



Universidad Tecnológica Nacional  
Facultad Regional Buenos Aires  
Programación Web Avanzado - 02c2022

## Proyecto Final

**Instructores:** Jhoan Carrero  
**Fecha:** 10/2022

# Introduccion

Debido a las nuevas tendencias en el mundo web, se han creado innumerables aplicaciones, las cuales se han destacado por resolver bien sea un problema o una necesidad o bien ofrecer una idea creativa o algun recurso.

Es por este motivo que debemos estar abiertos a nuevas tendencias y poderlas llevar a cabo mediante el uso de la programación, ya que acá no se encuentran limites o barreras naturales en las cosas que se puedan construir.

Asi que llevemos a cabo nuestra primera aplicacion de desarrollo FullStack, una aplicacion donde podremos utilizarla en cualquier momento o usarla de base para nuevos proyectos o porque no? emprendimientos.

El proyecto consiste de una aplicación de votación con emojis; este uso de emojis requiere que tengamos el conocimiento sobre la interpretación de texto y caracteres que realizan nuestras computadoras, en primer lugar, los caracteres de nuestro alfabeto, que comprenden las letras [**a-z,A-Z**], así como numeros [**1-9**] y caracteres especiales [**,.;;?¡...**], son codificados con una tabla llamada **tabla ASCII**.

Pero, pasado el tiempo se tuvo que adoptar otro sistema de codificación, más universal, y es conocido como **unicode**, este estandar pretende englobar tanto nuestro alfabeto como el de otros lenguajes, asi como los distintos **emojis** que podemos encontrar en muchas aplicaciones que usamos al día día.

# EmojiVote App



(<https://emoji-voter-production.up.railway.app>)

La idea es la siguiente, ud se encuentra en un equipo de desarrollo de alto nivel en el que se le ha dado unas tareas referidas a una aplicacion de votacion para emojis, que cumplen con estas consignas:

## 1. **Backend como servicio:**

Un backend como servicio sera el encargado de procesar las peticiones HTTP, bien sea para obtener, registrar, actualizar o borrar datos. Este backend sera encargado de procesar la informacion que se declara a continuacion usando conexion hacia una base de datos no relacional (MongoDB).

Debera estar configurado con las siguientes variables de entorno la conexion con la base de datos:

- a) **PORT** (Sera el puerto donde alojara el servicio)
- b) **DB\_URI** (Sera la uri para la conexion con la base de datos)  
protocolo://usuario:contraseña@dominio:puerto/base\_de\_datos

Para la manipulacion de datos sera necesario que se cargue en la base de datos la informacion extraida del siguiente JSON.

## 1. Backend como servicio:

- a) **GET /api/v1/emojis** (Obtener la lista de emojis, ordenados descendientemente por cantidad votos):

**Request:** La petición podrá controlar la cantidad de emojis que desea obtener, así como, la posición inicial de la búsqueda.

Estos parámetros serán:

- 1) **start**: será un número, el cual determinará la posición del vector por el cual iniciará la búsqueda de los emojis.
  - Si **start** es un número muy grande, es decir, sobrepasa la cantidad de emojis que existe, devolverá un vector vacío, por otro lado si es más pequeño, es decir, un número negativo, tendrá **start** un valor de **0**, que es el inicio del vector.
  - Si no es especificado este parámetro, por defecto, **start** será el inicio del array, que es **0**.
- 2) **limit**: será un número, comprendido en un rango del **1 al 10**, el cual determinará la cantidad de emojis de la respuesta.
  - Si **limit** es un número muy grande, es decir, es un número mayor a **10**, tendrá un valor de **10**, por el contrario si es más pequeño, es decir, menor a **1**, tendrá un valor de **1**.
  - Si no es especificado este parámetro, por defecto el **limit** será el máximo, que es **10**.

Estos parámetros serán enviados en la **url** como **query**.

**Ejemplo:** `/api/v1/emojis?start=0&limit=2`.

**Response:** La respuesta debera tener la siguiente estructura:

1) Caso que la informacion sea correctamente procesada:

- Codigo de estado: **200**
- Cuerpo de la respuesta:

```
1 {
2     "result": [
3         {
4             "_id": "05958bc5eeca",
5             "emoji": "\u1f600",
6             "name": "smile",
7             "votes": 2
8         },
9         {
10            "_id": "05958bc5eecb",
11            "emoji": "\u1f603",
12            "name": "other smile",
13            "votes": 0
14        }
15    ],
16    "total": 2
17 }
```

2) Caso que uno o varios errores hayan ocurrido:

- Codigo de estado: **500**
- Cuerpo de la respuesta:

```
1 {
2     "errors": [
3         {
4             "message": "Database
5             failed"
6         }
7     ],
8 }
```

- b) **GET** `/api/v1/emojis/{:id}` (Obtener la información de un emoji por `_id`):

**Request:** La petición podrá controlar que emoji desea obtener. Este parámetro será el **id** y será usado en la **url** como **params**.

**Ejemplo:** `/api/v1/emojis/05958bc5eeca`.

**Response:** La respuesta deberá tener la siguiente estructura:

- 1) Caso que el **id** exista y la conexión fue exitosa:

- Código de estado: **200**
- Cuerpo de la respuesta:

```
1 {  
2     "result": {  
3         "_id": "05958bc5eeca",  
4         "emoji": "\u1f600",  
5         "name": "grinning face",  
6         "votes": 2  
7     },  
8 }
```

- 2) Caso que el **id** no exista o la conexión no fue exitosa

- Código de estado: **404** o **500**
  - a) **404** - Si no existe el emoji
  - b) **500** - Si ocurrió un error
- Cuerpo de la respuesta:

```
1 {  
2     "errors": [  
3         {  
4             "message": "Not found"  
5         }  
6     ],  
7 }
```

c) **POST** `/api/v1/votes` (Publicar un voto a un emoji):

**Request:** La petición podrá controlar que emoji desea votar. Este parámetro será el **id** y será enviado en el cuerpo de la petición, como **body**.

```
1 { "id" : " 05958bc5eeeca" }
```

**Ejemplo:** `/api/v1/votes`, **body**={**"id"**: "05958bc5eeeca"}

**Response:** La respuesta deberá tener la siguiente estructura:

1) Caso que el **id** exista y la conexión fue exitosa:

- Código de estado: **200**
- Cuerpo de la respuesta:

```
1 {  
2     "result": {  
3         "_id": "05958bc5eeeca",  
4         "emoji": "\u1f600",  
5         "name": "grinning face",  
6         "votes": 2  
7     },  
8 }
```

2) Caso que el **id** no exista o la conexión no fue exitosa

- Código de estado: **404** o **500**
  - a) **404** - Si no existe el emoji
  - b) **500** - Si ocurrió un error
- Cuerpo de la respuesta:

```
1 { "errors": [ { "message": "Not found" } ] }
```

## 2. Frontend como servicio:

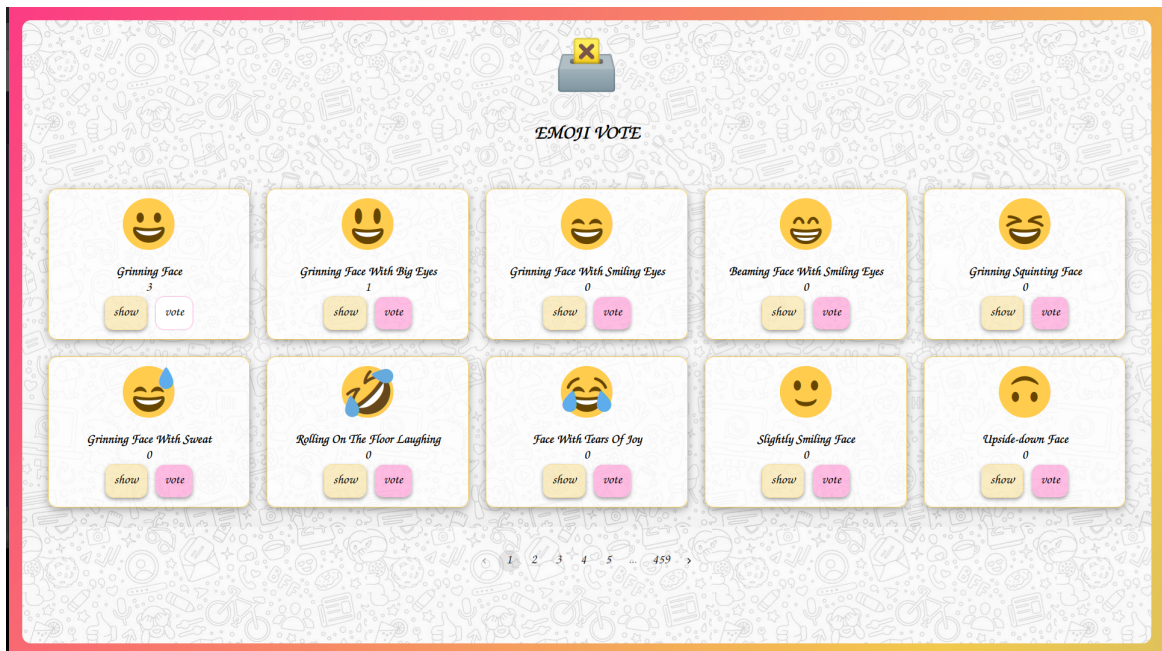
Un frontend como servicio sera el encargado de procesar las interacciones con el cliente, para obtener o publicar los datos.

Debera contar con las siguientes vistas:

a) Vista Principal [ Ruta = '/' ]

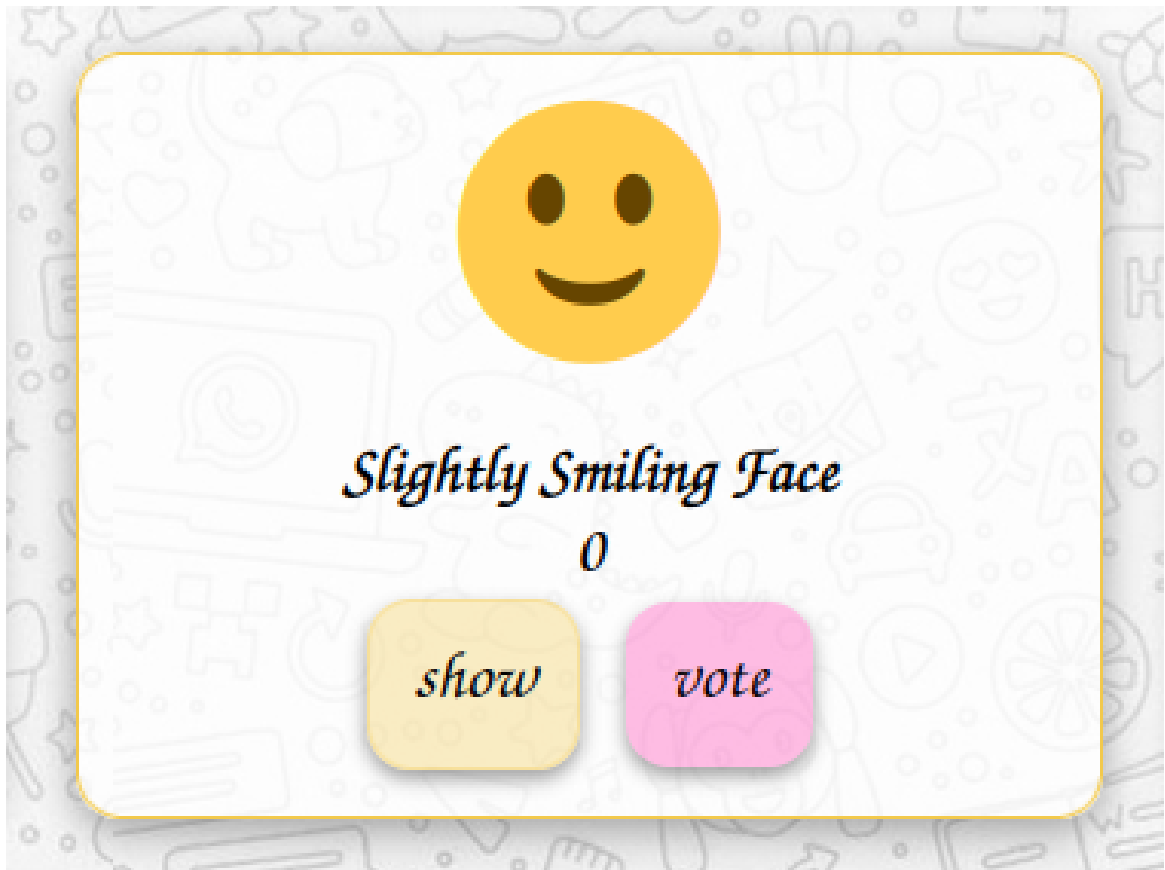
Esta vista debe realizar una redirección hacia la ruta **/emojis**

b) Vista Emojis [ Ruta = '/emojis' ]



Esta vista sera la encargada de mostrar todos los emojis que se encuentran registrados, mostrando los primeros **10** emojis mas votados, en una presentacion de **'cards'**





Estas '**cards**' deberán contener como principal resaltador, el **emoji** correspondiente, seguido del **nombre** y la cantidad de **votos**. Por ultimo dos botones, **show** y **vote**, que deberán cumplir lo siguiente:

- 1) **vote:** deberá ser un boton clickeable para poder realizar la votacion al emoji.
- 2) **show:** deberá ser un enlace para poder visualizar el emoji correspondiente, donde la ruta o el hipervinculo deberá seguir la siguiente estructura: **/emojis/:id**, donde el **:id** es el valor de la propiedad **\_id** del emoji.



Debera contar con una **paginación**, la cual le permitirá al usuario navegar entre distintas listas de emojis; poder encontrar el que desea votar.

c) Vista Show [ Ruta = /emojis/:id"]



Esta vista sera dinamica dependiendo del parametro de la url, que se declara como **:id**, que corresponde al **\_id** del emoji. Debera mostrar, la **'card'** con el **emoji**, el **nombre** y la cantidad de **votos**, asi como un boton con el enlace para regresarse a la pagina principal que tendra por ruta **'/emojis'**.

# Entrega

La fecha de entrega sera los primeros dias de diciembre, es importante que todos los cambios o dudas que tengan las aclaren antes de esta fecha.

El metodo de entrega del proyecto se realizara atraves de Github en sus ramas correspondientes, respetando la estructura de carpetas.

```
nombre_apellido
├── lessons
├── practices
├── project
│   ├── client
│   │   ├── package.json
│   │   └── server
│   │       ├── package.json
│   │       └── package.json
```

Todo el codigo tanto del Backend como del Frontend se encontrara en la carpeta **project**. Esta carpeta debera dividir el codigo en dos subcarpetas, por un lado el frontend en la carpeta **client**, con el codigo de react, por otro lado la carpeta **server** contendra todos los archivos con express.

Estas carpetas deberan contener los archivos de **package.json** y los scripts necesarios para levantar el proyecto.

En la carpeta principal, es decir, **project**, en la seccion de **scripts** del **package.json**, deberan existir los siguientes:

1. **prepare**: Sera el encargado de instalar las dependencias tanto del cliente como del servidor
2. **build**: Sera el encargado de compilar y construir la aplicacion del cliente de react
3. **start**: Sera el encargado de iniciar la aplicacion del servidor de express

# Evaluacion

El metodo de evaluacion, es atraves de testing automatico, esto quiere decir, que el codigo se evaluara atraves de pruebas unitarias, usando codigo javascript para garantizar que se cumplan los requisitos.

Se han dejado unas pruebas de codigo, las cuales seran ejecutadas, directamente con el uso de **npm test**.

Seran **25** pruebas en total, las cuales contemplan las siguientes características:

1. **5** para estructura de carpetas y archivos
2. **10** para evaluacion de codigo del backend
3. **10** para evaluacion de codigo del frontend

La aprobacion de la cursada sera considerada con haber aprobado almenos **15** tests, estos, pueden ser tanto de estructura de carpetas, como codigo en backend o como codigo en frontend.

Luego de haber completado las pruebas, debera estar subido al repositorio, y posteriormente haber solicitado una **PULL REQUEST**, la cual hara la aprobacion automatica utilizando la aprobacion de almenos **15** tests.