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

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

import com.twitter.geoduck.common.thriftscala.LocationSource

import com.twitter.geoduck.common.thriftscala.PlaceQuery

import com.twitter.geoduck.common.thriftscala.TransactionLocation

import com.twitter.geoduck.common.thriftscala.UserLocationRequest

import com.twitter.geoduck.thriftscala.LocationService

import com.twitter.stitch.Stitch

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class LocationServiceClient @Inject() (locationService: LocationService.MethodPerEndpoint) {

def getGeohashAndCountryCode(userId: Long): Stitch[GeohashAndCountryCode] = {

Stitch

.callFuture {

locationService

.userLocation(

UserLocationRequest(

Seq(userId),

Some(PlaceQuery(allPlaceTypes = Some(true))),

simpleReverseGeocode = true))

.map(\_.found.get(userId)).map { transactionLocationOpt =>

val geohashOpt = transactionLocationOpt.flatMap(getGeohashFromTransactionLocation)

val countryCodeOpt =

transactionLocationOpt.flatMap(\_.simpleRgcResult.flatMap(\_.countryCodeAlpha2))

GeohashAndCountryCode(geohashOpt, countryCodeOpt)

}

}

}

private[this] def getGeohashFromTransactionLocation(

transactionLocation: TransactionLocation

): Option[String] = {

transactionLocation.geohash.flatMap { geohash =>

val geohashPrefixLength = transactionLocation.locationSource match {

// if location source is logical, keep the first 4 chars in geohash

case Some(LocationSource.Logical) => Some(4)

// if location source is physical, keep the prefix according to accuracy

// accuracy is the accuracy of GPS readings in the unit of meter

case Some(LocationSource.Physical) =>

transactionLocation.coordinate.flatMap { coordinate =>

coordinate.accuracy match {

case Some(accuracy) if (accuracy < 50) => Some(7)

case Some(accuracy) if (accuracy < 200) => Some(6)

case Some(accuracy) if (accuracy < 1000) => Some(5)

case Some(accuracy) if (accuracy < 50000) => Some(4)

case Some(accuracy) if (accuracy < 100000) => Some(3)

case \_ => None

}

}

case Some(LocationSource.Model) => Some(4)

case \_ => None

}

geohashPrefixLength match {

case Some(l: Int) => geohash.stringGeohash.map(\_.take(l))

case \_ => None

}

}

}

}