Universidad Tecnologica 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) **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 (**https://raw.gi**
   api/main/src/emojis.json).

1. **Backend como servicio:**

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

   Request: La peticion podra controlar la cantidad de emojis que desea obtener, asi como, la posicion inicial de la busqueda.

   Estos parametros serán:

   1) **start**: sera un número, el cual determinará la posición del vector por el cual iniciara la búsqueda de los emojis.
      - Si **start** es un número muy grande, es decir, sobrepasa la cantidad de emojis que existe, devolvera un vector vacio, por otro lado si es mas pequeño, es decir, un numero negativo, tendrá **start** un valor de **0**, que es el inicio del vector.
      - Si no es especificado este parametro, por defecto, **start** sera el inicio del array, que es **0**.
   2) **limit**: sera 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**, tendra un valor de **10**, por el contrario si es mas pequeño, es decir, menor a **1**, tendrá un valor de **1**.
      - Si no es especificado este parametro, por defecto el **limit** sera el máximo, que es **10**.

   Estos parametros seran 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:

```
{
    "result": [
        {
            "_id": "05958bc5eeca",
            "emoji": "\u1f600",
            "name": "smile",
            "votes": 2
        },
        {
            "_id": "05958bc5eecb",
            "emoji": "\u1f603",
            "name": "other smile",
            "votes": 0
        }
    ],
    "total": 2
}
```

2) Caso que uno o varios errores hayan ocurrido:

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

```
{
    "errors": [
        {
            "message": "Database
              failed"
        }
    ],
}
```

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

Request: La peticion podra controlar que emoji desea obtener. Este parametro sera el **id** y sera usado en la **url** como **params**.
Ejemplo: /api/v1/emojis/**05958bc5eeca**.

Response: La respuesta debera tener la siguiente estructura:
1) Caso que el **id** exista y la conexion fue exitosa:
- Codigo de estado: **200**
- Cuerpo de la respuesta:

```
{
    "result":{
        "_id": "05958bc5eeca",
        "emoji": "\u1f600",
        "name": "grinning face",
        "votes": 2
    },
}
```

2) Caso que el **id** no exista o la conexion no fue exitosa
- Codigo de estado: **404** o **500**

a) **404** - Si no existe el emoji
b) **500** - Si ocurrio un error

- Cuerpo de la respuesta:

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

c) **POST /api/v1/votes/{:id}** (Publicar un voto a un emoji por **_id**):

Request: La peticion podra controlar que emoji desea votar. Este parametro sera el **id** y sera usado en la **url** como **params**.
Ejemplo: /api/v1/emojis/**05958bc5eeca**.

Response: La respuesta debera tener la siguiente estructura:
1) Caso que el **id** exista y la conexion fue exitosa:
- Codigo de estado: **200**
- Cuerpo de la respuesta:

```
{
    "result":{
        "_id": "05958bc5eeca",
        "emoji": "\u1f600",
        "name": "grinning face",
        "votes": 2
    },
}
```

2) Caso que el **id** no exista o la conexion no fue exitosa
- Codigo de estado: **404** o **500**

a) **404** - Si no existe el emoji
b) **500** - Si ocurrio un error

- Cuerpo de la respuesta:

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

2. **Frontend como servicio:**

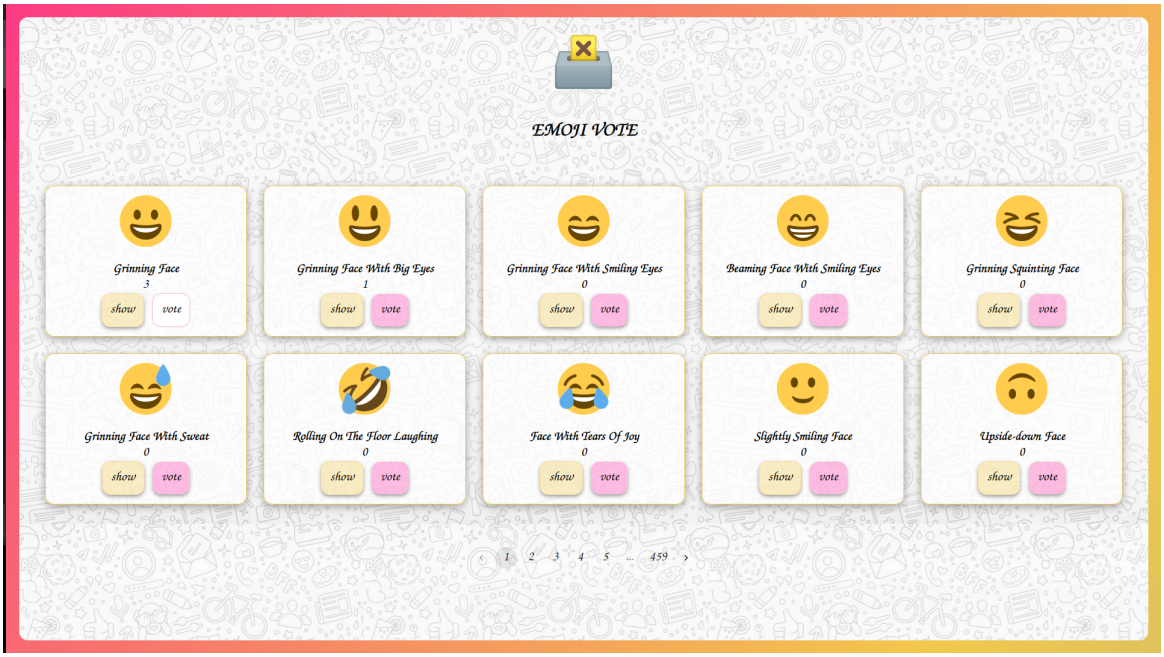
Un frontend como servicio sera el encargado de procesar las interacciones con el cliente, para obtener o publicar los datos.

Debera contar con las siguientes vistas:

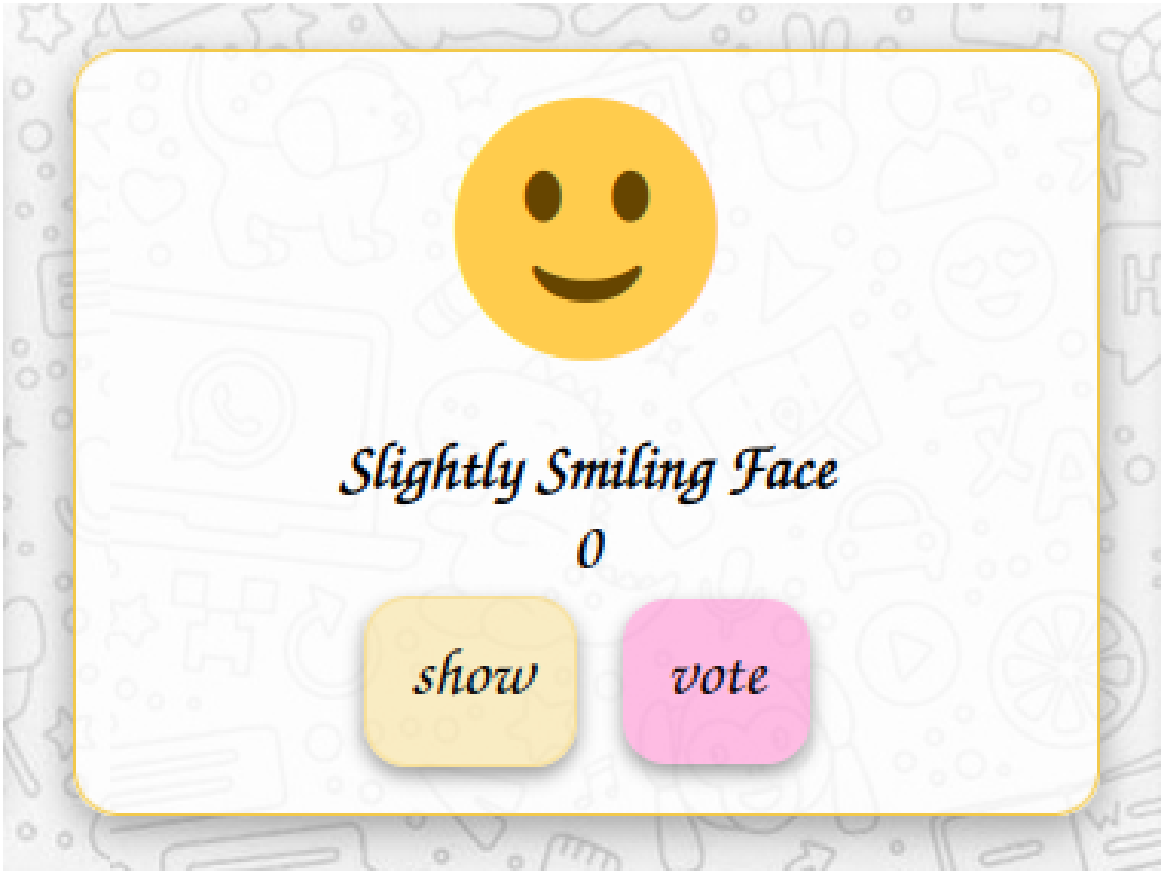
a) Vista Principal [Ruta = '/']

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

1) **vote:** debera ser un boton clickeable para poder realizar la votacion al emoji.

2) **show:** debera ser un enlace para poder visualizar el emoji correspondiente, donde la ruta o el hipervinculo debera 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
```

Todo el codigo tanto del Backend como del Frontend se encontrara en la carpeta **project**. Luego de subido al repositorio se debera pedir una Pull Request con el siguiente titulo:

Entrega Final - Nombre Apellido