Informe Proyecto Semestral IIC2513

Camilo Berríos
Christian Eilers
Maximiliano Schudeck

Introducción

La aplicación desarrollada durante el transcurso de este proyecto, corresponde una app que combina una red social con una página de compra-venta de items varios. La aplicación posee grupos, en los cuales se encuentran las publicaciones a las que corresponden los items a vender e intercambiar. En las publicaciones, los usuarios pueden comentar, así como hacerle ofertas al vendedor, de manera que éste decida cuáles ofertas son aquellas que más le interesan. Por último, los buenos vendedores serán reconocidos por los usuarios, los cuales serán capaces de darles un ránking de acuerdo al servicio entregado. También, los usuarios tanto registrados como no registrados, podrán buscar a otros usuarios, grupos, y publicaciones, para así poder explorar lo que XChange tiene para ofrecerles. La aplicación permite la exploración de ésta a usuarios no registrados, los cuales podrán revisar las entidades más relevantes de la app.

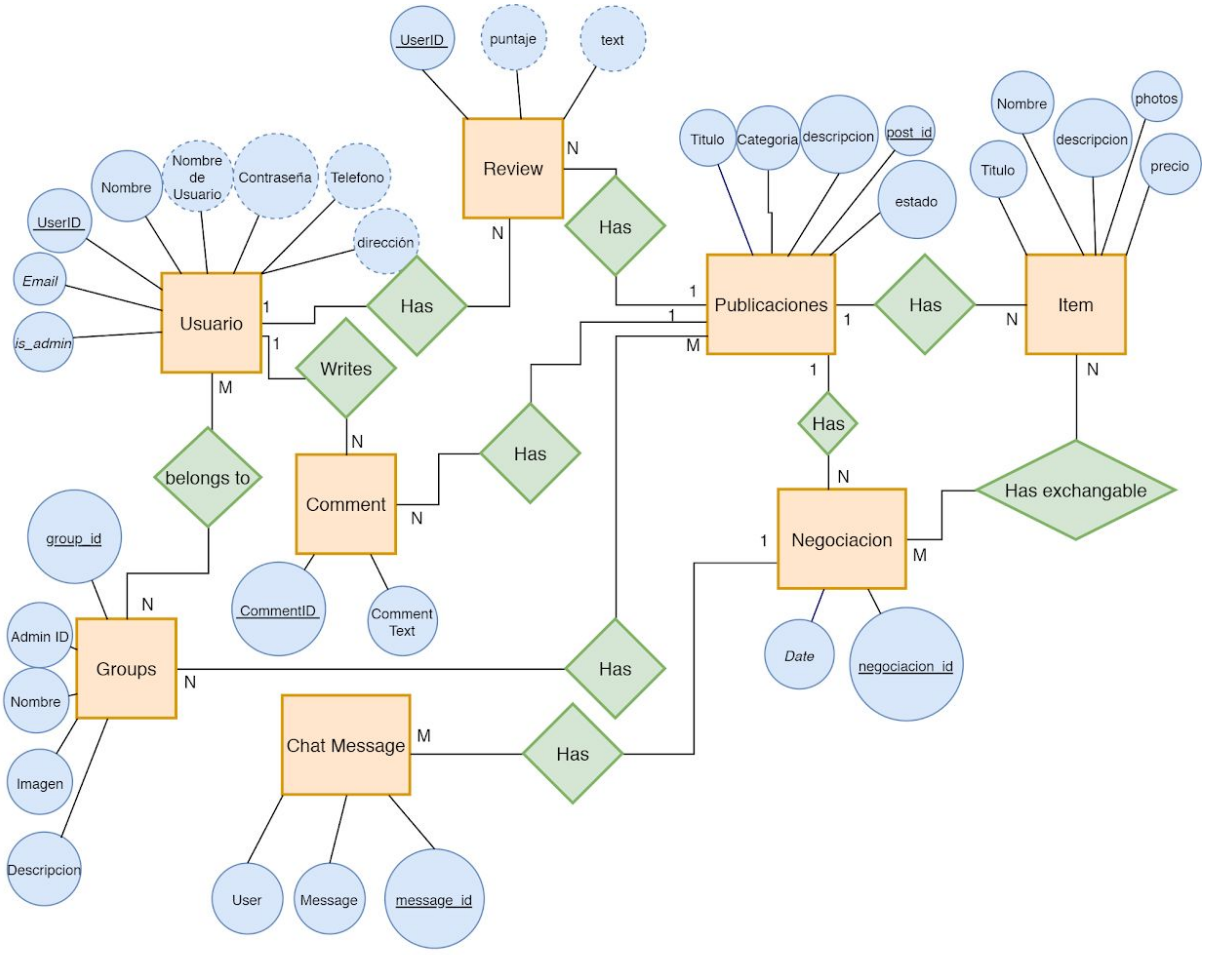
Arquitectura:

El proyecto sigue una estructura MVC, regida por NodeJS. Esto separa muy bien el proyecto entre backend, frontend y datos.
El sector de frontend fue desarrollado con React + Redux donde fue necesario, aunque aún se encuentran algunas “legacy-views” que están completamente en ejs/css.
Por otro lado, el backend trabaja sobre NodeJS (como fue mencionado), junto con otras librerías de routing como Koa.
El modelo de datos está basado en postgres con manejo de Object Relational Mapping utilizando Sequelize. El modelo utilizado por este es especificado en la siguiente sección.

Modelo de Datos:

Al momento de elegir como modelar los datos y entidades dentro de la App en casos de dependencia, decidimos usar “cascade”, para así no tener que preocuparnos de que hayan conflictos de enlaces que puedan llevar a la app a funcionar de maneras que no debería. Dado este caso, al momento de eliminar un grupo, por ejemplo, se eliminarían todas las publicaciones de este, así como los comentarios de las publicaciones. Esto fué modelado así, para simular el funcionamiento de una red social de verdad que también tuviera grupos.

El modelo más en detalle se puede ver en el siguiente bosquejo, el cual se siguió desde el principio del desarrollo:



Detalles de la Implementación:

Al momento de registrarse, la aplicación manda un mail al correo entregado por el usuario. Esta funcionalidad permite el envío de emails al correo de cada usuario, y si se quisiera ampliar el funcionamiento y la variedad de mails que se le envían, como puede ser con notificaciones de ofertas hechas por posibles compradores a los productos del usuario, u anuncios de distinta índole, se podría ampliar el funcionamiento de esta herramienta ya implementada en la aplicación, de manera rápida.

La app depende de un servidor Amazon S3 para el alojamiento de archivos como fotos de los usuarios, de los grupos, y de las publicaciones. Esto, para así no depender de un servidor local o active storage. Lo único que guarda la aplicación es el link correspondiente a cada foto subida, el cual es asignado al momento de concretarse que la foto se haya subido. Si se quisiera utilizar esta aplicación, es necesaria la adquisición o implementación de un servidor para almacenamiento de archivos.

La API se implementó usando JSON Web Token, para autenticarse simplemente se envía un PUT request a /sessions/ con un payload de {username, password}, lo cual te autentifica

y te entrega una cookie con la que puedes enviar requests libremente por la API. Esta cuenta con manejo completo sobre las publicaciones, aunque es fácilmente extensible para otras áreas. Se construyó una muy simple “mini-app” que consume la API, esta consta de un login, que corresponde a las credenciales de un usuario registrado. Luego se despliegan todas las publicaciones del sitio, su contenido y la opción para eliminar dicha publicación. Cabe mencionar que la API abarca más de lo que esta “mini-app” permite hacer. Como desplegar comentarios.

Por otro lado, la navbar y sidebar son renders parciales, por lo que trabajar sobre ellos es sencillo al estar modularizados. El navbar posee una barra de búsqueda react que despliega una lista de autocompletado, al clickear un elemento de ésta, se va a la página del elemento, al escribir solamente y presionar enter, se va a una página con los resultados de la búsqueda. La búsqueda es Case-Sensitive.