Integrantes: Marcelo Agustin Origoni Guillaume
Ayudante/Corrector: Ezequiel Garcia

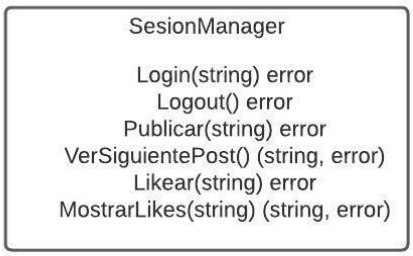
Curso: Algoritmos y programación 2 Buchwald

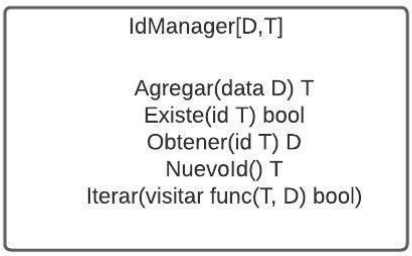
Informe TP2: Algogram

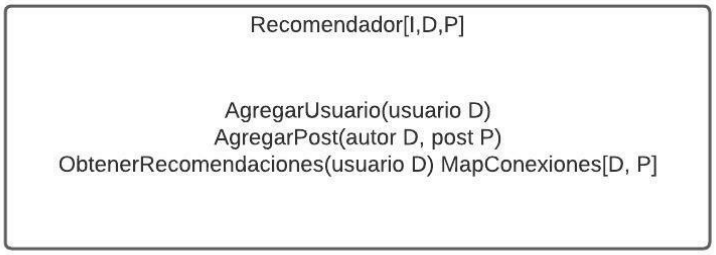
La solución diseñada de Algogram consta de dos partes, un programa principal que se encarga de manejar el input del usuario y una parte lógica que describe el funcionamiento de la red social.

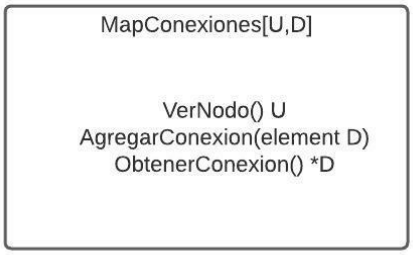
La captura del input del usuario y la validación de comandos ocurre en el programa principal sin una estructura o un TDA dedicado a esto. Esta parte se encarga de saber que ejecutar y lo delega a la parte lógica.

La parte lógica de Algogram tiene como base 4 interfaces. Y 2 interfaces con sus extensiones que se usarán para definir constraints en tipos. Las 4 interfaces base son:









Las dos interfaces para constraints son DatosUsuario y Post.

Con estas interfaces en Algogram se usan 7 TDAs.

1. Un UsuarioAlgogram que se encarga meramente de definir que datos se guardan del usuario y si se puede o no modificar estos. Se usan interfaces para saber más fácilmente los constraints de diferentes componentes sobre este TDA y para poder modificar más simplemente la implementación usada en el sistema. Todas parten de DatosUsuario.

2. El PostAlgogram, que va a guardar datos, como lo es su autor, su id, su contenido y quienes le dieron like. Y va a también a definir como sería el formato de visualización, que mostrara cuando se vea el post y para cuando se vean sus likes y permitir que los usuarios le den like. También se utilizan diferentes interfaces para definir diferentes comportamientos las cuales otros componentes usarán de como constraints. Con la base de Post y extensiones como lo sería un PostLikeable que puede ser likeado. PostAlgogram implementa PostLikeable, y para los likes dado el requerimiento de

complejidad de agregar like de O(log P) y mostrar likes de O(U) se usara un ABB internamente.

3. Un IdManager con id numérico que implementa IdManager. Para efectos de velocidad y fomentado por el que no se pueda borrar se usa un arreglo por detrás.

4. El TDA UserManagerAlgogram, implementación de IdManager, que es el encargado de guardar a los Usuarios con sus identificadores (sus nombres) y también validar si estos existen. Para fines de velocidad, dado el requerimiento de login O(1) y dado que la clave es una string, se usará un hash cerrado.

5. La ConexionAlgogram, que implementa MapasConexiones, que se encarga de representar la conexión en este caso de un Usuario, a uno o varios Posts, que en este caso representan los posts a ver. Las conexiones, posts agregados serán ordenados según el criterio que se dio al crearse y al obtener se cambia a otra conexion, para esto y con propositos de tener agregar y obtener O(log p) se usa un Heap, la lógica de comparacion, otra vez, es dada al crearse.

6. El RecomendadorAlgogram que implementa Recomendador, se encarga de ser el definidor final de la conexión entre Usuarios y Posts, este TDA se encarga de definir como será la conexión entre estos. En este caso la relación de un usuario a varios posts, bajo una lógica de obtención ordenada por una función afinidad o cronología de los posts. Permitirá obtener las recomendaciones para un Usuario en especifico, agregar usuarios al cual hacer recomendaciones y agregar posts a recomendar. Para fines de velocidad las Conexiones de cada usuario, sus recomendaciones a ver, serán guardadas en un IdManager numérico (3.) aprovechando que se tiene el índice de archivo para la función de afinidad.

7. Finalmente el SesionAlgogram que implementa SesionManager, que inicializará algogram y seria el TDA con el que tendría contacto el programa principal que recibe el input. La idea esta que este SesionManager tome la decisión final sobre los componentes, seleccionar un tipo de Usuario y Post, los tipos de identificadores, un recomendador, y ademas se encargue de mantener la sesion de logeo y permita una abstracción en acciones que representan los comandos que el programa principal tendra a disposicion.