Nombre del Proyecto
Gossip

Descripción del Proyecto
Se trata de una aplicación donde los usuarios pueden crear mensajes con rumores acerca de su entorno (algo como Twitter, pero sin el pajarito). Los usuarios pueden evaluar los gossips de los demás mediante botones que indican que un rumor es cierto o no. Los gossips tienen karma, el cual se representa con un número (que puede ser positivo, negativo o cero), indicando qué tan cierto o falso es un rumor. En esencia, se tiene una pequeña "~~*red social*~~" donde se pueden compartir mensajes y promoverlos aprobándolos o rechazándolos.

La API para consumir los recursos les será entregada el día **miércoles 30 de noviembre**, así que pueden ir trabajando en las interfaces de la aplicación sin añadirle los métodos que emplean XMLHttpRequest.

Limitaciones

1. No se permite la utilización de frameworks (con excepción de los frameworks CSS), esto incluye Angular, React (aunque este no es un framework realmente), Knockout, Ember, Backbone, entre otros; tampoco se permite la utilización de librerías de terceros, como jQuery (a menos que sea **sólo** como dependencia de un framework CSS), momentJS, bootstrap-notify, entre otras. El asunto es que no quiero ver una sola "$" en todo el código, a menos que ustedes mismos hayan definido un objeto con ese nombre.
2. No se permite copiarse el proyecto. En serio, ya tienen que parar. El problema no es que **crean** que no me doy cuenta, es que **están seguros** de que es así.

* Violar alguna de las limitaciones anteriores implica la pérdida de 19,99999… puntos en la evaluación del proyecto.

Requerimientos
Todos los usuarios de la aplicación deben tener acceso a interfaces que les permitan realizar, como mínimo, las siguientes acciones:

1. Escoger un nombre para publicar gossips.
2. Publicar un nuevo gossip.
3. Ver los gossips de todos los usuarios.
4. Eliminar un gossip publicado por sí mismo.
5. Evaluar un gossip como verdadero (aumentar su karma).
6. Evaluar un gossip como falso (disminuir su karma).

Requerimientos Adicionales (no son opcionales, son obligatorios)
La aplicación debe tener un módulo de administrador, donde se empleen endpoints especiales que permiten manipular el estado de los gossips; del mismo modo, el módulo de administrador tendrá acceso a endpoints que permiten visualizar los logs de la aplicación, donde se podrá observar todas las acciones que los usuarios realizan con Gossip.

Los usuarios que acceden al módulo de administrador deben tener acceso a interfaces que les permitan realizar, como mínimo, las siguientes acciones:

1. Escoger un nombre para ingresar en el módulo de administrador.
2. Ver los gossips de todos los usuarios (independientemente de su estado).
3. Eliminar un gossip.
4. Recuperar (algo así como *deseliminar*) un gossip.
5. Ver los logs de la aplicación.

Requerimientos Opcionales
Si aún teniendo en cuenta los requerimientos anteriores crees que tienes lo necesario para **ganarte los stickers brutales**, sigue leyendo.

1. Búsqueda de gossips por nombre de usuario.
2. Búsqueda de gossips por palabra (o frase, como prefieras) clave.
3. Búsqueda de gossips por karma (mayor a 100, karma negativo, karma neutro, por ejemplo).
4. Actualización automática de la lista de gossips de todos los usuarios empleando Workers.

Estos requerimientos opcionales pueden ser cumplidos sin la necesidad de un endpoint en el servidor, razón por la cual no se crearán dichos elementos, que permitirían realizar estas acciones de manera más sencilla para el desarrollador front-end.

Observaciones
Aunque los requerimientos opcionales no suponen un reto mayúsculo, traen beneficios para aquellos que decidan completarlos. A continuación, la lista de premios por completar los requerimientos opcionales:

1. Cuando se completan todos los requerimientos obligatorios y dos requerimientos opcionales, se obtienen 2 puntos en el examen teórico.
2. Cuando se completan todos los requerimientos obligatorios y todos los requerimientos opcionales, se obtienen 3 puntos en el examen teórico y un sticker brutal para cada integrante del equipo.

Recomendaciones
1. La aplicación puede ser realizada en equipos conformados por un máximo de 2 personas.
2. Estudien mucho antes de comenzar a desarrollar la aplicación.
3. Nunca se han ganado los stickers brutales.
4. Pregunten, si tienen dudas.
5. Si ven que cometí algún error explicando en las páginas siguientes, pueden hacérmelo saber, si desean.

Endpoints de Gossip

Nombre del Endpoint	URL	Método	Datos de Solicitud	Datos de Respuesta
Nuevo gossip	/gossip/create	POST	- id_usuario - de_gossip	- status - message
Ver todos los gossips	/gossip/all	GET	N/A	- status - message - gossips[] (Gossip)
Eliminar un gossip (propio)	/gossip/delete	GET	- id_gossip - id_usuario	- status - message
Evaluar un gossip como verdadero	/gossip/up	POST	- id_gossip - id_usuario	- status - message
Evaluar un gossip como falso	/gossip/down	POST	- id_gossip - id_usuario	- status - message
Ver todos los gossips (administrador)	/admin/gossip/all	GET	N/A	- status - message - gossips[] (Gossip)
Eliminar un gossip (administrador)	/admin/gossip/delete	GET	- id_gossip - id_usuario	- status - message
Recuperar un gossip	/admin/gossip/recover	POST	- id_gossip - id_usuario	- status - message
Ver los logs de la aplicación	/admin/log/all	GET	N/A	- status - message - logs[] (Log)

* Todas las solicitudes deben hacerse enviando los datos en formato JSON. El servidor responderá empleando el formato JSON, así que preparen su front-end para manejar dicha estructura.

Morfología de Objetos de los Endpoints

A continuación se presenta la forma de los objetos que se mencionan en la tabla de endpoints de la parte superior. Esto se construye con la finalidad de que conozcan los datos que el back-end les estará enviando y con cuáles nombres estará haciéndolo.

Nombre del objeto	Propiedades
Gossip	- id_gossip (Integer) - de_gossip (String) - ka_gossip (Integer) - da_gossip (Date) - id_gossip_status (Integer) - de_gossip_status (String)
Log	- id_gossip_log (Integer) - de_gossip_log (String) - da_gossip_log (Date)

Diccionario de Datos de la Aplicación

Uno de los apartados más importantes para conocer cómo funciona una aplicación y prepararse adecuadamente para la integración entre el front-end y el back-end, es saber cuál es la utilidad de los datos que se están intercambiando. A continuación se presenta el diccionario de datos, donde se describe cada propiedad de cada tabla de la base de datos (separadas por tabla).

Tabla: gossip_status

Nombre de la Propiedad	Utilidad
id_gossip_status	Es el ID del estado de un gossip. Básicamente indica numéricamente el estado en el cual se encuentra un gossip.
de_gossip_status	Es la representación en forma de palabra del estado en el cual se encuentra un gossip.

Tabla: gossip

Nombre de la Propiedad	Utilidad
id_gossip	Es el ID de un gossip. Permite identificar universalmente a un gossip dentro de la aplicación.
id_user	Es el ID del usuario que creó al gossip. Se trata de una cadena de caracteres con una longitud máxima de 25.
id_gossip_status	– VER TABLA gossip_status –
de_gossip	Es el contenido del gossip como tal, aquí se encuentra lo que el usuario decidió escribir.
ka_gossip	Es el karma del gossip. Se trata de un entero que puede ser positivo, negativo o cero.
da_gossip	Se trata de una fecha en formato AAAA-MM-DD, que refleja el día de creación del gossip.

Tabla: gossip_log

Nombre de la Propiedad	Utilidad
id_gossip_log	Es el ID de un log, permite identificar universalmente a un registro dentro de la aplicación.
id_gossip	Es el ID del gossip involucrado en el log.
da_gossip_log	Es la fecha (en formato AAAA-MM-DD) en la cual se creó el registro en la base de datos.
de_gossip_log	Es el contenido del registro como tal, aquí se encuentra la frase que indica la acción que se grabó en base de datos, como *"EL USUARIO wilquitriyeilo10 HA ELIMINADO EL GOSSIP"*.

Diagrama Entidad-Relación de la Aplicación