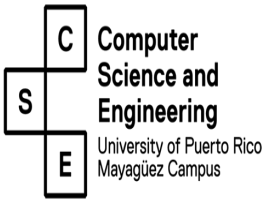
UNIVERSITY OF PUERTO RICO
MAYAGUEZ CAMPUS
COLLEGE OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING
DEPARTMENT



ER Diagram Report

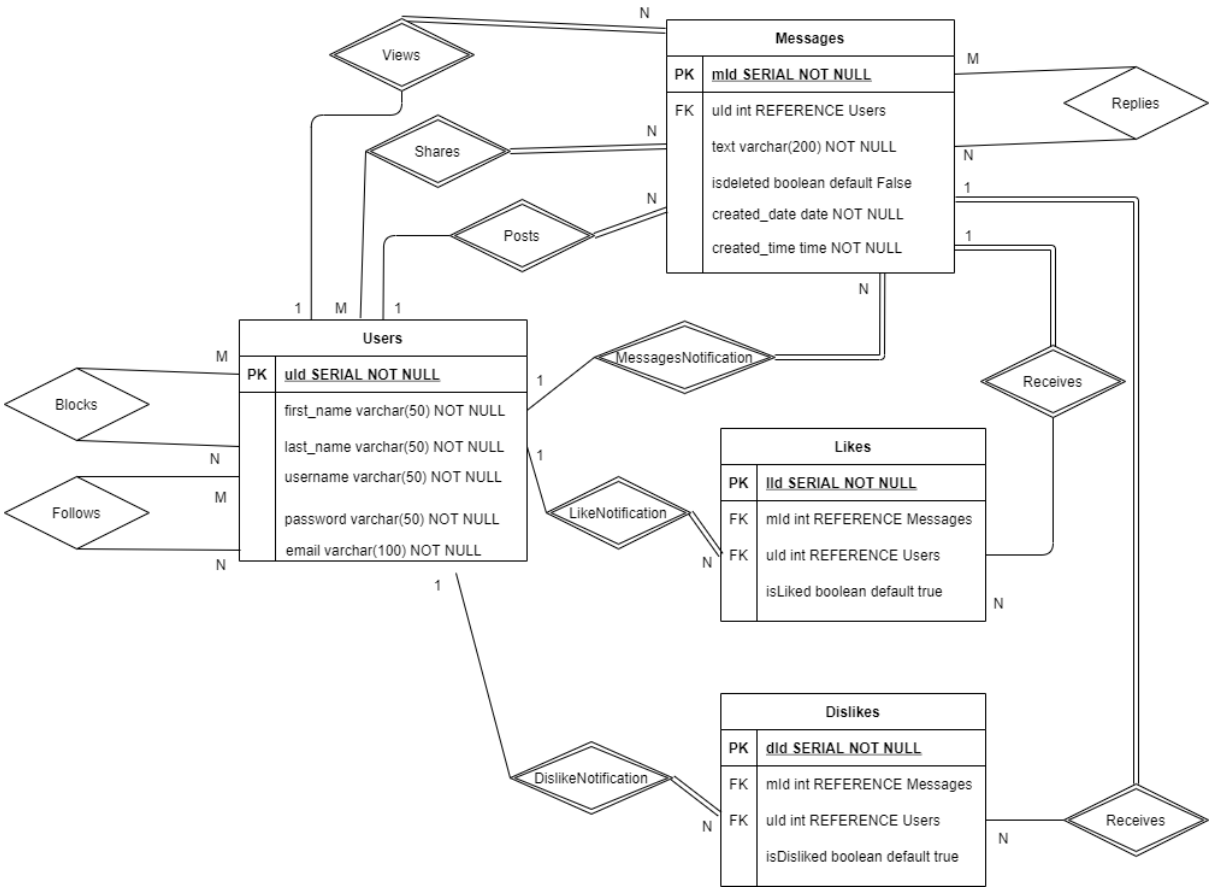
Anthony Márquez Camacho
David Carrión Beníquez
Estefanía Torres Collado
Christopher Vegerano López

CIIC 4060

Prof. Manuel Rodriguez

Abril 12, 2021

ER Diagram


Messages
PK
mId SERIAL NOT NULL
FK
uId int REFERENCE Users
text varchar(200) NOT NULL
isdeleted boolean default False
created_date date NOT NULL
created_time time NOT NULL
Users
PK
uId SERIAL NOT NULL
first_name varchar(50) NOT NULL
last_name varchar(50) NOT NULL
username varchar(50) NOT NULL
password varchar(50) NOT NULL
email varchar(100) NOT NULL
Likes
PK
lId SERIAL NOT NULL
FK
mId int REFERENCE Messages
FK
uId int REFERENCE Users
isLiked boolean default true
Dislikes
PK
dId SERIAL NOT NULL
FK
mId int REFERENCE Messages
FK
uId int REFERENCE Users
isDisliked boolean default true
Views
Shares
Posts
Replies
Blocks
Follows
MessagesNotification
LikeNotification
DislikeNotification
Receives
Receives
N
M
1

ER a Tablas

Users

La tabla de Users hizo un mapping de la tabla literal donde se añadió su uid como Primary Key incrementando automáticamente, su nombre, sus apellidos, un nombre de usuario, una contraseña y un correo electrónico. Todos estos atributos son varchar() NOT NULL, ya que son textos requeridos en todo momento como parte de las características del usuario.

Messages

La tabla de Messages hizo un mapping en la cual se añadió su mid como Primary Key incrementando automáticamente, uid como int con Referencia al uid de la tabla de Users, text que es el mensaje, isdeleted como un boolean cuya función es prevenir el borrar datos de la tabla sino que simplemente se muestran o no de acuerdo al estado de la variable, created_date y created_time que son la fecha y hora de cuando se creó el mensaje. Todos estos atributos son NOT NULL, ya que son necesarios en todo momento. En cuanto a la relación de Posts no se creó una nueva tabla, ya que la tabla de Messages guarda el id del usuario que creó el mensaje (uid), para la relación de Shares y Replies se crearon tablas donde se guarda el uid del usuario existente que desea compartir/responder a un mensaje, junto con el id de la persona que creó el mensaje y en el caso de Replies con el texto que se respondió, esto se hizo ya que son relaciones 1-N y M-N y pues es más sencillo guardarlos en una tabla aparte.

Blocks

La relación de blocks, que es many to many (M a N), se le hizo un mapping a tabla, en donde se incluyeron las columnas de registered_uid, blocked_uid y is_deleted. El registered_uid bloquea al blocked_uid. También se incluyó la columna de is_deleted, que tiene como valor default “false” para no eliminar récords de la tabla. Esta columna también asiste en la operación de desbloquear a un usuario, en donde simplemente is_deleted se ajusta a true. Un usuario (A) tendrá a otro usuario (B) bloqueado cuando el usuario A aparezca en registered_uid, el usuario B aparezca en blocked_uid y is_deleted sea falso.

Follows

Para la relación de follows, la cual es many to many (M a N), se hizo un mapping a tabla utilizando el fid como su Primary Key que incrementa automáticamente, también se incluyen las columnas uid, followed_user y is_deleted. El uid, que sería el usuario "en sesión", sigue al followed_user. La columna de is_deleted evita la eliminación del "follow" y tiene como valor default "false". Un usuario (A) sigue a otro usuario (B) cuando el usuario A aparezca como uid, el usuario B aparezca como followed_user y is_deleted sea falso.

Likes

La tabla de Likes se le hizo un mapping utilizando el lid como su Primary Key que incrementa automáticamente, mid como int lo que hace referencia a al mid de la tabla de Message y uid que hace referencia al uid de Users. Ambos mid y uid se interpretan como Foreing Keys en la tabla. También se implementó el isLiked lo cual es un boolean que nos dice si el usuario le remueve el like a message sale True y False si es lo contrario, esto se hizo para evitar remover de la base de datos los likes. También se implementó un query que determina si un usuario le da like a un mensaje y anteriormente tenia un dislike, cambia el boolean de isDisliked a falso de la tabla Dislikes y hace true el isLiked.

En las relaciones, se tomó en consideración Receives como una relación many to one (N-1) ya que un mensaje puede tener muchos likes pero solo es posibles si se hace a un mensaje. Dicha relación fue posible tomando el id de message (mid), user id (uid) y conectandolos al id de likes (lid). Esto nos permite ver que like está asociado al mensaje y de qué usuario viene dicho like.

Dislikes

La tabla de Dislikes se le hizo un mapping utilizando el did como su Primary Key que incrementa automáticamente, mid como int lo que hace referencia a al mid de la tabla de Message y uid que hace referencia al uid de Users. Ambos mid y uid se interpretan como Foreing Keys en la tabla. También se implementó el isDisliked lo cual es un boolean que nos dice si el usuario le remueve el dislike a message sale True y False si es lo contrario, esto se hizo para evitar remover de la base de datos los

dislikes. También se implementó un query que determina si un usuario le da dislike a un mensaje y anteriormente tenia un like, cambia el boolean de isLiked a falso de la tabla likes y hace true el isDisliked.

En las relaciones, se tomó en consideración Receives como una relación many to one (N-1) ya que un mensaje puede tener muchos dislikes pero sólo es posibles si se hace a un mensaje. Dicha relación fue posible tomando el id de message (mid), user id (uid) y conectandolos al id de dislikes (did). Esto nos permite ver que dislike está asociado al mensaje y de qué usuario viene dicho dislike.