



Ana Quesada García
Proyecto fin de ciclo DAW
IES Roque Amagro
Curso 2020/2021

Índice

1. Tecnologías empleadas
2. Angular
 - 2.1. Principales características de Angular
 - 2.2. Ventajas de Angular
 - 2.3. Estructura de un proyecto de Angular
 - 2.4. Módulos de Angular
 - 2.5. Componentes de Angular
 - 2.6. Instalación de Angular
3. Node
 - 3.1. Principales ventajas de Node
4. MongoDB
 - 4.1. Principales características de MongoDB
 - 4.2. Principales características de MongoDB Compass
5. Git
 - 5.1. Principales características de Git
 - 5.2. GitHub
6. Heroku
 - 6.1. ¿De qué trata la computación en la nube
 - 6.2. Características principales
 - 6.3. Principales ventajas
 - 6.4. Principales desventajas
7. Instalación de Angular paso a paso
 - 7.1. Instalación de Visual Studio Code
 - 7.2. Instalación de Git
 - 7.3. Instalación de TortoiseGit
 - 7.4. Instalar NVM y Node
 - 7.5. Instalar Angular CLI
8. Creación de un proyecto de Angular y sus componentes
9. Que va dentro de cada componente
 - 9.1. Componente Búsqueda
 - 9.2. Componente gifs-page
 - 9.3. Sidebar
 - 9.4. Index.html
10. Giphy Developers
11. Backend
 - 11.1. MongoDB
 - 11.2. Archivo .env
 - 11.3. Auth.js
 - 11.4. Helpers
12. Frontend
13. Unir frontend y backend
14. Subir a heriku
15. Cómo se usa la aplicación

1. Tecnologías empleadas

1. Angular
2. Node
3. MondoDB
4. Git
5. Heroku

2: Angular

Angular es un framework opensource desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, las webs SPA (Single Page Application).

Angular separa completamente el frontend y el backend en la aplicación, evita escribir código repetitivo y mantiene todo más ordenado gracias a su patrón MVC (Modelo-Vista-Controlador) asegurando los desarrollos con rapidez, a la vez que posibilita modificaciones y actualizaciones.

2.1: Principales características de Angular:

Velocidad y rendimiento:

- Ofrece la capacidad de crear una single page application, así los cambios son instantáneos, y hace que las webs sean más rápidas dado que gracias al diseño basado en componentes reutilizables no hay que estar continuamente recargando la página
- Otra característica que hace que Angular sea más rápido en su ejecución en el navegador es que genera código optimizado para las máquinas virtuales de javascript.

Productividad:

- Angular es un framework con uso a largo plazo, lo que hace que no se tenga que mirar a una nueva tecnología en un futuro cercano.
- Los editores e IDEs obtienen sugerencias de código inteligente, detección de errores y comentarios
- Las herramientas de línea de comandos permiten empezar a desarrollar rápidamente, añadir componentes y realizar test, así como previsualizar de forma instantánea la aplicación.

Testing y accesibilidad:

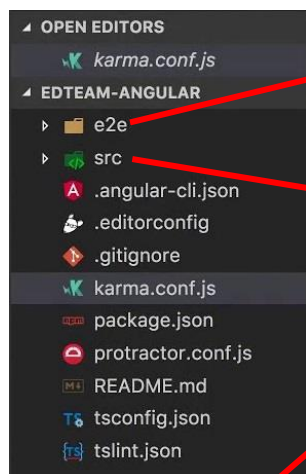
- 1) Previene la detección de errores gracias a su sistema de testing unitario, integración y e2e, con lo que se puede blindar la aplicación para evitar fallos en cualquier despliegue.

2.2: Ventajas de Angular

1. **Typescript:** Como lenguaje de programación, un superset de Javascript que, precisamente, permite usarlo de una forma mucho más sencilla que utilizando Javascript puro, y que además, directamente compila en Javascript.
2. **MVC:** A la hora de ejecutar un proyecto de cierta envergadura es que la estructura de un proyecto está ligada directamente al patrón de diseño MVC (Modelo Vista Controlador), que permite afrontar un proyecto de forma solvente y altamente escalable.
3. **Flexibilidad:** Una característica que lo hace realmente interesante es su flexibilidad en cuanto a la plataforma para desarrollar, podemos realizar apps para iOS y Android que pueden competir con apps nativas, mucho más rápido y de una forma totalmente solvente.
4. **Componentes web:** es una porción de código que es posible reutilizar en otros proyectos de Angular sin apenas esfuerzo, lo que permite un desarrollo de aplicaciones mucho más ágil.
5. **Gran soporte de herramientas:** Los principales editores e IDEs ofrecen ya extensiones para poder trabajar con este framework de la manera más cómoda posible.

2.3: Estructura de un proyecto de Angular

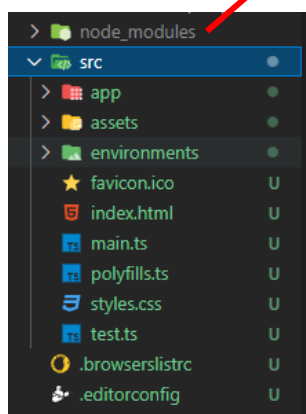
La estructura básica de un proyecto de angular es la que vemos a continuación tras crear un proyecto y abrirlo en nuestro editor de código.



e2e: esta carpeta, denominada “end to end”, engloba unas series de ficheros cuya función es la realización de test automáticos, como si un usuario interactuase con la app. Se ejecuta con el comando `ng e2e`.

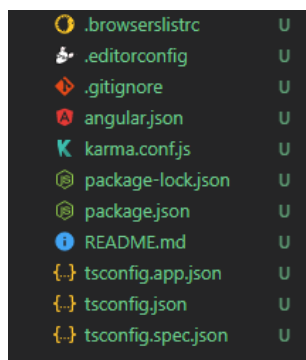
Src: es el directorio donde trabajaremos nuestros módulos. Además, es el más importante, ya que toda nuestra aplicación se creará dentro de este.

node_modules: es la carpeta que contiene todas las dependencias de nuestro proyecto



Dentro de la carpeta **SRC** tenemos los siguientes directorios de la estructura:

- **Carpeta app:** donde se ubica toda la implementación de los componentes principales, junto a su template html y archivos de estilos css.
- **Carpeta assets:** Contendrá todos los asset y archivos adicionales para hacer que el proyecto funcione.
- **Carpeta enviroments:** Donde se encuentra las configuraciones y variables de entorno para poner el proyecto tanto en desarrollo como en producción.
- **Archivo favicon:** Es el archivo del icono del proyecto.
- **Index.html:** Es el archivo de la página principal del proyecto.
- **Archivo main.ts:** Es el archivo Type Script inicial del proyecto donde podrás configurar todas las configuraciones globales del proyecto.



Otros documentos importantes son:

- **Editorconfig:** es la configuración de nuestro editor de código.
- **Gitignore:** son las carpetas o archivos que debe ignorar el git cuando lo añadamos al repositorio. Lo más habitual es ignorar node-modules
- **Angular.json:** contiene la configuración de Angular. Además, incluye rutas, versiones, etc. es la configuración de nuestra aplicación. Contiene el nombre de la app, las dependencias necesarias para su correcta ejecución y muchas otras cosas.
- **Package.json:** es la configuración de nuestra aplicación. Contiene el nombre de la app, las dependencias necesarias para su correcta ejecución y muchas otras cosas.
- **Readme.md:** aquí podemos añadir información sobre la aplicación. Este archivo es leído por GIT y los muestra en el repositorio.
- **Tsconfig.json:** contiene la configuración TypeScript.

2.4: Módulos de Angular

Un concepto muy importante en Angular son los módulos, pero ¿qué es un módulo?:

Agrupar un conjunto de artefactos Angular, como son componentes, directivas, pipes y servicios que forman parte de ese mismo módulo. Dicho esto, representa una agrupación lógica en lo que podríamos llamar área funcional de nuestra aplicación (ej. módulo de contactos, módulo de agenda...).

2.5: Componentes de Angular

El otro concepto importante a comprender en Angular son los componentes, pero ¿qué es un componente?:

Es un bloque de código re-utilizable, que consta básicamente de 3 archivos: un CSS, un HTML (también conocido como plantilla o en inglés, template) y un TypeScript (en adelante, TS). La carpeta app con la que viene Angular por defecto es un componente, aunque un tanto especial.

3: node

Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables.

HTTP es un elemento destacado en Node.js, diseñado teniendo en cuenta la transmisión de operaciones con streaming y baja latencia. Esto hace que Node.js sea muy adecuado para la base de una librería o un framework web.

El funcionamiento interno del entorno de ejecución para JavaScript, Node.js, es bastante interesante. En comparación con las técnicas tradicionales de servicio web donde cada conexión (que crea una solicitud) genera un nuevo subproceso, ocupando la RAM del sistema y regularmente maximizando la cantidad de RAM disponible, Node.js opera en un solo subproceso, utilizando el modelo entrada y entrada sin bloqueo de la salida, lo que le permite soportar decenas de miles de conexiones al mismo tiempo mantenidas en el bucle de eventos.

El nodo está completamente controlado por eventos. Resumiendo, podemos decir que el servidor consta de un subproceso que procesa un evento tras otro.

3.1: Principales ventajas de Node.js

- **Javascript incorporado:** en la plataforma Node.js, siendo un lenguaje fácil de aprender.
- **Comunidad NODE.js:** Con la implementación de plataformas de desarrollo de software como GitHub Inc., la comunidad Node.js ha crecido de forma exponencial y activa.
- **Node.js es la plataforma de software más utilizada** en este momento estando por encima en entornos de ejecución y lenguajes de programación como PHP y C a la vez que tiene un tiempo de ejecución (tiempo real) muy superior.
- **No es un código muy complejo** pero requiere muchas más líneas de codificación y mayor comprensión que el lenguaje PHP.

4: Mongo DB

Es una base de datos distribuida de código abierto, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube.

Es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico.

4.1: Principales características de Mongo DB

- **MongoDB:** es una base de datos documental, lo que significa que almacena datos en forma de documentos tipo JSON. Creemos que esta es la forma más natural de concebir los datos; frente al tradicional modelo de filas y columnas, esta es mucho más expresiva y potente.
- **Documentos JSON ricos:**
 - ♦ La forma más natural y productiva de trabajar con datos
 - ♦ Admite matrices y objetos anidados como valores
 - ♦ Trabaje con esquemas dinámicos y flexibles

- **Consultas:**
 - ♦ Lenguaje de consulta rico y expresivo que permite filtrar y ordenar por cualquier campo, independientemente de cómo esté incrustado en un documento.
 - ♦ Admite agregaciones y otros casos de uso modernos, como búsqueda de gráficos o texto, y búsqueda basada en información geoespacial.
 - ♦ Las propias consultas son también en JSON, por lo que se programan fácilmente. Olvídate de concatenar cadenas para generar consultas SQL de forma dinámica

4.2: Principales características de Mongo DB Compass

Mongo Compass es la versión de “escritorio” nos permite explorar la estructura de los documentos de las distintas colecciones que componen una base de datos de una manera fácil e intuitiva.

- **Consultas Visuales:** Permite realizar algunas queries de manera visual a base de clics de ratón sin necesidad de saber la sintaxis del CRUD de MongoDB.
- **Visualiza y explora:** GUI (del inglés Graphical User Interface) intuitiva y de fácil uso.
- **Insertar, modificar y eliminar:** Modifica tus datos con una poderosa herramienta de edición visual.
- **Extensible a través de complementos:** Compass Plugin Framework se expone como una API, lo que lo hace extensible por los usuarios. ¿Buscas una funcionalidad que no viene en Compass? Instala un complemento o crea el tuyo propio

5: git

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

5.1: Principales características de Git

- Es software libre.
- Multiplataforma.
- Con ella podemos mantener un historial completo de versiones.
- Podemos movernos, como si tuviéramos un puntero en el tiempo, por todas las revisiones de código y desplazarnos una manera muy ágil.
- Es muy rápida.
- Tiene un sistema de trabajo con ramas que lo hace especialmente potente.
- En cuanto a la funcionalidad de las ramas, las mismas están destinadas a provocar proyectos divergentes de un proyecto principal, para hacer experimentos o para probar nuevas funcionalidades.
- Las ramas pueden tener una línea de progreso diferente de la rama principal donde está el core de nuestro desarrollo. En algún momento podemos llegar a probar algunas de esas mejoras o cambios en el código y hacer una fusión a nuestro proyecto principal, ya que todo esto lo maneja Git de una forma muy eficiente.

5.2 GitHub

Creado en 2011 por GitHub y es la forma de trabajo sugerida por las funcionalidades propias de GitHub. Está centrado en un modelo de desarrollo iterativo y de despliegue constante. Está basado en cuatro principios:

- Todo lo que está en la rama master está listo para ser puesto en producción
- Para trabajar en algo nuevo, debes crear una nueva rama a partir de la rama master con un nombre descriptivo. El trabajo se irá integrando sobre esa rama en local y regularmente también a esa rama en el servidor
- Cuando se necesite ayuda o información o cuando creemos que la rama está lista para integrarla en la rama master, se debe abrir una pull request (solicitud de integración de cambios).
- Alguien debe revisar y visar los cambios para fusionarlos con la rama master

- Los cambios integrados se pueden poner en producción.
- GitHub intenta simplificar la gestión de ramas, trabajando directamente sobre la rama master y generando integrando las distintas features directamente a esta rama

6: Heroku

Heroku es una plataforma de servicios en la nube (conocidos como PaaS o Platform as a Service), cuya popularidad ha crecido en los últimos años debido a su facilidad de uso y versatilidad para distintos proyectos. Se pueden alojar aplicaciones de diferentes lenguajes de programación como Python, Java, PHP y más.

6.1: ¿De qué trata la computación en la nube?

La computación en la nube es un paradigma de la tecnología de la información. Es un modelo que permite el acceso a grupos compartidos de recursos configurables de forma ininterrumpida, tales como redes de computadoras, servidores, almacenamiento, aplicaciones y otros servicios.

Los servicios en la nube pueden suministrarse rápidamente con un mínimo esfuerzo de gestión por parte de la empresa, pues la logística de la administración de estos servicios está distribuida y funciona desde internet. La nube es una opción atractiva para las empresas con diversas necesidades. Sin embargo, las particularidades del modelo de negocio y las necesidades de cada proyecto determinarán qué proveedor de servicios en la nube será más óptimo para lo que necesitas en tu empresa.

6.2: Características principales

- En Heroku el código corre siempre dentro de un dyno que es el que proporciona a la plataforma la capacidad de cómputo, es un proceso que puede usarse para ejecutar contenido web, para ejecutar procesos batch...
- Los dynos garantiza la escalabilidad en caso de que una aplicación se convierta en viral (automáticamente se levantan varios dynos)
- Los Dynos pueden ser de tres tipos: web, worker o cron.
- WEB: se encarga del desarrollo de la aplicación web.
- WORKER: ejecuta la base de datos.
- CRON: se emplea para procesos de corta vida o conexiones Secure Shell (interprete de órdenes seguro).
- Los dynos aíslan de comunidades SSL, enrutamiento o blanqueo.
- Los dynos son transparentes y pueden ser levantados en otras máquinas de manera transparente.
- Heroku es “poliglota”, es decir, Heroku permite la utilización de diferentes lenguajes de programación.
- Heroku internamente se apoya en GitHub, pero no es necesario realizar pagos adicionales.
- A la hora de trasladar cualquier aplicación a Heroku, hay que adaptar cualquier tipo de grabación de fichero a filesystem que estuviéramos haciendo, y pasarlo a otros servicios Amazon.
- Las aplicaciones Java en Heroku, no necesitan un contenedor servlets.
- Heroku incluye Logplex, asumiendo que un log es un stream de eventos.

6.3: Principales ventajas

- Heroku es gratuito para aplicaciones de poco consumo.
- Permite el uso de diferentes lenguajes de programación.
- Es una plataforma fácil de usar.
- Integra varios servicios dentro de su estructura.
- Las actualizaciones en Heroku no afectan a nuestra plataforma informática.
- Se puede tener acceso desde cualquier lugar y dispositivo compatible con la computación en la nube.

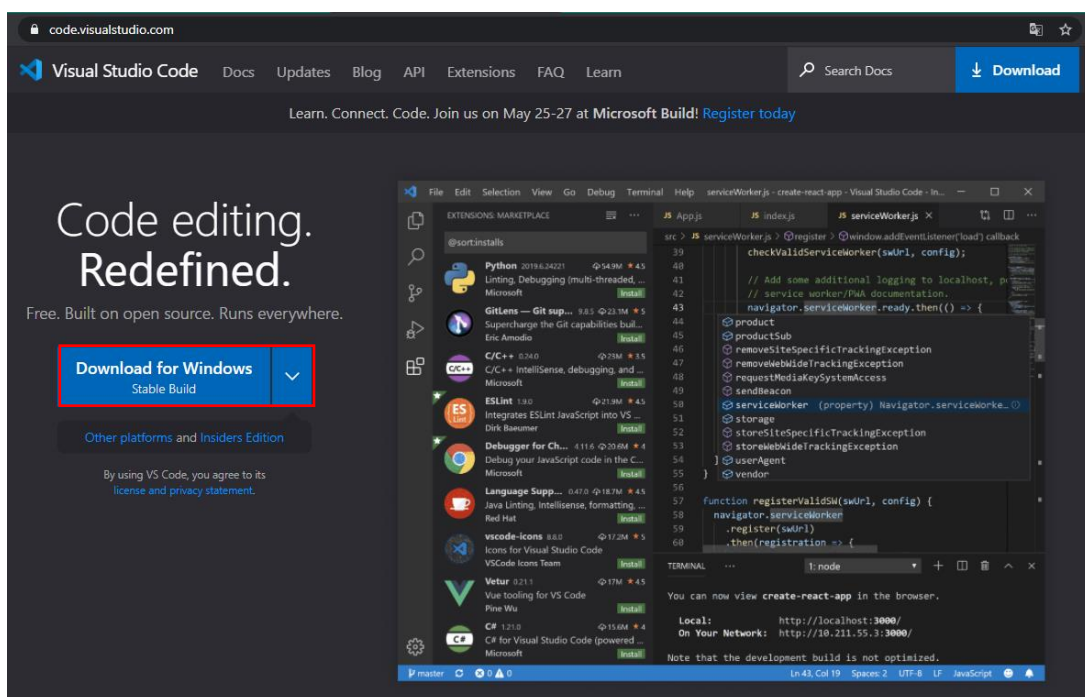
6.4: Principales desventajas

- Es necesario contratar un servicio de base de datos externo y un hosting.
- Poca personalización y mínima optimización cuando se requiere más infraestructura (Gratis).
- La disponibilidad de las aplicaciones está limitada a la disponibilidad del acceso a internet.

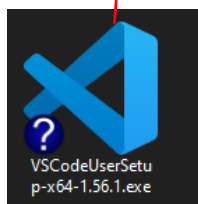
7. Instalación de Angular: Paso a paso

7.1.: Instalar Visual Studio Code

Vamos a la web de este editor de texto y pulsamos sobre el botón de **Download** con nuestro sistema operativo.



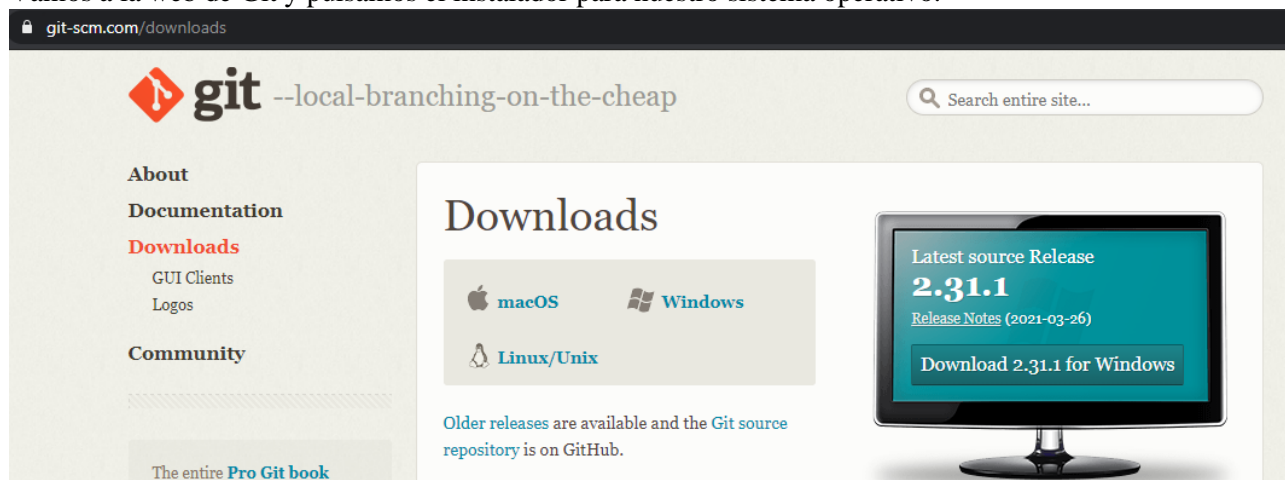
Instalador



Vamos a la carpeta de descargas de nuestro ordenador e iniciamos el instalador del programa que se nos ha descargado. Es una sencilla instalación donde seguimos las instrucciones pulsando botones de **Siguiente** para completar la instalación.

7.2: Instalación de Git

Vamos a la web de Git y pulsamos el instalador para nuestro sistema operativo.



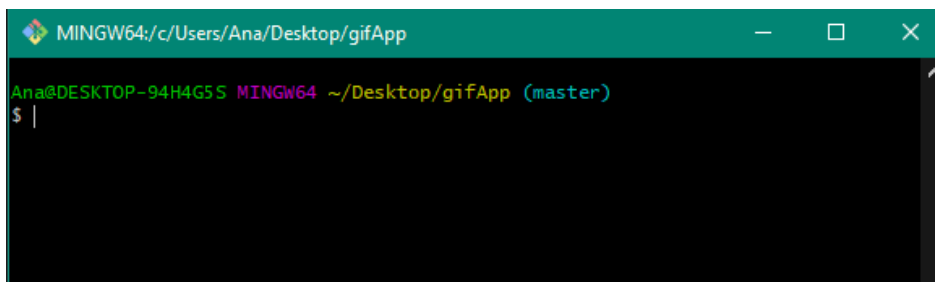
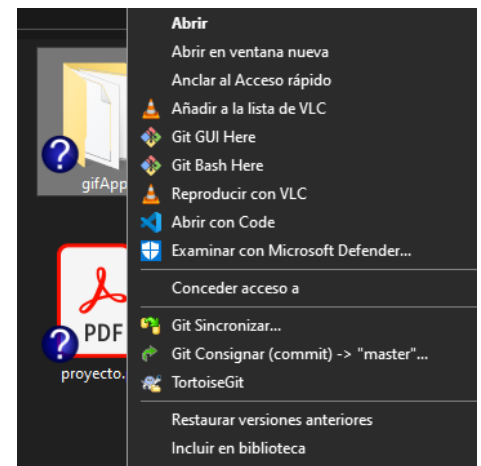
La instalación la realizaremos sin modificar casi ninguna opción de las que están marcadas por defecto.

El primer paso es aceptar las condiciones de la licencia. Después elegimos el directorio de instalación y seleccionamos los componentes marcados por defecto para instalar, verificando que esté marcada la opción “Git Bash here”.

A continuación, seguimos avanzando en las siguientes pantallas pulsando en Next para aceptar las opciones por defecto, hasta que comience la instalación de Git.

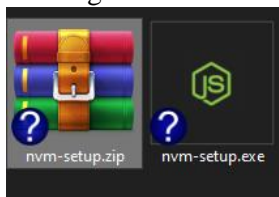
Una vez finalizada la instalación de Git, pulsamos con el botón derecho del ratón sobre la una carpeta, y nos aparecen dos nuevas opciones para abrirla.

Elegimos la opción “Git Bash Here” y comprobamos que se abre la consola y reconoce los comandos de Git, por lo que está correctamente instalado.

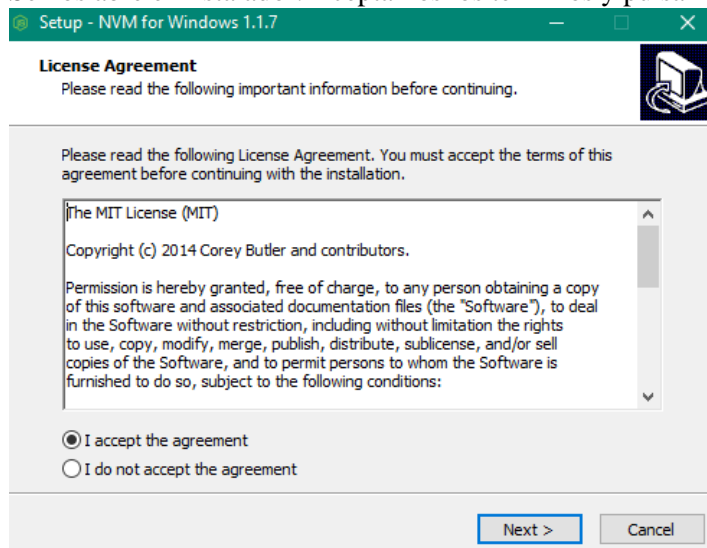


7.3: Instalación de NVM y Node

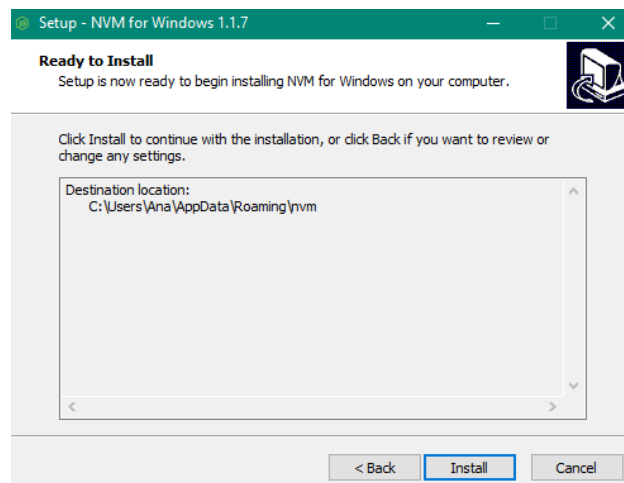
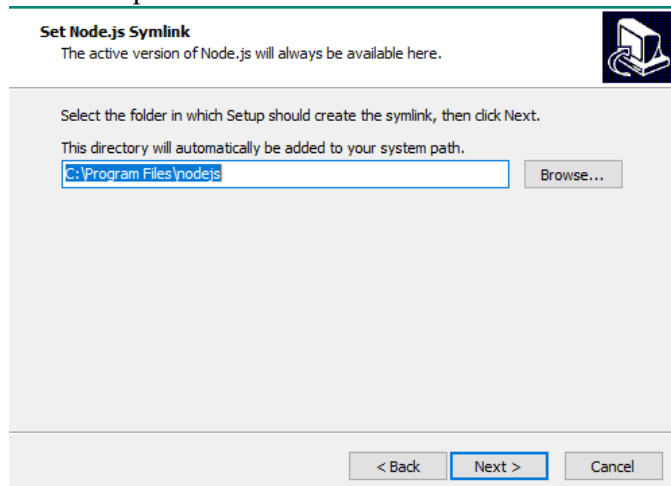
Descargamos el instalador y lo ejecutamos.



Se nos abre el instalador. Aceptamos los términos y pulsamos *Next*.



Seguimos pulsando Next hasta que lleguemos a la pantalla donde vemos el botón **Install**. Dejamos que se finalice el proceso de instalación.



Abrimos la terminal de comandos y seleccionamos la versión de Node que queremos instalar. La ventaja de hacerlo así es que podemos tener varias versiones instaladas y arrancamos la que nos interesa utilizar sin que tener que desinstalar e instalar la nueva versión.

```
C:\Users\Ana>nvm install 14.15.4
```

Con el siguiente comando arrancamos nuestra versión de Node que hemos instalado. Y ya estamos listos para instalar Angular.

```
C:\Users\Ana>nvm use 14.15.4
Now using node v14.15.4 (64-bit)
```

7.4: Instalación de Angular CLI

Con el siguiente comando instalamos la versión CLI de Angular.

```
C:\Users\Ana>npm install -g @angular/cli
[.....] - fetchMetadata: sill resolveWithNewModule mimic-fn@2.1.0 checking installable status
```

8: Creación de un proyecto de Angular

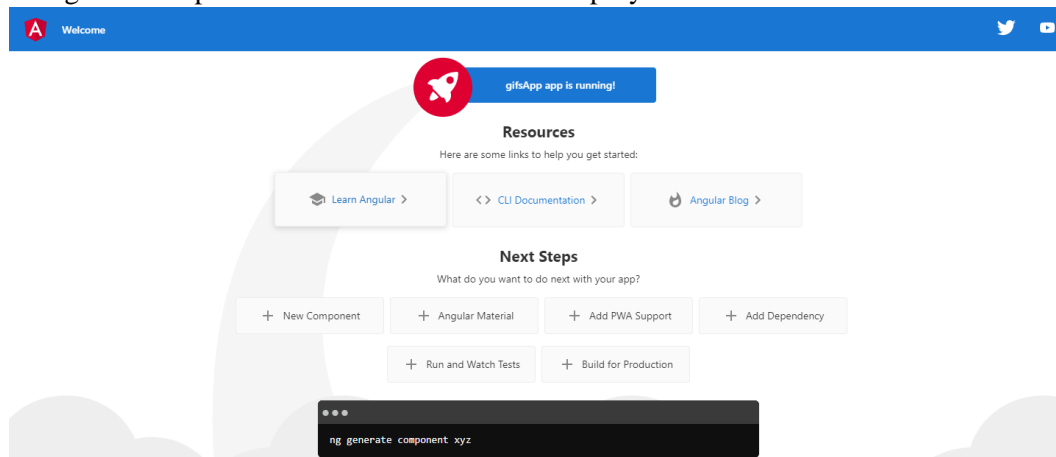
Desde la terminal de comandos nos situamos en la carpeta donde vamos a crear el proyecto, una vez finalizado el proceso abrimos nuestro proyecto en nuestro editor de código.

```
C:\Users\Ana\Desktop\angular>ng new gifsApp
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE gifsApp/angular.json (3045 bytes)
CREATE gifsApp/package.json (1070 bytes)
```

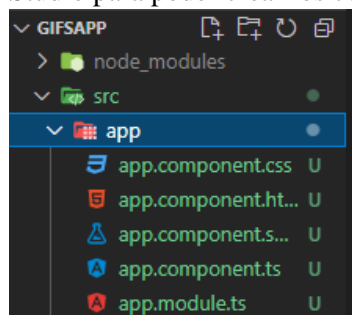
Arrancamos nuestro proyecto con el comando **ng serve -o**

```
C:\Users\Ana\Desktop\angular>cd gifsApp
C:\Users\Ana\Desktop\angular\gifsApp>ng serve -o
? Would you like to share anonymous usage data about this project with the Angular Team at Google under Google's Privacy Policy at https://policies.google.com/privacy? For more details and how to change this setting, see https://angular.io/analytics. (y/N) n
```

Una vez termine de procesar el último comando que hemos lanzado en la terminal se nos abrirá nuestro navegador web predeterminado abriendo nuestro proyecto.



Comprobamos que lo único que hay dentro de nuestro directorio APP es el componente básico llamada app.component el cual se crea de manera automática al crear nuestro proyecto. Abrimos la terminal del Visual Studio para poder crear los componentes y módulos necesarios.



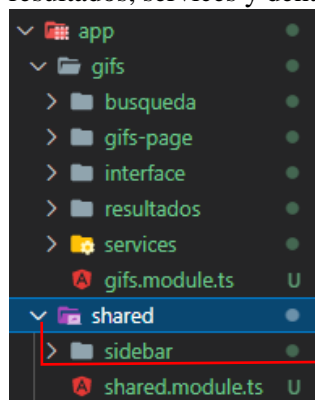
Creamos los módulos principales dentro de los cuales crearemos otros que nos ayudarán a que nuestra aplicación sea más fácil de programar.

```
PS C:\Users\Ana\Desktop\angular\gifsApp> ng g m gifs --skip-Tests
CREATE src/app/gifs/gifs.component.ts (267 bytes)
PS C:\Users\Ana\Desktop\angular\gifsApp> ng g m shared
CREATE src/app/shared/shared.module.ts (192 bytes)
```

Creamos el componente **Sidebar** dentro del directorio del módulo **Shared** que acabamos de crear.

```
PS C:\Users\Ana\Desktop\angular\gifsApp> ng g c shared/sidebar --skip-Tests
CREATE src/app/shared/sidebar/sidebar.component.html (22 bytes)
CREATE src/app/shared/sidebar/sidebar.component.ts (279 bytes)
CREATE src/app/shared/sidebar/sidebar.component.css (0 bytes)
UPDATE src/app/shared/shared.module.ts (280 bytes)
```

De esta manera creamos todos los componentes que vamos a necesitar: búsqueda, gifs-page interface resultados, services y dentro de este último creamos gifs.services



El directorio Shared nos servirá para el componente Sidebar que ha de estar siempre publicado

9: Que va dentro de cada componente

Ya tenemos la estructura base de nuestro proyecto, ahora es saber qué va dentro de cada componente. Creo que esto es fácil de deducir dado que una buena práctica de programación es que cuando le pones el nombre a un componente, servicio o módulo sea algo relacionado con la funcionalidad del mismo.

9.1: Componente Búsqueda

En el HTML creamos un input que nos servirá para hacer las búsquedas, para ello tenemos el evento `keyup.enter` para disparar el evento buscar el cual nos realizará la búsqueda.

```
<h5>Buscar:</h5>
<input
  type="text"
  class="form-control"
  placeholder="Buscar gifs..."
  (keyup.enter)="buscar()"
  #txtBuscar
>
```

En el `busqueda.component.ts` creamos la función `buscar`.

```
export class BusquedaComponent {

  @ViewChild('txtBuscar') txtBuscar!: ElementRef<HTMLInputElement>;

  constructor( private gifsService: GifsService ) {}

  buscar() {

    const valor = this.txtBuscar.nativeElement.value;

    if ( valor.trim().length === 0 ) {
      return;
    }

    this.gifsService.buscarGifs( valor );

    this.txtBuscar.nativeElement.value = '';
  }

}
```

Hemos de importar el `GifsService` para poderlo llamar y usar su contenido

Aquí vemos la función a la que estamos llamando desde el componente búsqueda en el `GifsService`.

```
buscarGifs( query: string = '' ) {

  query = query.trim().toLocaleLowerCase();

  if( !this._historial.includes( query ) ) {
    this._historial.unshift( query );
    this._historial = this._historial.splice(0,10);
    localStorage.setItem('historial', JSON.stringify( this._historial ) );
  }

  const params = new HttpParams()
    .set('api_key', this.apiKey)
    .set('limit', '100')
    .set('q', query );

  this.http.get<SearchGifsResponse>(`${ this.servicioUrl }/search`, { params } )
    .subscribe( ( resp ) => {
      this.resultados = resp.data;
      localStorage.setItem('resultados', JSON.stringify( this.resultados ) );
    });
}
```

Con esta línea estamos limitando a 10 las búsquedas que se nos muestran en el sidebar

Con esta línea estamos limitando a 100 gifs a los que nos salen en la pantalla

Este es el resultado en la vista de nuestra aplicación web.

Buscar:

9.2: Componente Gifs-Page

En este componente podemos comprobar la versatilidad de angular porque con solo una línea llamamos a la clase del componente que necesitamos.

```
<div class="row p-3">
  <div class="col">
    <!--Incluimos el componente búsqueda para que se nos muestre en esta vista-->
    <app-busqueda></app-busqueda>
  </div>
</div>

<hr />

<div class="row">
  <div class="col">
    <!--Incluimos el componente resultados para que se nos muestre en esta vista-->
    <app-resultados></app-resultados>
  </div>
</div>
```

9.3: Componente Resultados

En este componente cargamos los resultados de la búsqueda realizada por el usuario.

Con este bucle for hacemos que se nos carguen un gif tras otro hasta que llega a 100, que es el número al que hemos limitado las búsquedas

```
<div class="row">
  <div *ngFor="let gif of resultados"
    class="col-md-4 col-sm-6 animate__animated animate__fadeIn">
    <div class="card">
      <img [src]="gif.images.downsized_medium.url"
        [alt]="gif.title">
      <div class="card-body">
      </div>
    </div>
  </div>
</div>
```

En el resultado.component.ts vemos lo que alimenta al bucle for, el cual llama a un servicio creado en Gifs.service.

```
export class ResultadosComponent {

  get resultados() {
    return this.gifsService.resultados;
  }

  constructor( private gifsService: GifsService ) { }

}
```

Aquí vemos el array que estamos llamando desde el TS del componente Resultados.

```
export class GifsService {  
  
  private apiKey : string = 'CtbSNZMFpF1bw1x5KQjtqPOHnE1ema5';  
  private servicioUrl: string = 'https://api.giphy.com/v1/gifs';  
  private _historial : string[] = [];  
  
  public resultados: Gif[] = [];
```

Con estas dos líneas hacemos la conexión con la api Giphy Developers la cual nos suministra los gifs. En el punto 10 explico como se hace la conexión a la misma

Aquí vemos el resultado de una búsqueda.



9.4: Sidebar

En el componente sidebar se nos muestra el logo de la app y se nos almacenan las últimas 10 búsquedas realizadas por el usuario.

```
src > app > shared > sidebar > sidebar.component.html > div#sidebar.bg-dark.bg-dark.b...  
Go to component  
1 <div class="bg-dark border-right p-3" id="sidebar">  
2     
3   <hr class="text-white" />  
4  
5   <div class="list-group list-reset">  
6     <a *ngFor="let item of historial"  
7       (click)="buscar( item )"  
8       href="#"  
9       class="list-group-item list-group-item-action">  
10      {{ item | titlecase }}  
11    </a>  
12  </div>  
13 </div>
```

Logotipo de la aplicación

Bucle FOR que nos mantiene a la vista las últimas 10 búsquedas del usuario

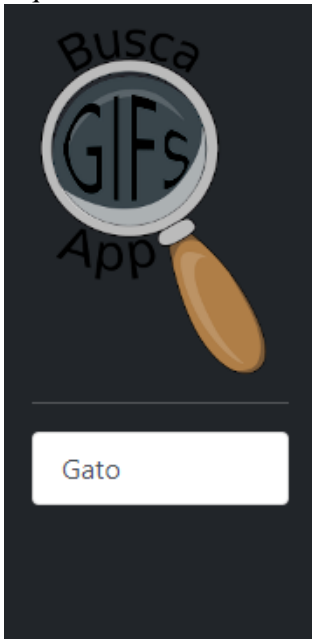
Desde el sidebar.component.ts llamamos a una función que tenemos en el gifsService.

```
export class SidebarComponent {  
  
  get historial() {  
    return this.gifsService.historial;  
  }  
  
  constructor( private gifsService: GifsService ) { }  
  
  buscar( termino: string ) {  
    this.gifsService.buscarGifs( termino );  
  }  
}
```

Aquí tenemos la función historial en nuestro gifsService.

```
get historial() {  
  return [...this._historial];  
}
```

Aquí vemos como se muestra el sidebar en



9.5: Index.html

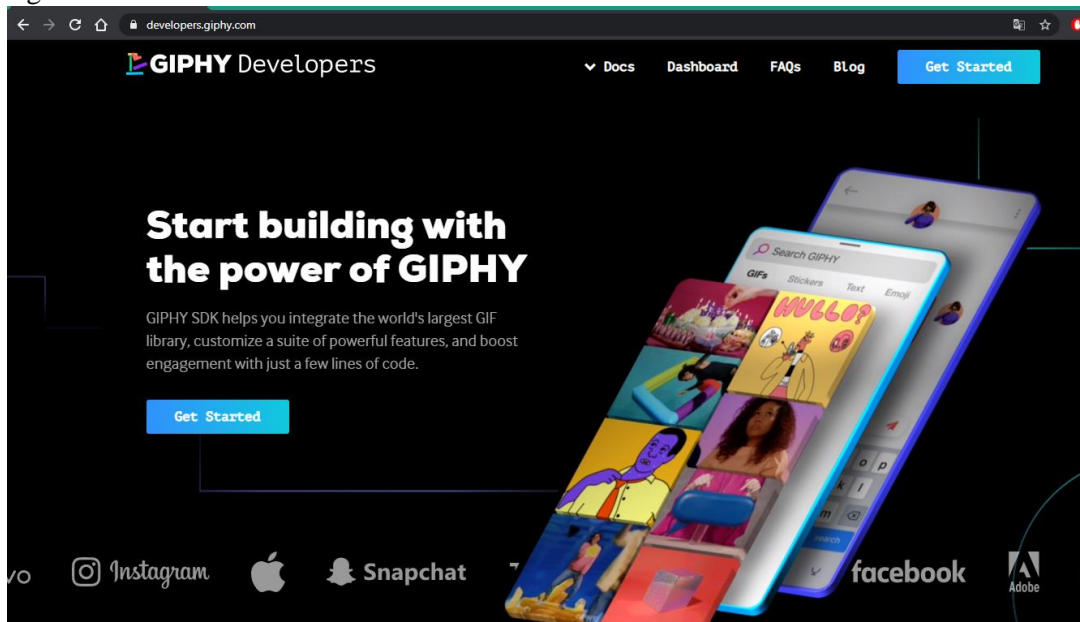
En el index.html tan solo tenemos las cdns de las reglas css de Bootstrap, y el app-root el cual carga el app.component. El app-component lo que hace es cargar todos los componentes que hemos creado dentro de este.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>GifsApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">

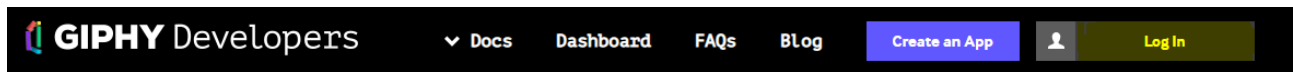
  <!-- CSS only -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-alpha3/dist/css/bootstrap.min.css"
    rel="stylesheet" integrity="sha384-CuOF+2SnTUfTwSzjCXf01h7uYhfOBuxIhGKPbfEJ3+FqH/s6cIFN9bGr1HmAg4fQ"
    crossorigin="anonymous">
  <link
    rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/animate.css@4.1.1/animate.min.css"
  />
</head>
<body>
  <app-root></app-root>
</body>
</html>
```


10: Giphy Developers

Es una api la cual suministra gifs a aplicaciones como Facebook, Whatsapp, Twitter, etc. La web es la siguiente. Pulsamos en *Get Started*

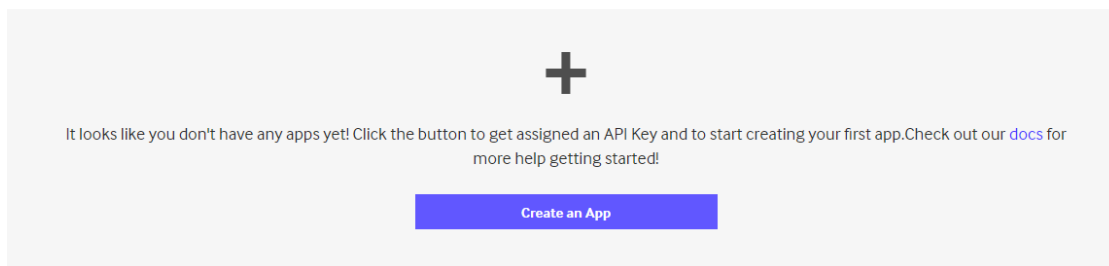


Iniciamos sesión o nos creamos una cuenta si no la tenemos.

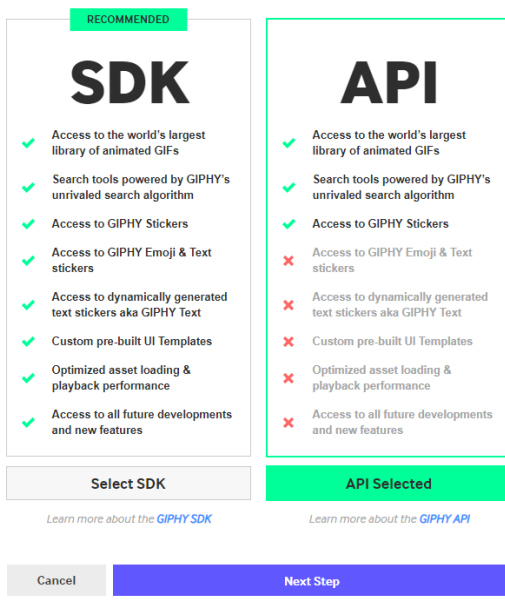


Pulsamos en Create an App.

Your Apps



Seleccionamos la opción API que es la que nos interesa para nuestra aplicación y pulsamos *Next Step*.



Ponemos el nombre de nuestra aplicación y una breve descripción de la misma y pulsamos en **Create App**

Create A New App 2/2

Just a bit more info and you'll have your **API** beta key. 🗝

Your App Name

GifsApp

App Description

App buscadora de gifs



By checking this box, I am confirming that I have read & agree to the [GIPHY API Terms](#)

Cancel

Create App

Aquí tenemos la Api KEY la cual nos servirá para cargar esta api en nuestra aplicación.

Your Apps


GifsApp **API** [Edit](#)

API Key

u3ZN8JSIK9e5gbig1UMzRVqsVfSz0wnq

[Upgrade to Production](#)

En el apartado Docs buscamos la url que ellos nos proporcionan para que la conexión se haga correctamente con las dos líneas mostradas en el apartado 9.3.

 **GIPHY Developers**

[Docs](#) [Dashboard](#) [FAQs](#) [Blog](#) [Create an App](#) [anaquesadagarcia](#)

GIPHY Search gives you instant access to our library of millions of GIFs and Stickers by entering a word or phrase. With our unparalleled search algorithm, users can easily express themselves and animate their conversations.

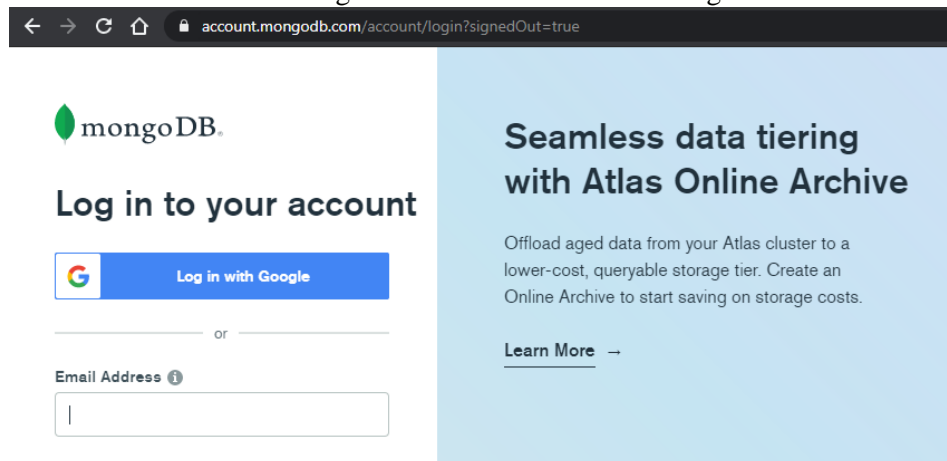
- GIPHY requires the Search API call be made from the client side.
- The search keyword should be sent to GIPHY in &q parameter in the API call. Search terms could simply be words or phrases typed by the user. Users can also add the @ sign before a GIPHY username to return content from a specific GIPHY channel. All special characters should be supported and not re-encoded in your search request.
- This keyword should be the exact terms the user searched for without any correction/enhancement
- Optionally, use the &lang parameter to indicate the language the user has typed in. This will help return unique regional content, if available. See [here](#) for supported languages.
- You may use the &rating param to tailor the response per your preferences. Read more [here](#) about content ratings

Gif URL	Sticker URL
api.giphy.com/v1/gifs/search	api.giphy.com/v1/stickers/search

11: Back end

11.1: MongoDB

Entramos en la web de mongoDB en la cual necesitamos logearnos o crear una cuenta.



Pulsamos sobre **New Project** para crear una nueva base de datos.

New Project

Ponemos el nombre que queremos darle a nuestra base de datos y pulsamos **Next**.

Create a Project

Name Your Project Add Members Next

Name Your Project

Project names have to be unique within the organization (and other restrictions).

ProyectoFp

Cancel Next

Pulsamos sobre Create Project.

Create a Project

✓ Name Your Project Add Members Go Back Create Project

Add Members and Set Permissions

Invite new or existing users via email address...

Give your members access permissions below.

anaquesadagarcia84@gmail.com (you) Project Owner

Cancel Go Back Create Project

Creamos el cluster Clusters

Find a cluster...



Create a cluster

Choose your cloud provider, region, and specs.

Build a Cluster

Elegimos la opción gratuita.

Dedicated Multi-Cloud & Multi-Region Clusters

For teams developing world-class applications that require multi-region resiliency or ultra-low latency.

- ✓ Includes all features from Shared and Dedicated Clusters
- ✓ Replicate data across clouds and regions
- ✓ Globally distributed read and write operations
- ✓ Control data residency at the document level

Create a cluster

Starting at
\$0.13/hr*
*estimated cost \$98.55/month

Dedicated Clusters

For teams building applications that need advanced development and production-ready environments.

- ✓ Includes all features from Shared Clusters
- ✓ Auto-scaling
- ✓ Network isolation
- ✓ Realtime performance metrics

Create a cluster

Starting at
\$0.08/hr*
*estimated cost \$56.94/month

Shared Clusters

For teams learning MongoDB or developing small applications.

- ✓ Highly available auto-healing cluster
- ✓ End-to-end encryption
- ✓ Role-based access control

Create a cluster

Starting at
FREE

Elegimos el servidor que queramos y la zona más cercana a nosotros. Creamos el cluster el cual solo podrá almacenar 512MB.

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

Cloud Provider & Region

AWS, Ireland (eu-west-1) ▼



★ Recommended region ⓘ

AUSTRALIA

🇦🇺 Sydney (ap-southeast-2) ★

ASIA

🇸🇬 Singapore (ap-southeast-1) ★

🇮🇳 Mumbai (ap-south-1)

EUROPE

🇮🇪 Ireland (eu-west-1) ★

🇩🇪 Frankfurt (eu-central-1) ★

NORTH AMERICA

🇺🇸 Oregon (us-west-2) ★

🇺🇸 N. Virginia (us-east-1) ★

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

[Back](#)

Create Cluster

Una vez que se nos crea el cluster pulsamos el botón Connect para realizar la conexión a la base de datos.

Clusters

SANDBOX

Cluster0

Version 4.4.6

CONNECT

METRICS

COLLECTIONS

...

CLUSTER TIER

M0 Sandbox (General)

REGION

AWS / Ireland (eu-west-1)

TYPE

Replica Set - 3 nodes

LINKED REALM APP

None Linked

Ponemos una IP y un usuario para poder crear la base de datos.

1 Add a connection IP address

IP Address

Description (Optional)

Cancel

Add IP Address

2 Create a Database User

This first user will have [atlasAdmin](#) permissions for this project.

Keep your credentials handy, you'll need them for the next step.

Username


Password

SHOW

Autogenerate Secure Password

Create Database User


Seleccionamos el tipo de conexión. En este caso nos conectaremos usando MongoDB Compass.



Connect your application

Connect your application to your cluster using MongoDB's native drivers

>



Connect using MongoDB Compass

Explore, modify, and visualize your data with MongoDB's GUI

>

Ana Quesada García

Página 20 | 27

Seleccionamos la opción según si ya tenemos MongoDB Compass instalado en nuestro terminal.

I do not have MongoDB Compass

I have MongoDB Compass

1 Choose your version of Compass:

1.12 or later

See your Compass version in "About Compass"

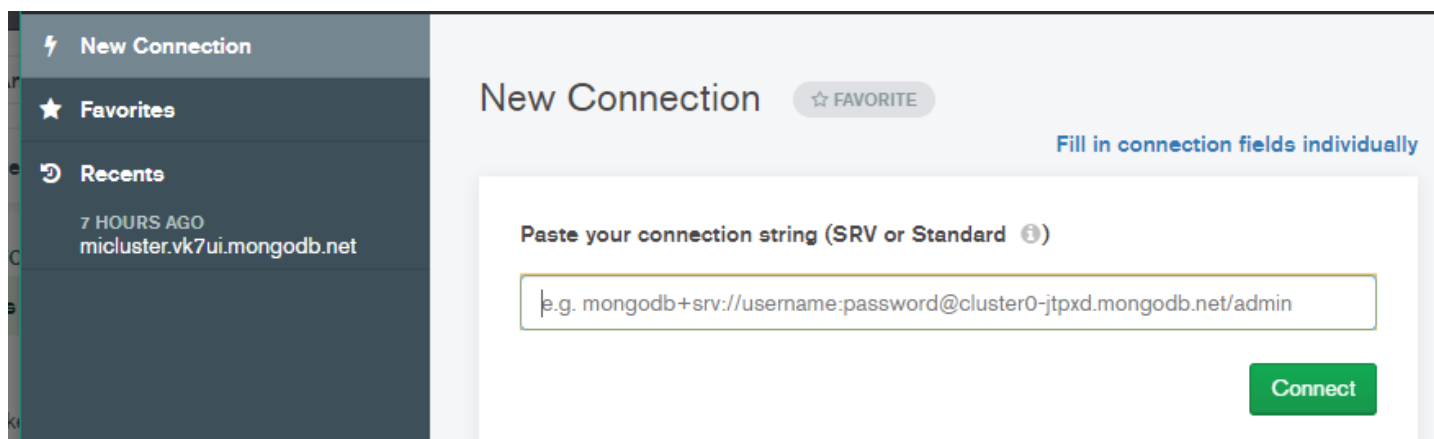
2 Copy the connection string, then open MongoDB Compass.

```
mongodb+srv://Ana:<password>@cluster0.tih75.mongodb.net/test
```

Necesitaremos esta línea para realizar la conexión

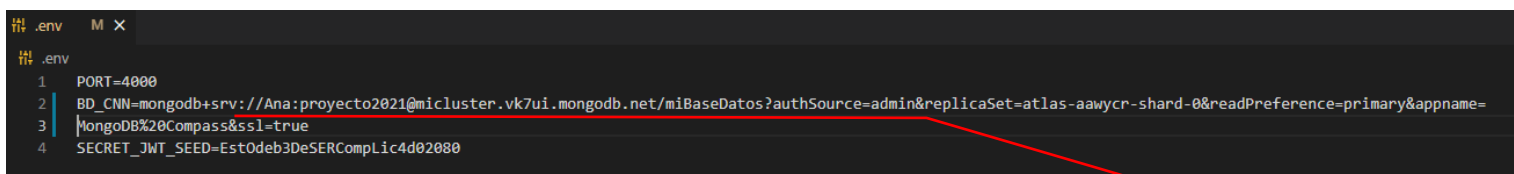
You will be prompted for the password for the **Ana** user's (Database User) username.
When entering your password, make sure that any special characters are [URL encoded](#).

Abrimos el programa en la cual introduciremos la línea seleccionada en el paso anterior, pero después de haberle realizado un par de modificaciones necesarias.



11.2 Archivo .env

Abrimos el archivo de configuración en la cual añadimos el puerto, la conexión a la base de datos y la frase que se usara para codificar el json web token.



A la url que hemos copiado le añadimos el usuario y la contraseña y más adelante en la línea hemos de añadir el nombre que le queremos dar a la base de datos

11.3: auth.js

En este archivo creamos las rutas que necesitaremos para movernos por la aplicación, como el login, editar y renovar token.

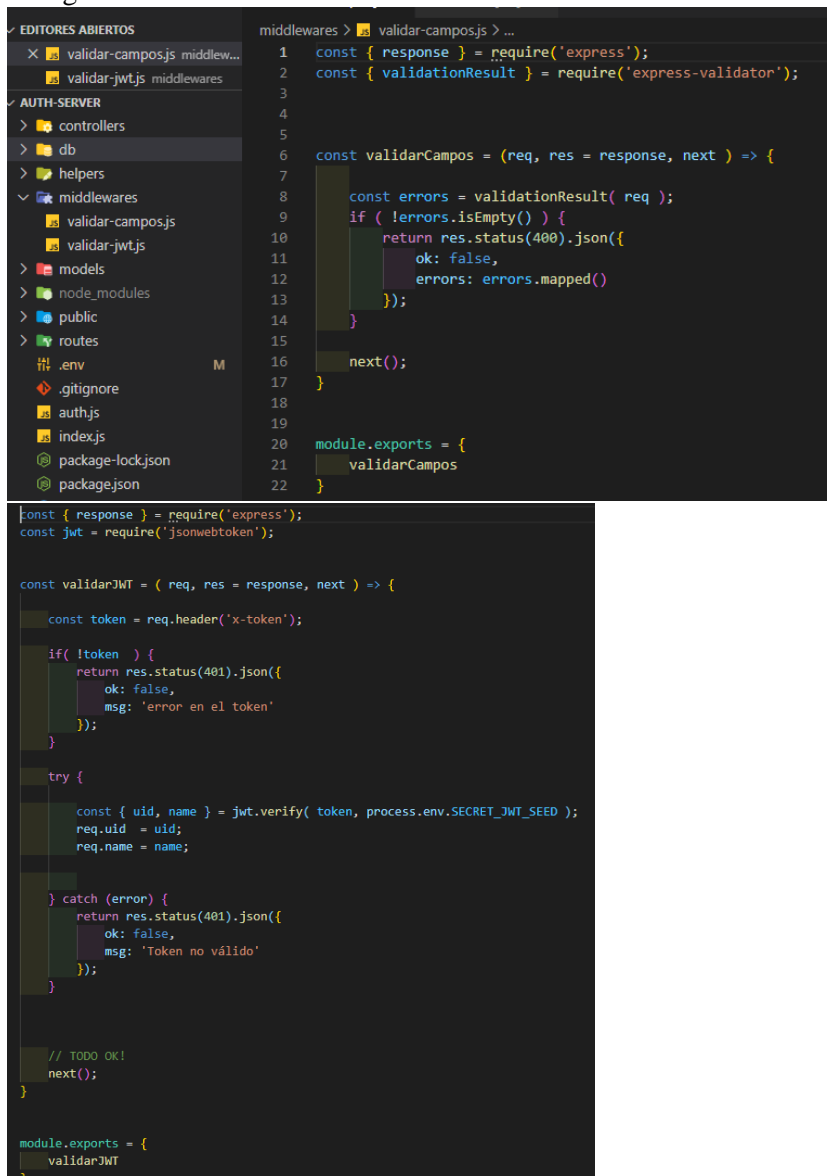
```
const router = Router();

// Crear un nuevo usuario
router.post( '/new', [
  check('name', 'El nombre es obligatorio').not().isEmpty(),
  check('mail', 'El e-mail es obligatorio').isEmail(),
  check('password', 'La contraseña es obligatoria').isLength({ min: 6 }),
  validarCampos
], crearUsuario );

// Login de usuario
router.post( '/', [
  check('mail', 'El e-mail es obligatorio').isEmail(),
  check('password', 'La contraseña es obligatoria').isLength({ min: 6 }),
  validarCampos
], loginUsuario );

// Validar y revalidar token
router.get( '/renew', validarJWT , revalidarToken );
```

En la carpeta middlewares tenemos dos middlewares los cuales hacen las peticiones del usuario al servidor. Uno es para validar los campos del formulario y el otro es para validar el json web token que es un sistema de seguridad que identifica a cada usuario. El JWT (json web token) se renueva cada vez que actualizamos nuestro navegador web.



```
const { response } = require('express');
const { validationResult } = require('express-validator');

const validarCampos = (req, res = response, next) => {
  const errors = validationResult( req );
  if ( !errors.isEmpty() ) {
    return res.status(400).json({
      ok: false,
      errors: errors.mapped()
    });
  }
  next();
}

module.exports = {
  validarCampos
}
```

```
const { response } = require('express');
const jwt = require('jsonwebtoken');

const validarJWT = ( req, res = response, next ) => {
  const token = req.header('x-token');

  if( !token ) {
    return res.status(401).json({
      ok: false,
      msg: 'error en el token'
    });
  }

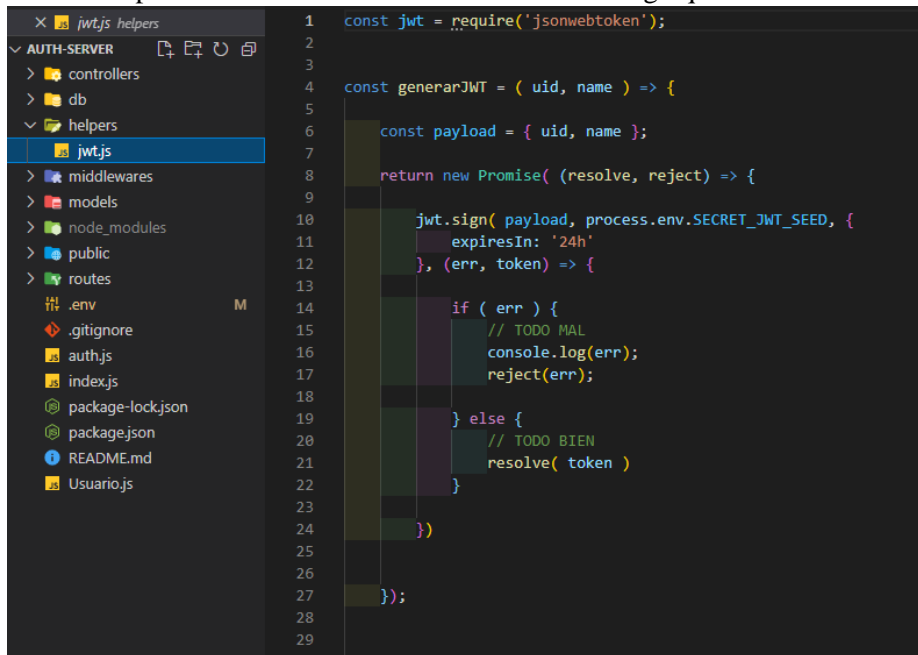
  try {
    const { uid, name } = jwt.verify( token, process.env.SECRET_JWT_SEED );
    req.uid = uid;
    req.name = name;
  } catch (error) {
    return res.status(401).json({
      ok: false,
      msg: 'Token no válido'
    });
  }

  // TODO OK!
  next();
}

module.exports = {
  validarJWT
}
```


11.4: Helpers

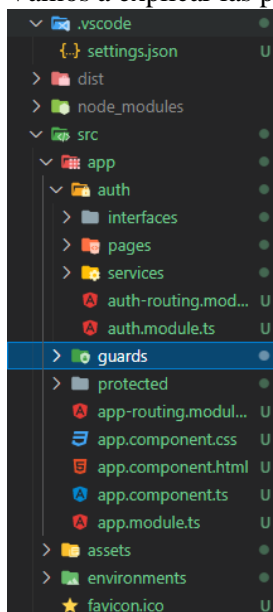
En esta carpeta hemos creado una función con un código que usaremos varias veces a lo largo de nuestra app.



```
1 const jwt = require('jsonwebtoken');
2
3
4 const generarJWT = ( uid, name ) => {
5
6   const payload = { uid, name };
7
8   return new Promise( (resolve, reject) => {
9
10     jwt.sign( payload, process.env.SECRET_JWT_SEED, {
11       expiresIn: '24h'
12     }, (err, token) => {
13
14       if ( err ) {
15         // TODO MAL
16         console.log(err);
17         reject(err);
18       } else {
19         // TODO BIEN
20         resolve( token )
21       }
22     } )
23   } )
24 }
25
26
27
28
29
30
```

12: Frontend

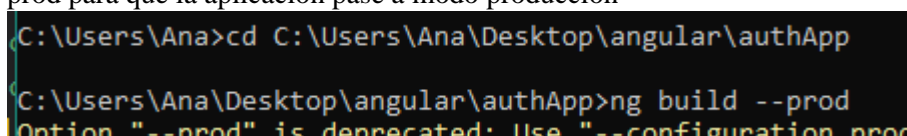
Vamos a explicar las partes más importantes de nuestro frontend.



- **Dist:** esta es la carpeta que se crea después de pasarla por producción con el comando: `ng build --prod`
- **Interfaces:** estipulamos como se nos han de mostrar los campos requeridos con json.
- **Guards:** aquí tenemos un archivo que nos revalidará y actualizará el token cada vez que actualizamos la web en nuestro navegador.
- **Pages:** tenemos los componentes para logearte, registrarte y el main el cual usamos para crear rutas que naveguen por los dos componentes anteriores.
- **Protected:** en este directorio tenemos el componente el cual se nos abrirá una vez que hayamos abierto sesión con nuestro usuario.

13. Unir Frontend y Backend

Abrimos la terminal y sacamos la ruta de nuestra carpeta con el frontend. Lanzamos el comando `ng build --prod` para que la aplicación pase a modo producción



```
C:\Users\Ana>cd C:\Users\Ana\Desktop\angular\authApp
C:\Users\Ana\Desktop\angular\authApp>ng build --prod
Option "--prod" is deprecated. Use "--configuration=production" instead.
```

Vemos el final del proceso en el terminal ya que solo nos deja unos pocos archivos. Abrimos la carpeta dist que es la que se nos crea al finalizar el proceso anterior.

```
✓ Browser application bundle generation complete.
✓ Copying assets complete.
✓ Index html generation complete.

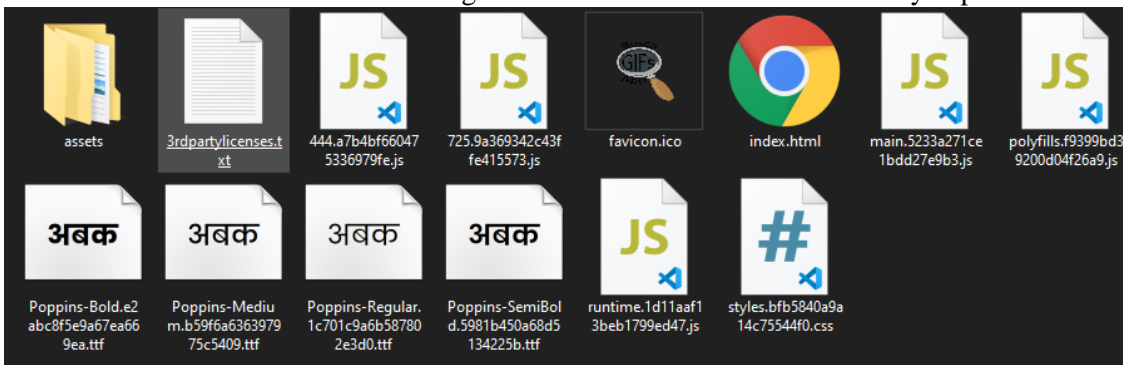
Initial Chunk Files | Names | Size
main.5233a271ce1bdd27e9b3.js | main | 233.80 kB
styles.bfb5840a9a14c75544f0.css | styles | 61.08 kB
polyfills.f9399bd39200d04f26a9.js | polyfills | 35.94 kB
runtime.1d11aaf13beb1799ed47.js | runtime | 2.52 kB

| Initial Total | 333.34 kB

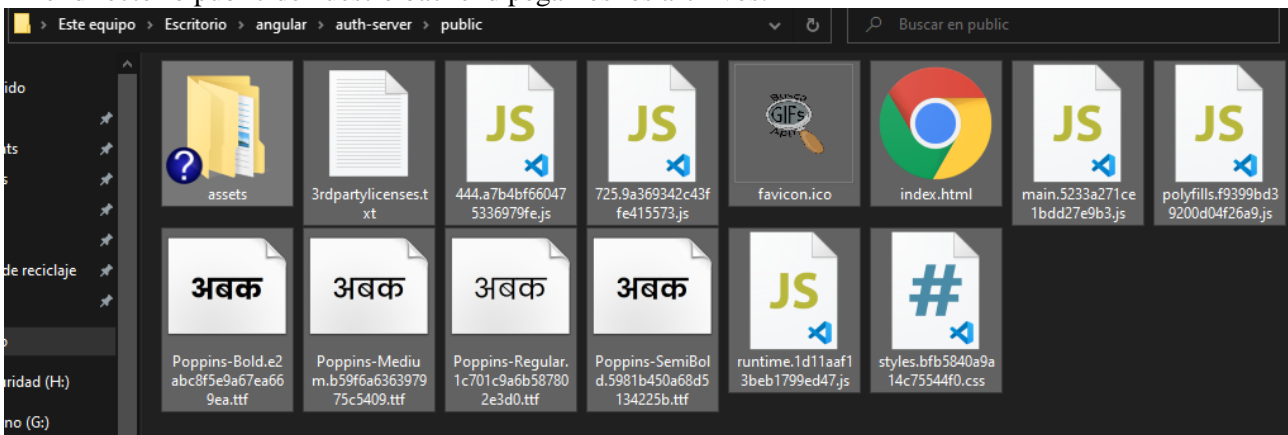
Lazy Chunk Files | Names | Size
444.a7b4bf660475336979fe.js | - | 102.46 kB
725.9a369342c43ffe415573.js | - | 1.29 kB

Build at: 2021-05-20T03:47:45.439Z - Hash: f784827b8c728314e0de - Time: 38951ms
```

En el directorio dist encontramos los siguientes archivos. Los seleccionamos y copiamos.

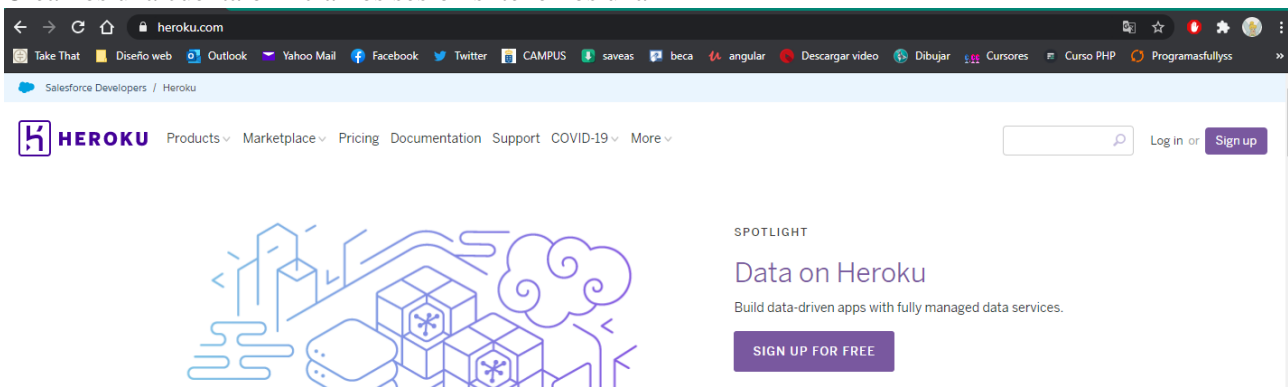


En el directorio public de nuestro backend pegamos los archivos.

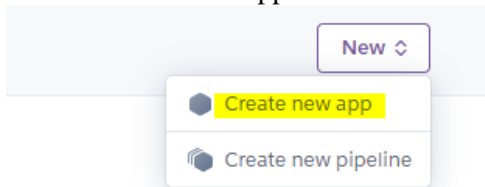


14: Subir app a Heroku

Creamos una cuenta o iniciamos sesión si tenemos una



Creamos una nueva app



Damos nombre al proyecto y elegimos nuestra región. Creamos app.

App name
 ✓
proyecto-gifs2 is available

Choose a region

Europe

[Add to pipeline...](#)

[Create app](#)

Nos descargamos el CLI de Heroku para que nos funcionen los siguientes comandos en la terminal de nuestro pc.

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository

Initialize a git repository in a new or existing directory

```
$ cd my-project/  
$ git init  
$ heroku git:remote -a proyecto-gifs2
```

Deploy your application

Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .  
$ git commit -am "make it better"  
$ git push heroku master
```

Cuando ya se nos ha subido nuestra app en heroku apretamos Open app.

Salesforce Platform

HEROKU

Personal > proyecto-gifs ☆ [Open app](#) [More](#)

GitHub AnaQuesadaGarcia/ProyectoFP

Overview Resources Deploy Metrics Activity Access Settings

Installed add-ons **\$0.00/month** [Configure Add-ons](#)

There are no add-ons for this app
You can add add-ons to this app and they will show here. [Learn more](#)

Dyno formation **\$0.00/month** [Configure Dynos](#)

This app is using free dynos

web	npm start	ON
-----	-----------	----

Latest activity [All Activity](#)

anaquesadagarcia@yahoo.es: Deployed `7840d781`
May 17 at 9:58 PM · v4 · [Compare diff](#)

anaquesadagarcia@yahoo.es: Build succeeded
May 17 at 9:58 PM · [View build log](#)

anaquesadagarcia@yahoo.es: Deployed `50a86926`
May 17 at 9:43 PM · v3

anaquesadagarcia@yahoo.es: Build succeeded
May 17 at 9:43 PM · [View build log](#)

15: Cómo se usa la aplicación

Iniciamos sesión con nuestro usuario y contraseña

proyecto-gifs.herokuapp.com/auty/login

Login

Email
pepe@mail.com

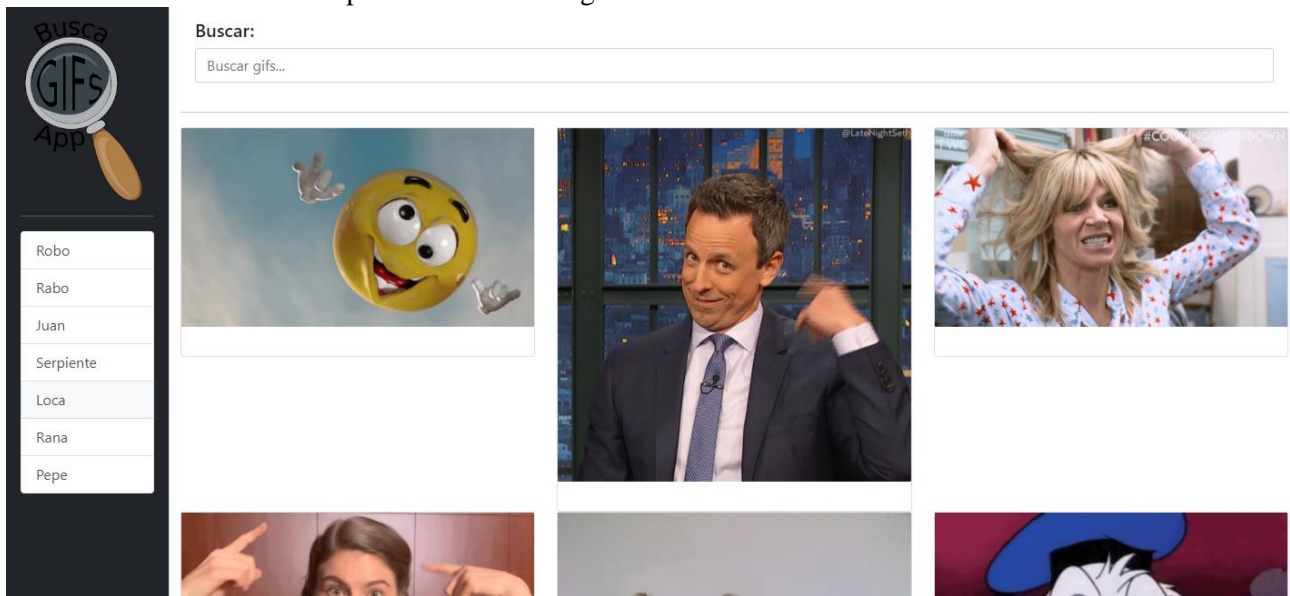
Password
.....

LOGIN

¿No tienes cuenta?
[CREAR UNA AQUÍ](#)

Si no tenemos usuario vamos a darnos de alta aquí

Lo cual nos llevará hasta la aplicación de buscar gifs



Introducimos nuestros datos para darnos de alta

Nuevo usuario

Nombre

Pepe Mejías

Email

pepe@mail.com

Password

.....

REGISTRAR

¿Ya tienes una cuenta?

[INGRESA AQUÍ AQUÍ](#)