package com.twitter.tweetypie

package handler

import com.twitter.tweetypie.store.UpdatePossiblySensitiveTweet

import com.twitter.tweetypie.thriftscala.UpdatePossiblySensitiveTweetRequest

import com.twitter.tweetypie.util.TweetLenses

object UpdatePossiblySensitiveTweetHandler {

type Type = FutureArrow[UpdatePossiblySensitiveTweetRequest, Unit]

def apply(

tweetGetter: FutureArrow[TweetId, Tweet],

userGetter: FutureArrow[UserId, User],

updatePossiblySensitiveTweetStore: FutureEffect[UpdatePossiblySensitiveTweet.Event]

): Type =

FutureArrow { request =>

val nsfwAdminMutation = Mutation[Boolean](\_ => request.nsfwAdmin).checkEq

val nsfwUserMutation = Mutation[Boolean](\_ => request.nsfwUser).checkEq

val tweetMutation =

TweetLenses.nsfwAdmin

.mutation(nsfwAdminMutation)

.also(TweetLenses.nsfwUser.mutation(nsfwUserMutation))

for {

originalTweet <- tweetGetter(request.tweetId)

\_ <- tweetMutation(originalTweet) match {

case None => Future.Unit

case Some(mutatedTweet) =>

userGetter(getUserId(originalTweet))

.map { user =>

UpdatePossiblySensitiveTweet.Event(

tweet = mutatedTweet,

user = user,

timestamp = Time.now,

byUserId = request.byUserId,

nsfwAdminChange = nsfwAdminMutation(TweetLenses.nsfwAdmin.get(originalTweet)),

nsfwUserChange = nsfwUserMutation(TweetLenses.nsfwUser.get(originalTweet)),

note = request.note,

host = request.host

)

}

.flatMap(updatePossiblySensitiveTweetStore)

}

} yield ()

}

}