package com.twitter.follow\_recommendations.common.clients.geoduck

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.models.GeohashAndCountryCode

import com.twitter.stitch.Stitch

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class UserLocationFetcher @Inject() (

locationServiceClient: LocationServiceClient,

reverseGeocodeClient: ReverseGeocodeClient,

statsReceiver: StatsReceiver) {

private val stats: StatsReceiver = statsReceiver.scope("user\_location\_fetcher")

private val totalRequestsCounter = stats.counter("requests")

private val emptyResponsesCounter = stats.counter("empty")

private val locationServiceExceptionCounter = stats.counter("location\_service\_exception")

private val reverseGeocodeExceptionCounter = stats.counter("reverse\_geocode\_exception")

def getGeohashAndCountryCode(

userId: Option[Long],

ipAddress: Option[String]

): Stitch[Option[GeohashAndCountryCode]] = {

totalRequestsCounter.incr()

val lscLocationStitch = Stitch

.collect {

userId.map(locationServiceClient.getGeohashAndCountryCode)

}.rescue {

case \_: Exception =>

locationServiceExceptionCounter.incr()

Stitch.None

}

val ipLocationStitch = Stitch

.collect {

ipAddress.map(reverseGeocodeClient.getGeohashAndCountryCode)

}.rescue {

case \_: Exception =>

reverseGeocodeExceptionCounter.incr()

Stitch.None

}

Stitch.join(lscLocationStitch, ipLocationStitch).map {

case (lscLocation, ipLocation) => {

val geohash = lscLocation.flatMap(\_.geohash).orElse(ipLocation.flatMap(\_.geohash))

val countryCode =

lscLocation.flatMap(\_.countryCode).orElse(ipLocation.flatMap(\_.countryCode))

(geohash, countryCode) match {

case (None, None) =>

emptyResponsesCounter.incr()

None

case \_ => Some(GeohashAndCountryCode(geohash, countryCode))

}

}

}

}

}