package com.twitter.follow\_recommendations.common.candidate\_sources.addressbook

import com.twitter.cds.contact\_consent\_state.thriftscala.PurposeOfProcessing

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.clients.addressbook.AddressbookClient

import com.twitter.follow\_recommendations.common.clients.addressbook.models.EdgeType

import com.twitter.follow\_recommendations.common.clients.addressbook.models.RecordIdentifier

import com.twitter.follow\_recommendations.common.clients.email\_storage\_service.EmailStorageServiceClient

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

import com.twitter.follow\_recommendations.common.utils.RescueWithStatsUtils.rescueOptionalWithStats

import com.twitter.follow\_recommendations.common.utils.RescueWithStatsUtils.rescueWithStats

import com.twitter.hermit.model.Algorithm

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidateSource

import com.twitter.product\_mixer.core.model.common.identifier.CandidateSourceIdentifier

import com.twitter.product\_mixer.core.model.marshalling.request.HasClientContext

import com.twitter.stitch.Stitch

import com.twitter.strato.generated.client.onboarding.userrecs.ReverseEmailContactsClientColumn

import com.twitter.timelines.configapi.HasParams

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class ReverseEmailBookSource @Inject() (

reverseEmailContactsClientColumn: ReverseEmailContactsClientColumn,

essClient: EmailStorageServiceClient,

addressBookClient: AddressbookClient,

statsReceiver: StatsReceiver = NullStatsReceiver)

extends CandidateSource[HasParams with HasClientContext, CandidateUser] {

override val identifier: CandidateSourceIdentifier = ReverseEmailBookSource.Identifier

private val rescueStats = statsReceiver.scope("ReverseEmailBookSource")

/\*\*

\* Generate a list of candidates for the target

\*/

override def apply(target: HasParams with HasClientContext): Stitch[Seq[CandidateUser]] = {

val reverseCandidatesFromEmail = target.getOptionalUserId

.map { userId =>

val verifiedEmailStitchOpt =

rescueOptionalWithStats(

essClient.getVerifiedEmail(userId, PurposeOfProcessing.ContentRecommendations),

rescueStats,

"getVerifiedEmail")

verifiedEmailStitchOpt.flatMap { emailOpt =>

rescueWithStats(

addressBookClient.getUsers(

userId = userId,

identifiers = emailOpt

.map(email =>

RecordIdentifier(userId = None, email = Some(email), phoneNumber = None)).toSeq,

batchSize = ReverseEmailBookSource.NumEmailBookEntries,

edgeType = ReverseEmailBookSource.DefaultEdgeType,

fetcherOption =

if (target.params(AddressBookParams.ReadFromABV2Only)) None

else Some(reverseEmailContactsClientColumn.fetcher)

),

rescueStats,

"AddressBookClient"

)

}

}.getOrElse(Stitch.Nil)

reverseCandidatesFromEmail.map(

\_.take(ReverseEmailBookSource.NumEmailBookEntries)

.map(

CandidateUser(\_, score = Some(CandidateUser.DefaultCandidateScore))

.withCandidateSource(identifier))

)

}

}

object ReverseEmailBookSource {

val Identifier: CandidateSourceIdentifier = CandidateSourceIdentifier(

Algorithm.ReverseEmailBookIbis.toString)

val NumEmailBookEntries: Int = 500

val IsPhone = false

val DefaultEdgeType: EdgeType = EdgeType.Reverse

}