MC - Mi Comunidad

Nos encargaron el diseño y desarrollo de MC - “Mi comunidad”, una red social que está destinada a conectar amigos. En esta comunidad cada persona podrá realizar posteos así como likear o comentar posteos de otros amigos.

Parte 1 – Gestión de usuarios

En esta parte nos encargaremos de diseñar y desarrollar el ABML de los usuarios.
Cada usuario deberá tener mínimamente: nombre, apellido, email, fecha de nacimiento y biografía (texto de máximo 500 caracteres). Además, de cada uno sabemos la provincia y localidad donde reside; así como también sus gustos musicales (“pop”, “rock”, etc.).
Es importante considerar que no pueden existir dos o más usuarios con el mismo email ya que este dato debe ser único por cada uno.

Endpoints mínimos a desarrollar

- ***POST /usuarios***
- ***GET /usuarios/:id***
- ***PUT /usuarios/:id***

También considerar los endpoints:

- ***POST /usuarios/:id/gustos-musicales***
- ***GET /usuarios/:id/gustos-musicales***

Parte 2 - Estadísticas sobre usuarios

En esta parte nos encargaremos, de forma grupal, de calcular algunas estadísticas sobre los usuarios:

- Cuántos usuarios comparten el gusto musical por “X” (pop, rock, etc.).
- Cuántos usuarios residen en determinada provincia
- Cuántos usuarios residen en determinada localidad
- Cuántos usuarios son mayores de X años

Parte 3 - Mis amigos

En esta parte nos encargaremos de diseñar y desarrollar la gestión de amigos de un usuario. Cada usuario puede “hacerse amigo” de muchos otros, teniendo especial cuidado de que no pueda “hacerse amigo” de “él mismo”.

Cuando un usuario quiere hacerse amigo de otro usuario, primero se deberá crear una “solicitud de amistad”. La solicitud de amistad debe ser aceptada por el usuario receptor para que finalmente se concrete el lazo de amistad. Si la solicitud fue rechazada, entonces el lazo de amistad no debe hacerse.

Endpoints mínimos a desarrollar

- ***POST /solicitudes-de-amistad*** permite que un usuario cree una solicitud de amistad hacia otro usuario.
- ***PUT /solicitudes-de-amistad/:id*** se utiliza para poder aceptar o rechazar una solicitud de amistad
- ***GET /usuarios/:id/amigos*** devuelve los amigos que tiene un usuario
- ***GET /usuarios/:id/amigos-pendientes*** devuelve las solicitudes de amistad enviadas que estén pendientes de aceptación
- ***GET /usuarios/:id/solicitudes-de-amistad*** devuelve las solicitudes de amistades recibidas que estén pendientes de aceptación
- ***GET /usuarios/:id/falsos-amigos*** devuelve las solicitudes de amistad enviadas que hayan sido rechazadas por los receptores.

Parte 4 - Estadísticas sobre amigos

En esta parte nos encargaremos, nuevamente de forma grupal, de calcular algunas estadísticas referidas a los enlaces de amistad:

- Qué usuarios son los más “spammers”. Se considera que un usuario es “spammer” cuando tiene más de X solicitudes de amistad enviadas pendientes de aceptación.
- Qué usuarios son los más “callados”. Se considera que un usuario es “callado” cuando tiene menos de X amigos en esta red.
- Qué usuarios son los más “rechazados”, considerando que si un usuario tiene al menos 1 solicitud de amistad rechazada se lo considera “rechazado”.

Parte 5 - Posts

En esta parte nos encargaremos de diseñar y desarrollar los posts que pueden realizar los usuarios. Cada post debe contar mínimamente con un título y un cuerpo. El cuerpo del post no puede contener más de 3000 caracteres.

Los usuarios deben poder realizar como máximo 5 posts al día.

Endpoints mínimos a desarrollar

- ***POST /posts*** permite crear un post a un usuario.
- ***GET /posts?usuarios=id*** permite recuperar todos los posts de un usuario
- ***GET /posts/:id*** permite recuperar únicamente un único post

Parte 6 - Estadísticas sobre posts

En esta parte nos encargaremos, nuevamente de forma grupal, de calcular algunas estadísticas referidas a los posts:

- Quiénes son los usuarios más activos del día: los usuarios más activos de un día en particular son aquellos que han superado el 60% de la cantidad total de posts permitidos diarios.

- Quiénes son los usuarios que postean poco. Un usuario postea “poco” cuando ha realizado como máximo 1 post por semana en las últimas 4 semanas.
- Quiénes son los usuarios que “escriben mucho”. Un usuario escribe mucho cuando al menos en el 70% de sus posts alcanzó el 90% de caracteres máximos permitidos.

Parte 7 - Likes y comentarios

En esta parte nos encargaremos de diseñar y desarrollar los “likes y comentarios”.

Los usuarios deberán poder dejar likes en los posts de otros usuarios, teniendo en cuenta que como máximo se puede dejar uno solo (un solo like por usuario - post). Además, también deberán poder dejar uno o varios comentarios sobre los posts. Cada comentario debe admitir como máximo 1000 caracteres.

Endpoints mínimos a desarrollar

- ***POST /posts/:id/like*** permite que un usuario le de like a un post
- ***GET /posts/:id/likes*** permite obtener todos los likes de un post, detallando quién dió cada uno
- ***POST /posts/:id/comentarios*** permite que un usuario deje un comentario sobre un post
- ***GET /posts/:id/comentarios*** permite obtener todos los comentarios asociados a un post

Parte 8 - Estadísticas sobre posts

En esta parte nos encargaremos de realizar algunas estadísticas sobre los likes y comentarios de los posts:

- Cuál fue el post más “likeado” de un día en particular
- Cuál fue el post más “comentado” de un día en particular
- Cuáles son los posts que contienen más de X comentarios
- Cuáles son los posts que tienen más de X cantidad de likes

Distribución planteada

Grupo 1

Guillermo Mooney

Nahuel Herrera

Martin Ezequiel Diaz Villarrubia

Grupo 2

Leonardo Fernandez

Luca Viliani

Mikel Taberna

Grupo 3

Leandro Laiker

Joaquin Tarczydlo

Nancy Figueredo

Grupo 4

Joaquin Allue

Martina Benegas Nebbia

Matias Martyn

Grupo 5

Santiago Calvelo

Lucca Cavalcanti