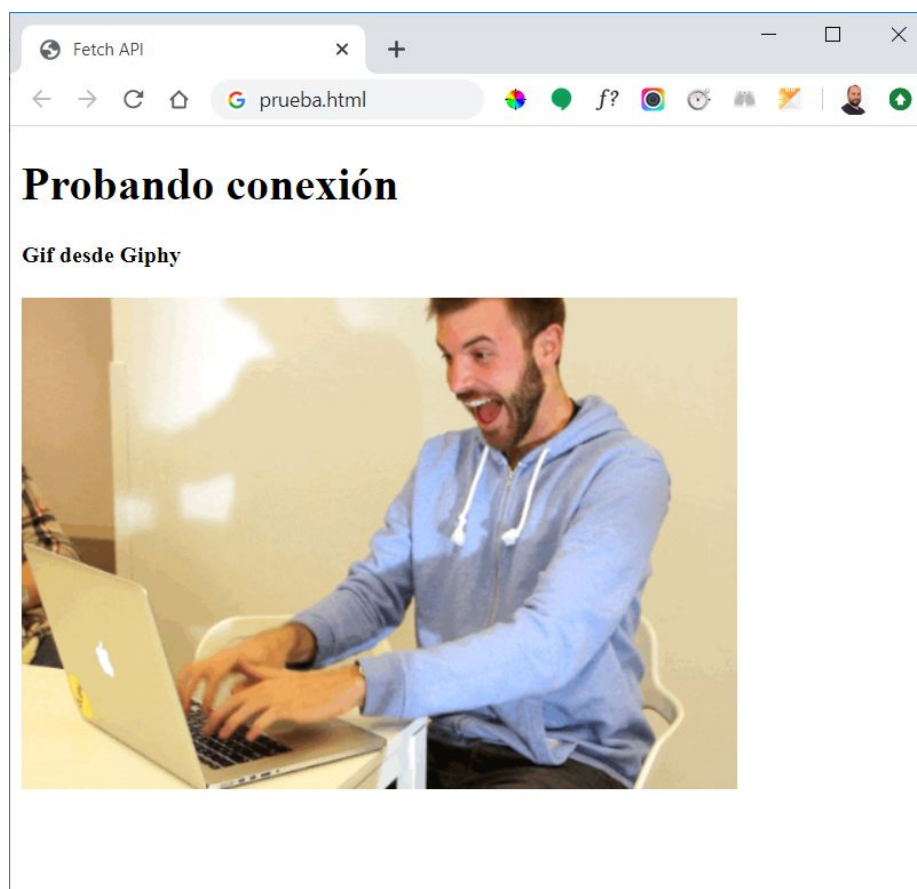
Al terminar estos pasos, deberías tener tu API Key: se trata de un string de 32 caracteres conformado por números y letras mayúsculas y minúsculas.

Teniendo esta información ya puedes hacer requests a la API. Revisa el listado de [Endpoints](#) disponibles para hacer tus requests.

Antes de comenzar a hacer requests desde tu proyecto, te recomendamos crear un archivo HTML por fuera del proyecto y probar tu conexión con la AP (con esto te aseguras que tu API Key y conexión con Giphy funcionan correctamente). Para ello, utiliza cualquier endpoint y muestra algún resultado por pantalla..

Para hacer todos tus requests usa `fetch`, revisa la [documentación oficial](#) y el material de asincronía, promesas, callbacks y los ejemplos que visto a lo largo del Sprint.

Tu HTML podría verse así:





### c) Empieza por los trendings y la barra de búsqueda

Te sugerimos comenzar por los *trendings* de la home: haz una llamada al endpoint correspondiente donde solo deberás enviar tu API Key y mostrar los datos.

Crea dinámicamente una tarjeta (¡ya la tenías maquettata!) por cada resultado que te provea el endpoint.

### d) Continúa con la acción de buscar

Continúa con la barra de búsqueda: como primera medida, resuelve el autocompletar. Cuando el **usuario** empieza a tipear debes realizar un request al endpoint que te devolverá algunas sugerencias.



Cuando el **usuario** hace click en alguna de las sugerencias del menú desplegable, la barra autocompleta el input buscado y dispara automáticamente dicha acción de búsqueda.

Clickeando la cruz en el margen derecho del input de búsqueda, el **usuario** puede borrar cualquier contenido de la barra, es decir, lo deja en blanco para una eventual búsqueda desde cero.

> Crea el request al endpoint cuando el usuario presione enter o haga click en la lupa.

- Captura el value del input para enviarlo a la API y que realice tu búsqueda.





Una vez que tengas en tu poder la información, empuja el bloque de *trending* al final de la pantalla y haz que aparezca el bloque de resultados con el título de la búsqueda que el **usuario** realizó y la grilla de resultados con 12 GIFs animados.

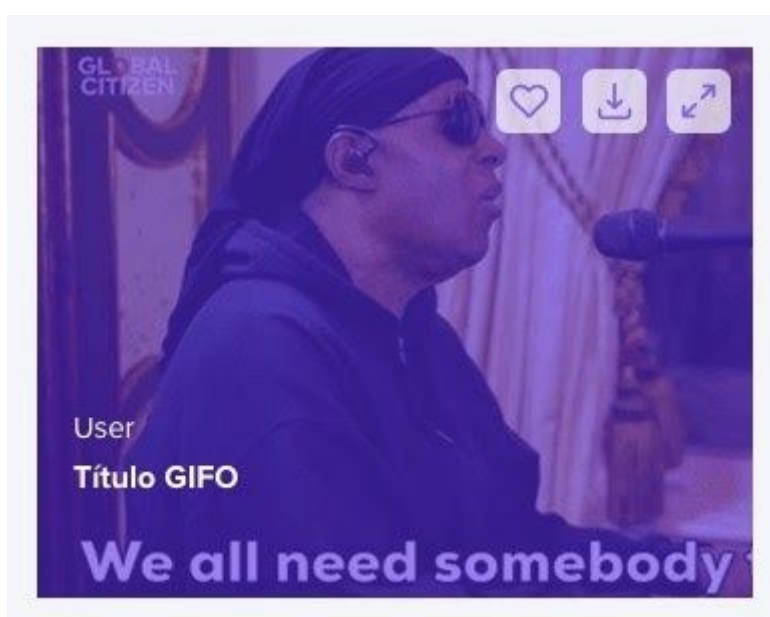
Con la grilla desplegada, tienes 2 opciones para mostrarle más resultados al **usuario**. Elige la que más te guste:

1. Un botón de 'Ver más': la acción es mostrar 12 resultados más cada vez que se apriete (es decir, que a medida que el **usuario** elija esta opción una y otra vez, deberán mostrarse en total 24, 36, 48 –y así sucesivamente– resultados por pantalla).  
El botón siempre debe quedar expuesto y desaparecer cuando no hay más resultados para mostrar.
2. Paginado: realiza una página y muestra 12 GIFs en todo momento. A través de esta acción, el **usuario** podrá avanzar y retroceder las veces que quiera.  
*Ayuda: el total de resultados dividido 12 te dará la cantidad de páginas.*

Encuentra el diseño de ambos en la página de componentes de los assets que te entregamos.

### e) Tarjetas (GIFs)

A continuación te detallamos las 3 opciones *clickables* de botones que encontrarás cuando el usuario posiciona el cursor del mouse (*mouseover*) sobre una tarjeta:



Aquí te detallamos la funcionalidad de cada ícono.





- **Corazón:** desde este ícono el **usuario** podrá guardar el GIF en sus Favoritos. *Almacena la información en el `localStorage`. En los componentes encuentra el estado activo / inactivo.*
- **Flecha para abajo:** con este ícono el **usuario** puede descargar el GIF a su máquina. *En el `Json` encontrarás una URL de descarga directa.*
- **Flechas en direcciones opuestas:** con el ícono restante, el **usuario** podrá ampliar el GIF para verlo en tamaño original.



## DESAFÍO OPCIONAL

- Te proponemos crear un slider para pasar los GIFs de a uno.

En el diseño mobile, al hacer click sobre un GIF el **usuario** accederá directamente a la versión full screen del mismo, en donde encontrará las opciones de descargar y guardar en Favoritos.


¿Por qué? No es recomendable tener acciones de *mouseover* en dispositivos mobile.

### d) Favoritos

En la página de Favoritos debes mostrar todos los GIFs que el **usuario** haya guardado.

- Levanta toda la información guardada en el `localStorage` y muéstrala por pantalla.

Al igual que los resultados de búsqueda, tenemos una grilla de 12 GIFs. Elige el botón de 'Ver más' o el recurso de paginado para esta sección.

 **Atención:** Si el **usuario** no tiene GIFs guardados, deberás mostrarle el diseño correspondiente.

### Checkpoint

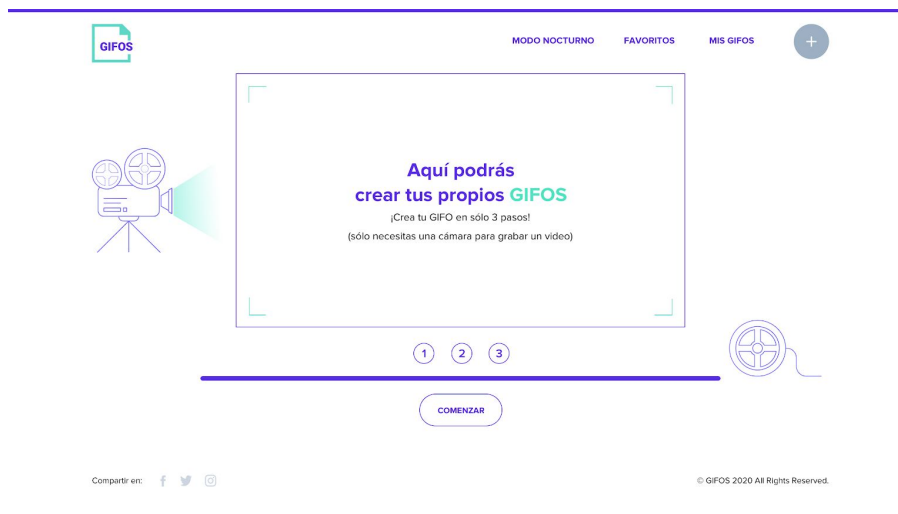
Hemos avanzado bastante. Es momento de hacer una pausa y revisar que todo funcione como esperamos.

Revisa una por cada una de las funcionalidades tanto en desktop como en mobile antes de continuar.



## e) Creación de GIFOS

El proceso consta de tres simples pasos: pedir permisos para acceder y abrir la cámara, grabar, y subir a Giphy. A continuación te detallamos cada uno de los pasos.

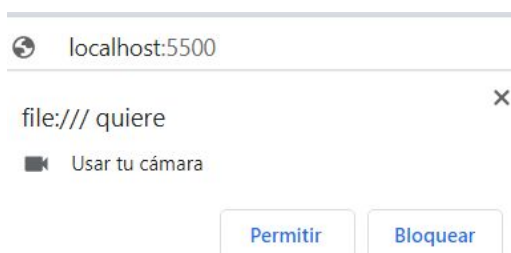


### 1. Permisos para acceder a la cámara

Para que el usuario pueda capturar video, primero debe obtenerlo por algún medio. Para esto, utiliza el método `navigator.mediaDevices.getUserMedia`.

- Puedes encontrar más información al respecto en la [documentación oficial](#).

Debes permitirle al usuario aceptar el uso de la cámara desde la confirmación del navegador:



El segundo requisito para capturar video en tu página es tener un contenedor que pueda reproducirlo. En este caso, una etiqueta `<video/>`.







- Agrega entonces (si no tienes ya) una etiqueta `<video/>` dentro del layout.
- Puedes declarar una función que inicie las funciones de captura.
- Dentro de la función, llama al método `navigator.mediaDevices.getUserMedia` y pásale las opciones de video que desees:

```
function getStreamAndRecord () {  
  
    navigator.mediaDevices.getUserMedia({  
  
        audio: false,  
  
        video: {  
  
            height: { max: 480 }  
  
        }  
  
    })  
  
    .then(function(stream) {  
  
        video.srcObject = stream;  
  
        video.play()  
  
    })  
  
}
```

- El método devuelve una promesa, por lo cual debes manejar la respuesta de manera asíncrona. Una vez que tienes tu stream, puedes utilizarlo como src de la tag `<video/>` y llamar al método `play()` para que comience a reproducirse.
- Ahora debes implementar que la función `getStreamAndRecord` se llame cada vez que presionas el botón “Grabar” y que el video aparezca o desaparezca según haga falta, esto se puede resolver con listeners e interactuando con el DOM.

### Checkpoint

Si sigues las instrucciones correctamente, deberías poder lograr que la cámara empiece a funcionar cada vez que el **usuario** presiona el botón





“Grabar”.

## 2. Grabar video

Ahora que es posible ver el stream de tu webcam, el siguiente paso es capturarlo.

- Para este paso utiliza una librería externa llamada [recordRTC](https://recordrtc.org/).

Para evitar descargar la librería en tu computadora, usa un CDN, es decir, una versión de la librería que se encuentra online.

- Puedes encontrar el link en <https://recordrtc.org/>.

Para comenzar a interactuar con la librería, debes crear un objeto recorder. Este objeto recibirá opciones y cuenta con varios métodos que puedes utilizar para grabar, por ejemplo:

```
recorder = RecordRTC(stream, {  
  type: 'gif',  
  frameRate: 1,  
  quality: 10,  
  width: 360,  
  height: 240,  
  onGifRecordingStarted: function() {  
    console.log('started')  
  },  
});
```

Para entender qué hace cada una de las opciones puedes volver a revisar la documentación. El recorder cuenta con muchos métodos, pero en este momento solo importan `startRecording` y `stopRecording`.


- `StartRecording`, como su nombre lo indica, comienza la grabación y no requiere de ningún parámetro extra.





- `StopRecording` recibe como parámetro un *callback*, donde se indica qué hacer con la información grabada una vez que se para la grabación.

Sabiendo esto puedes continuar con los siguientes pasos: cuando el **usuario** aprieta el botón 'grabar', debes permitirle crear un nuevo recorder (o sea que le dices que inicie la grabación con `startRecording`); cuando aprieta el botón 'stop', debes llamar a `stopRecording` pasándole como callback tu función anteriormente definida.

 Es importante en este paso agregar todos los cambios del DOM necesarios para que los botones muestren cuándo se está grabando y cuándo no.

### Checkpoint

Al finalizar estos pasos deberías ser capaz de iniciar una grabación y pararla, esto se puede confirmar a través del `console.log` que coloquaste al finalizar la grabación.

## 3. Generar un archivo para subir (upload)

Tu proceso de grabación está funcionando, pero... ¿cómo se accede al archivo creado para subirlo a Giphy?

Para realizar esto primero, es necesario revisar dos conceptos: el método `getBlob` y `FormData`.

- `getBlob()` es un método del objeto recorder creado con `RecordRTC` que permite acceder a los datos grabados. Un blob es una manera de guardar datos que eventualmente se puede transformar en un archivo para ser leído por el sistema operativo. Puedes profundizar más sobre ellos [aquí](#).
- [FormData](#) es un objeto que permite darle formato clave valor a tu información para enviarla de manera ordenada a través de body de un POST. Para agregar información se utiliza un método del objeto llamado `.append` que recibe dos parámetros clave y valor.

Ahora solo queda crear el archivo y organizar la información para enviar.

- Puedes crear el objeto `FormData` de la siguiente manera:

```
let form = new FormData();
```

E incluir la grabación al form usando el método `.append`:





```
form.append('file', recorder.getBlob(), 'myGif.gif');
```

Como puedes ver, `append` recibe “file” como clave, el blob como segundo parámetro y, en tercer lugar, un nombre de archivo, en este caso elegimos `myGif.gif`.

Con estos pasos ya tienes lista la información para subir a la API de Giphy.

### Checkpoint

Puedes confirmar que el `FormData` se está creando de manera correcta utilizando otro método de `FormData`: `.get()`.

Este método recibe como parámetro la clave de la información que estás buscando (en este caso la clave es “file”).

Puedes hacer un `console.log()` del método y revisar el contenido en la consola del navegador. Para este código:

```
let form = new FormData();

form.append('file', recorder.getBlob(), 'myGif.gif');

console.log(form.get('file'))
```

Deberías obtener una respuesta parecida a:

```
File {name: "myGif.gif", lastModified: 1562549678745,
lastModifiedDate: Sun Jul 07 2019 22:34:38 GMT-0300
(Argentina Standard Time), webkitRelativePath: "", size:
1973490, ...}
```

Si has pasado el checkpoint, quiere decir que tienes toda la información para que el usuario pueda subir su GIF a Giphy.

- Revisa el [endpoint de upload](#) para saber cómo enviarle la información.

Si tu subida es exitosa, el endpoint te devolverá el ID de GIF recién enviado.

- Almacena este dato en el `localStorage`. Lo necesitarás luego para mostrarlo en la sección Mis Gifs,





## DESAFÍOS OPCIONALES

- *Realiza animaciones con los objetos que están alrededor del video.* Puedes jugar con las luces de la cámara, con la rueda o las transiciones entre un paso y otro.
- En mobile solo es posible acceder a la cámara en un sitio seguro (https). *Sube tu desarrollo a un sitio seguro, toma los componentes de desktop y crea tu propia versión mobile para subir los GIFs.* [Git Pages](https://pages.github.com/) es una excelente opción para subir tu desarrollo.

### f) Mis GIFOS

El objetivo de Mis GIFOS es mostrar todos los GIFs que el **usuario** haya subido.

Recorre todos los GIFs almacenados en el `localStorage` y muéstralos por pantalla.



---

## ¡Felicitaciones!

¡Si llegaste hasta aquí, quiere decir que completaste tu sitio web!

Sube tu proyecto a algún servidor y comparte la URL con tus conocidos para que vean tu sitio web funcionando. Puedes reutilizar el mismo del proyecto de Podcast o buscar un nuevo hosting.





---

## Condiciones para aprobar

Antes de subir tu proyecto a la plataforma Acámica para ser evaluado, realiza las siguientes verificaciones (son las que los/as evaluadores/as tendrán en consideración al momento de corregir tu trabajo).

- Archivos HTML en el raíz del proyecto y validados en <https://validator.w3.org/>
  - Archivos CSS todos en una carpeta
  - Archivos JS todos en una carpeta
  - Orden: Los archivos deberán estar ordenados correctamente en carpetas para facilitar la corrección.
  - Barra de navegación superior con links funcionales.
  - Implementación de galería de GIFs traídos dinámicamente desde la [API de Giphy](#) en el buscador y *trendings*.
  - Autocompletar con sugerencias desde [la API](#) al empezar a tipear en el buscador.
  - Sugerecias de *trendings* traídas desde [la API](#) y que accionen una búsqueda al hacer click.
  - Funcionalidad de la tarjeta: Fullscreen, Descargar y Favoritear.
  - Implementación de captura de video usando RecordRTC.js.
  - POST a [la API](#) de Giphy para subir los GIFs capturados.
  - Implementar LocalStorage para guardar Mis GIFOS y Favoritos.
  - El objetivo del proyecto es replicar un esquema laboral de un Desarrollador Front-End, por lo que el diseño diurno y nocturno deben ser iguales a los de referencia (cuando aplique la posibilidad de elegir entre opciones, deberán elegir alguna de las provistas).
  - Asegurate que el ZIP contenga todos los archivos de tu desarrollo y un archivo txt en el raíz con la URL de tu proyecto live
- 
- Bonus Track (condiciones opcionales que no serán tenidas en cuenta para aprobar el proyecto):
    - Upload de GIF de Mobile
    - Slider de GIFs al ampliar un GIF
    - Animaciones cuando se sube un GIF

