package com.alvis.trace.activity;

import java.lang.reflect.Method;

import java.util.ArrayList;

import java.util.List;

import android.Manifest;

import android.annotation.TargetApi;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Build;

import android.os.Bundle;

import android.provider.Settings;

import android.view.KeyEvent;

import com.alvis.trace.R;

/\*\*

\* 继承了Activity，实现Android6.0的运行时权限检测

\* 需要进行运行时权限检测的Activity可以继承这个类

\*

\* @since 2.5.0

\*/

public class CheckPermissionsActivity extends Activity {

//是否需要检测后台定位权限，设置为true时，如果用户没有给予后台定位权限会弹窗提示

private boolean needCheckBackLocation = false;

//如果设置了target > 28，需要增加这个权限，否则不会弹出"始终允许"这个选择框

private static String BACKGROUND\_LOCATION\_PERMISSION = "android.permission.ACCESS\_BACKGROUND\_LOCATION";

/\*\*

\* 需要进行检测的权限数组

\*/

protected String[] needPermissions = {

Manifest.permission.ACCESS\_COARSE\_LOCATION,

Manifest.permission.ACCESS\_FINE\_LOCATION,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE,

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.READ\_PHONE\_STATE

};

private static final int PERMISSON\_REQUESTCODE = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

if(Build.VERSION.SDK\_INT > 28

&& getApplicationContext().getApplicationInfo().targetSdkVersion > 28) {

needPermissions = new String[] {

Manifest.permission.ACCESS\_COARSE\_LOCATION,

Manifest.permission.ACCESS\_FINE\_LOCATION,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE,

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.READ\_PHONE\_STATE,

BACKGROUND\_LOCATION\_PERMISSION

};

}

}

/\*\*

\* 判断是否需要检测，防止不停的弹框

\*/

private boolean isNeedCheck = true;

@Override

protected void onResume() {

super.onResume();

if (Build.VERSION.SDK\_INT >= 23

&& getApplicationInfo().targetSdkVersion >= 23) {

if (isNeedCheck) {

checkPermissions(needPermissions);

}

}

}

/\*\*

\*

\* @param permissions

\* @since 2.5.0

\*

\*/

private void checkPermissions(String... permissions) {

try {

if (Build.VERSION.SDK\_INT >= 23

&& getApplicationInfo().targetSdkVersion >= 23) {

List<String> needRequestPermissonList = findDeniedPermissions(permissions);

if (null != needRequestPermissonList

&& needRequestPermissonList.size() > 0) {

String[] array = needRequestPermissonList.toArray(new String[needRequestPermissonList.size()]);

Method method = getClass().getMethod("requestPermissions", new Class[]{String[].class,

int.class});

method.invoke(this, array, PERMISSON\_REQUESTCODE);

}

}

} catch (Throwable e) {

}

}

/\*\*

\* 获取权限集中需要申请权限的列表

\*

\* @param permissions

\* @return

\* @since 2.5.0

\*

\*/

private List<String> findDeniedPermissions(String[] permissions) {

List<String> needRequestPermissonList = new ArrayList<String>();

if (Build.VERSION.SDK\_INT >= 23

&& getApplicationInfo().targetSdkVersion >= 23){

try {

for (String perm : permissions) {

Method checkSelfMethod = getClass().getMethod("checkSelfPermission", String.class);

Method shouldShowRequestPermissionRationaleMethod = getClass().getMethod("shouldShowRequestPermissionRationale",

String.class);

if ((Integer)checkSelfMethod.invoke(this, perm) != PackageManager.PERMISSION\_GRANTED

|| (Boolean)shouldShowRequestPermissionRationaleMethod.invoke(this, perm)) {

if(!needCheckBackLocation

&& BACKGROUND\_LOCATION\_PERMISSION.equals(perm)) {

continue;

}

needRequestPermissonList.add(perm);

}

}

} catch (Throwable e) {

}

}

return needRequestPermissonList;

}

/\*\*

\* 检测是否所有的权限都已经授权

\* @param grantResults

\* @return

\* @since 2.5.0

\*

\*/

private boolean verifyPermissions(int[] grantResults) {

for (int result : grantResults) {

if (result != PackageManager.PERMISSION\_GRANTED) {

return false;

}

}

return true;

}

@TargetApi(23)

public void onRequestPermissionsResult(int requestCode,

String[] permissions, int[] paramArrayOfInt) {

if (requestCode == PERMISSON\_REQUESTCODE) {

if (!verifyPermissions(paramArrayOfInt)) {

showMissingPermissionDialog();

isNeedCheck = false;

}

}

}

/\*\*

\* 显示提示信息

\*

\* @since 2.5.0

\*

\*/

private void showMissingPermissionDialog() {

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle(R.string.notifyTitle);

builder.setMessage(R.string.notifyMsg);

// 拒绝, 退出应用

builder.setNegativeButton(R.string.cancel,

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

finish();

}

});

builder.setPositiveButton(R.string.setting,

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

startAppSettings();

}

});

builder.setCancelable(false);

builder.show();

}

/\*\*

\* 启动应用的设置

\*

\* @since 2.5.0

\*

\*/

private void startAppSettings() {

Intent intent = new Intent(

Settings.ACTION\_APPLICATION\_DETAILS\_SETTINGS);

intent.setData(Uri.parse("package:" + getPackageName()));

startActivity(intent);

}

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

if(keyCode == KeyEvent.KEYCODE\_BACK){

this.finish();

return true;

}

return super.onKeyDown(keyCode, event);

}

}

package com.alvis.trace.activity;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.app.Service;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.IntentFilter;

import android.graphics.Color;

import android.os.Bundle;

import android.os.Looper;

import android.os.Vibrator;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.localbroadcastmanager.content.LocalBroadcastManager;

import com.alvis.trace.common.QU\_AGRQ\_C;

import com.alvis.trace.common.QU\_AGRQ\_P;

import com.alvis.trace.common.Vertex;

import com.alvis.trace.model.MapData;

import com.alvis.trace.model.TMessage;

import com.alvis.trace.model.User1;

import com.alvis.trace.util.CoordinateConversion;

import com.alvis.trace.util.MapUtils;

import com.alvis.trace.util.MyActivityManager;

import com.amap.api.location.AMapLocation;

import com.amap.api.location.AMapLocationClient;

import com.amap.api.location.AMapLocationClientOption;

import com.amap.api.location.AMapLocationClientOption.AMapLocationMode;

import com.amap.api.location.AMapLocationListener;

import com.amap.api.maps2d.AMap;

import com.amap.api.maps2d.AMapUtils;

import com.amap.api.maps2d.CameraUpdateFactory;

import com.amap.api.maps2d.LocationSource;

import com.amap.api.maps2d.MapView;

import com.amap.api.maps2d.model.BitmapDescriptorFactory;

import com.amap.api.maps2d.model.Circle;

import com.amap.api.maps2d.model.LatLng;

import com.amap.api.maps2d.model.Marker;

import com.amap.api.maps2d.model.MarkerOptions;

import com.alvis.trace.R;

import java.io.IOException;

import java.io.ObjectOutputStream;

public class DriverActivity extends Activity implements LocationSource,

AMapLocationListener {

private AMap aMap;

private MapView mapView;

private OnLocationChangedListener mListener;

private AMapLocationClient mlocationClient;

private AMapLocationClientOption mLocationOption;

private EditText editTextLat;

private EditText editTextLng;

private double lat, lng;

private Marker destMarker;

private Circle circleReveal;

private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String message = intent.getStringExtra("message");

Log.d("receiver", "Got message: " + message);

if (message.equals("SocketFailed")) {

OnSocketFailed();

// Toast.makeText(getApplicationContext(), "hahaha", Toast.LENGTH\_LONG).show();

} else if (message.equals("Tmessage")) {

String emMapData = intent.getStringExtra("TmessageEM");

OnReceivedCircleData(emMapData);//这里要执行一个将自身位置与加密的圆形数据进行计算的方法。

} else if (message.equals("TmCircleData")) {//这里还有一个判断是当车主位于圆内的时候，客户端通过服务器返回True让车主选择是否同意，同意的话将会显示在打车的人手机上

int toUserId = Integer.parseInt(intent.getStringExtra("toUserId"));

int fromUserId = Integer.parseInt(intent.getStringExtra("fromUserId"));

String fromUserName = intent.getStringExtra("fromUserName");

String destData = intent.getStringExtra("destination");

OnReceivedRespCircleData(toUserId, fromUserId, fromUserName, destData);

} else if (message.equals("UserRealPos")) {

String userNameReal = intent.getStringExtra("userName");

int userIdReal = Integer.parseInt(intent.getStringExtra("userId"));

String[] realPos = intent.getStringExtra("userPos").split(",");

double realPosLat = Double.parseDouble(realPos[0]);

double realPosLng = Double.parseDouble(realPos[1]);

Log.e("用户真实信息的消息", userNameReal);

OnReceivedRealUserPos(userIdReal, userNameReal, realPosLat, realPosLng);

}else if(message.equals("UserDenial")){

String userNameDenial = intent.getStringExtra("userName");

OnReceivedUserDenial(userNameDenial);

}

}

};

private void OnReceivedUserDenial(String userNameDenial) {

Toast.makeText(getApplicationContext(),"用户："+userNameDenial+" 拒绝了您的接单。",Toast.LENGTH\_LONG).show();

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_carer);

mapView = (MapView) findViewById(R.id.map);

editTextLat = (EditText) findViewById(R.id.x);

editTextLat.setCursorVisible(false);

editTextLat.setSelectAllOnFocus(true);

editTextLng = (EditText) findViewById(R.id.y);

editTextLng.setCursorVisible(false);

editTextLng.setSelectAllOnFocus(true);

mapView.onCreate(savedInstanceState);// 此方法必须重写

init();

LocalBroadcastManager.getInstance(this).registerReceiver(mMessageReceiver,

new IntentFilter("notify-to-refresh-driver"));

}

/\*\*

\* 初始化AMap对象

\*/

private void init() {

if (aMap == null) {

aMap = mapView.getMap();

setUpMap();

}

}

/\*\*

\* 设置一些amap的属性

\*/

private void setUpMap() {

// 自定义系统定位小蓝点

aMap.setLocationSource(this);// 设置定位监听

aMap.getUiSettings().setMyLocationButtonEnabled(true);// 设置默认定位按钮是否显示

aMap.setMyLocationEnabled(true);// 设置为true表示显示定位层并可触发定位，false表示隐藏定位层并不可触发定位，默认是false

AMap.OnMarkerDragListener markerDragListener = new AMap.OnMarkerDragListener() {

@Override

public void onMarkerDragStart(Marker arg0) {

Vibrator vib = (Vibrator) getApplicationContext().getSystemService(Service.VIBRATOR\_SERVICE);

if(vib.hasVibrator()){ //判断手机硬件是否有振动器

vib.vibrate(100);

}

editTextLat.clearFocus();

editTextLng.clearFocus();

}

@Override

public void onMarkerDragEnd(Marker arg0) {

lat = arg0.getPosition().latitude;

lng = arg0.getPosition().longitude;

aMap.clear();

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_BLUE));

otMarkerOptions.position(new LatLng(lat, lng));

otMarkerOptions.draggable(true);

aMap.addMarker(otMarkerOptions);

aMap.moveCamera(CameraUpdateFactory.changeLatLng(new LatLng(lat, lng)));

editTextLat.setText(lat + "");

editTextLng.setText(lng + "");

final double latf = lat;

final double lngf = lng;

new Thread() {

@Override

public void run() {

Looper.prepare();

sendEncryptedMapData(latf, lngf);

Toast.makeText(getApplicationContext(), "位置信息加密完成！", Toast.LENGTH\_LONG).show();

Looper.loop();

}

}.start();

}

@Override

public void onMarkerDrag(Marker arg0) {

// TODO Auto-generated method stub

}

};

aMap.setOnMarkerDragListener(markerDragListener);

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onResume() {

super.onResume();

mapView.onResume();

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onPause() {

super.onPause();

mapView.onPause();

deactivate();

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

mapView.onSaveInstanceState(outState);

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onDestroy() {

super.onDestroy();

mapView.onDestroy();

LocalBroadcastManager.getInstance(this).unregisterReceiver(mMessageReceiver);

}

/\*\*

\* 定位成功后回调函数

\*/

@Override

public void onLocationChanged(final AMapLocation amapLocation) {

if (mListener != null && amapLocation != null) {

if (amapLocation != null

&& amapLocation.getErrorCode() == 0) {

lat = amapLocation.getLatitude();

lng = amapLocation.getLongitude();

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_BLUE));

otMarkerOptions.position(new LatLng(amapLocation.getLatitude(), amapLocation.getLongitude()));

otMarkerOptions.draggable(true);

aMap.addMarker(otMarkerOptions);

aMap.moveCamera(CameraUpdateFactory.zoomTo(12));

aMap.moveCamera(CameraUpdateFactory.changeLatLng(new LatLng(amapLocation.getLatitude(), amapLocation.getLongitude())));

new Thread() {

@Override

public void run() {

sendEncryptedMapData(amapLocation.getLatitude(), amapLocation.getLongitude());

Looper.prepare();

Toast.makeText(getApplicationContext(), "位置信息加密完成！", Toast.LENGTH\_LONG).show();

Looper.loop();

}

}.start();

mListener.onLocationChanged(amapLocation);

} else {

String errText = "定位失败," + amapLocation.getErrorCode() + ": " + amapLocation.getErrorInfo();

Log.e("AmapErr", errText);

}

}

}

/\*\*

\* 激活定位

\*/

@Override

public void activate(OnLocationChangedListener listener) {

mListener = listener;

if (mlocationClient == null) {

mlocationClient = new AMapLocationClient(this);

mLocationOption = new AMapLocationClientOption();

//设置定位监听

mlocationClient.setLocationListener(this);

//设置定位时间间隔

mLocationOption.setInterval(2000);

//设置为高精度定位模式

mLocationOption.setLocationMode(AMapLocationMode.Hight\_Accuracy);

mLocationOption.setOnceLocation(true);

//设置定位参数

mlocationClient.setLocationOption(mLocationOption);

// 此方法为每隔固定时间会发起一次定位请求，为了减少电量消耗或网络流量消耗，

// 注意设置合适的定位时间的间隔（最小间隔支持为2000ms），并且在合适时间调用stopLocation()方法来取消定位请求

// 在定位结束后，在合适的生命周期调用onDestroy()方法

// 在单次定位情况下，定位无论成功与否，都无需调用stopLocation()方法移除请求，定位sdk内部会移除

mlocationClient.startLocation();

}

}

/\*\*

\* 停止定位

\*/

@Override

public void deactivate() {

mListener = null;

if (mlocationClient != null) {

mlocationClient.stopLocation();

mlocationClient.onDestroy();

}

mlocationClient = null;

}

public void OnSocketFailed() {

Dialog dialog = new AlertDialog.Builder(DriverActivity.this).setTitle("操作结果")

.setItems(new String[]{"服务异常，请退出后重新登录 "}, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

}

}).setPositiveButton("CONFIRM", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

mapView.onDestroy();

LocalBroadcastManager.getInstance(getApplicationContext()).unregisterReceiver(mMessageReceiver);

finish();

}

}).create();

dialog.show();

}

private void OnReceivedRealUserPos(int userIdReal, String userNameReal, double realPosLat, double realPosLng) {//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Log.e("OnReceivedRealUserPos: ", "收到用户真实位置:" + realPosLat + "," + realPosLng);

if (destMarker != null) destMarker.remove();

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_GREEN));

otMarkerOptions.title("Customer");

otMarkerOptions.snippet(userNameReal);

otMarkerOptions.position(new LatLng(realPosLat, realPosLng));

destMarker = aMap.addMarker(otMarkerOptions);

User1 user1Real = new User1();

user1Real.setUserName(userNameReal);

user1Real.setID(userIdReal);

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage ntMessage = new TMessage();

ntMessage.setMsgType(8);

ntMessage.setFromUser(MyActivityManager.user);

ntMessage.setToUser(user1Real);//寄回给打车用户

ntMessage.setDestData(lat + "," + lng);

oos.writeObject(ntMessage);

Log.e("dialog: ", "写入数据完成！");

} catch (IOException e) {

Log.e("dialog: ", "读取输出流失败！");

}

float distance = AMapUtils.calculateLineDistance(new LatLng(lat, lng), new LatLng(realPosLat, realPosLng));

Log.e("两人距离： ", distance + "");

if (circleReveal != null) circleReveal.remove();

double centerLat = (lat + realPosLat) / 2;

double centerLng = (lng + realPosLng) / 2;

if (distance < 1000) aMap.animateCamera(CameraUpdateFactory.zoomTo(16));

else if (distance < 3000) aMap.animateCamera(CameraUpdateFactory.zoomTo(15));

else if (distance < 5000) aMap.animateCamera(CameraUpdateFactory.zoomTo(14));

else if (distance < 7000) aMap.animateCamera(CameraUpdateFactory.zoomTo(13));

aMap.animateCamera(CameraUpdateFactory.changeLatLng(new LatLng(centerLat, centerLng)));

destMarker.showInfoWindow();

}

public void OnReceivedRespCircleData(int toUserId, int fromUserId, String fromUserName, String destData) {

//弹窗用来表达是否愿意接单

AlertDialog.Builder dialog = new AlertDialog.Builder(DriverActivity.this);

dialog.setTitle("周围的用户");

dialog.setMessage("用户“" + fromUserName + "”正在打车，目的地：" + destData + ",是否愿意接单？");

final User1 fromUser = new User1();

fromUser.setID(fromUserId);

dialog.setPositiveButton("确定", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage ntMessage = new TMessage();

ntMessage.setMsgType(6);

ntMessage.setFromUser(MyActivityManager.user);

ntMessage.setToUser(fromUser);//寄回给打车用户

oos.writeObject(ntMessage);

Log.e("dialog: ", "写入数据完成！");

} catch (IOException e) {

Log.e("dialog: ", "读取输出流失败！");

}

Toast.makeText(getApplicationContext(), "接单信息已发送至服务器。", Toast.LENGTH\_LONG).show();

}

});

dialog.setNegativeButton("拒绝",

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

Toast.makeText(getApplicationContext(), "您已拒绝对方的下单申请。", Toast.LENGTH\_LONG).show();

}

}

);

dialog.setCancelable(false);

dialog.show();

}

public void OnReceivedCircleData(String emMapData) {

QU\_AGRQ\_C qu\_agrq\_c = new QU\_AGRQ\_C();

StringBuffer stringBuffer = new StringBuffer(emMapData);

CoordinateConversion coor = new CoordinateConversion();

String coorResult = coor.latLon2UTM(lat, lng);

String[] coorResults = coorResult.split(" ");

Vertex vertex = new Vertex(Long.parseLong(coorResults[2]), Long.parseLong(coorResults[3]));

StringBuffer queryMessage = null;

try {

queryMessage = new StringBuffer(qu\_agrq\_c.UF\_AGRQ\_C\_RDC(stringBuffer.toString(), vertex));

} catch (Exception e) {

Log.e("OnReceivedCircleData: ", "DriverActivity中数据处理出错！");

e.printStackTrace();

}

sendEncryptedData(queryMessage, 4);

Log.e("与圆进行计算: ", "计算数据发送完毕！");

}

/\*\*

\* 将加密的地图数据和某个位置进行计算

\*

\* @return TMessage类型的数据

\*/

public StringBuffer EncryptedCaledData(double latitude, double longitude) {

CoordinateConversion coor = new CoordinateConversion();

String coorResult = coor.latLon2UTM(latitude, longitude);

String[] coorResults = coorResult.split(" ");

Vertex vertex = new Vertex(Long.parseLong(coorResults[2]), Long.parseLong(coorResults[3]));

//这个Vertex是自身的位置，然后需要和84个加密的地图数据进行计算，得到一个mapdata的数据然后用Tmessage封装

String[] mapDatas = MapUtils.mapDatas;

MapData mapData = new MapData();

StringBuffer stringBuffer = new StringBuffer("");

QU\_AGRQ\_P qu\_agrq\_p = new QU\_AGRQ\_P();

for (String str : mapDatas) {

try {

stringBuffer.append(qu\_agrq\_p.UF\_AGRQ\_P\_RDC(str, vertex) + ";");

} catch (Exception e) {

Log.e("EncryptedCaledData: ", "数据处理出错！");

e.printStackTrace();

}

}

return stringBuffer;

}

/\*\*

\* 向服务器发送当前位置与加密地图计算后的数据

\*

\* @param latitude

\* @param longitude

\*/

public void sendEncryptedMapData(double latitude, double longitude) {

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage tMessage = new TMessage();

tMessage.setMsgType(1);

MapData mapData = new MapData();

mapData.setEncryptedMap(EncryptedCaledData(latitude, longitude));

mapData.setDataType(MapData.ECPM);

tMessage.setMapData(mapData);

oos.writeObject(tMessage);

Log.e("sendEncryptedMapData: ", "写入数据完成！");

} catch (IOException e) {

Log.e("ccst init: ", "读取输出流失败！");

}

}

/\*\*

\* 向输出流写入加密数据

\*

\* @param stringBuffer

\* @param type

\*/

public void sendEncryptedData(StringBuffer stringBuffer, int type) {

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage tMessage = new TMessage();

tMessage.setMsgType(type);

tMessage.setFromUser(MyActivityManager.user);

Log.e("sendEncryptedData\_id: ", MyActivityManager.user.getID() + "");

MapData mapData = new MapData();

mapData.setEncryptedMap(stringBuffer);

//其实这个属性并没有用到

if (type == 3) mapData.setDataType(MapData.ECSD);

else if (type == 2) mapData.setDataType(MapData.ECD);//Encrypted Circle Data

tMessage.setMapData(mapData);

oos.writeObject(tMessage);

Log.e("sendED: ", "写入数据完成！");

} catch (IOException e) {

Log.e("ED: ", "读取输出流失败！");

}

}

}

package com.alvis.trace.activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.app.ProgressDialog;

import android.app.Service;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.IntentFilter;

import android.graphics.Color;

import android.os.Bundle;

import android.os.Handler;

import android.os.Looper;

import android.os.Message;

import android.os.Vibrator;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.AdapterView;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.localbroadcastmanager.content.LocalBroadcastManager;

import com.alvis.trace.R;

import com.alvis.trace.adapter.DriverAdapter;

import com.alvis.trace.common.CircleArea;

import com.alvis.trace.common.QU\_AGRQ\_C;

import com.alvis.trace.common.QU\_AGRQ\_P;

import com.alvis.trace.common.Vertex;

import com.alvis.trace.model.ListDriver;

import com.alvis.trace.model.MapData;

import com.alvis.trace.model.TMessage;

import com.alvis.trace.model.User1;

import com.alvis.trace.util.CoordinateConversion;

import com.alvis.trace.util.MapUtils;

import com.alvis.trace.util.MyActivityManager;

import com.amap.api.location.AMapLocation;

import com.amap.api.location.AMapLocationClient;

import com.amap.api.location.AMapLocationClientOption;

import com.amap.api.location.AMapLocationClientOption.AMapLocationMode;

import com.amap.api.location.AMapLocationListener;

import com.amap.api.maps2d.AMap;

import com.amap.api.maps2d.AMapUtils;

import com.amap.api.maps2d.CameraUpdateFactory;

import com.amap.api.maps2d.LocationSource;

import com.amap.api.maps2d.MapView;

import com.amap.api.maps2d.model.BitmapDescriptorFactory;

import com.amap.api.maps2d.model.Circle;

import com.amap.api.maps2d.model.CircleOptions;

import com.amap.api.maps2d.model.LatLng;

import com.amap.api.maps2d.model.Marker;

import com.amap.api.maps2d.model.MarkerOptions;

import com.amap.api.services.core.AMapException;

import com.amap.api.services.core.LatLonPoint;

import com.amap.api.services.geocoder.GeocodeResult;

import com.amap.api.services.geocoder.GeocodeSearch;

import com.amap.api.services.geocoder.RegeocodeQuery;

import com.amap.api.services.geocoder.RegeocodeResult;

import java.io.IOException;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

public class LocationSourceActivity extends CheckPermissionsActivity implements LocationSource,

AMapLocationListener, View.OnClickListener, AMap.OnMapClickListener, GeocodeSearch.OnGeocodeSearchListener {

private static final int EVENT\_CLOSE\_PROGRESSDIALOG = 100;

private static final int EVENT\_CHANGE\_DRIVERDATA = 200;

private static final int EVENT\_SHOW\_LIST = 300;

public CoordinateConversion coor = new CoordinateConversion();

private AMap aMap;

private MapView mapView;

private OnLocationChangedListener mListener;

private AMapLocationClient mlocationClient;

private AMapLocationClientOption mLocationOption;

private EditText editTextLat;

private EditText editTextLng;

private EditText editTextRadius;

private Button buttonDo;

private EditText destTextLat;

private EditText destTextLng;

private TextView destInfo;

private Button buttonPick;

private double lat, lng;

private LatLng latLngLeftTop;

private LatLng latLngRightTop;

private LatLng latLngRightBottom;

private LatLng latLngLeftBottom;

private CircleArea queryCircle;

private Boolean isDestPicked;

private ProgressDialog progressDialog;

private GeocodeSearch geocoderSearch;

private String destInfoStr;

private Marker destMarker;

private DriverAdapter adapter;

private Circle circleReveal;

private List<ListDriver> driverList = new ArrayList<ListDriver>();

private String driverName;

private Handler mHandler = new Handler() {

@Override

public void handleMessage(@NonNull Message msg) {

switch (msg.what) {

case EVENT\_CLOSE\_PROGRESSDIALOG:

closeProgressDialog();

break;

case EVENT\_CHANGE\_DRIVERDATA:

changeList();

break;

case EVENT\_SHOW\_LIST:

showBuilder();

progressDialog.show();

break;

}

}

};

private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String message = intent.getStringExtra("message");

Log.d("receiver", "Got message: " + message);

if (message.equals("SocketFailed")) {

OnSocketFailed();

//Toast.makeText(getApplicationContext(), "hahaha", Toast.LENGTH\_LONG).show();

} else if (message.equals("CaledCircleData")) {

final String EMData = intent.getStringExtra("TmessageEM");

final String fromUserName = intent.getStringExtra("fromUserName");

final int fromUserId = Integer.parseInt(intent.getStringExtra("fromUserId"));

new Thread() {

@Override

public void run() {

OnReceivedCaledCircleData(EMData, fromUserName, fromUserId);

}

}.start();

} else if (message.equals("DriverAccepted")) {

Message msg = mHandler.obtainMessage(EVENT\_CLOSE\_PROGRESSDIALOG);

mHandler.sendMessage(msg);//关闭加载框

String fromUserName = intent.getStringExtra("fromUserName");

int fromUserId = Integer.parseInt(intent.getStringExtra("fromUserId"));

Toast.makeText(getApplicationContext(), fromUserName + ":" + fromUserId + " 接单了！", Toast.LENGTH\_LONG).show();

String tel = fromUserId == 2 ? "13090907878" : "13867963367";

ListDriver dr = new ListDriver(fromUserName, tel, R.drawable.car\_1);

driverList.add(dr);

Message msg1 = mHandler.obtainMessage(EVENT\_CHANGE\_DRIVERDATA);

mHandler.sendMessage(msg1);//通知刷新列表信息

} else if (message.equals("DriverRealPos")) {

String driverNameReal = intent.getStringExtra("driverName");

String[] realPos = intent.getStringExtra("driverPos").split(",");

double realPosLat = Double.parseDouble(realPos[0]);

double realPosLng = Double.parseDouble(realPos[1]);

OnReceivedRealDriverPos(driverNameReal, realPosLat, realPosLng);

}

}

};

/\*\*

\* 已知距离，第一个点的坐标和两点的夹角，得到另一个点的坐标

\*

\* @param distance 距离单位为km

\* @param latlngA 第一个点的经纬度

\* @param angle 从正北顺时针开始的角度

\* @return

\*/

public static LatLng getLatlng(float distance, LatLng latlngA, double angle) {

return new LatLng(latlngA.latitude + (distance \* Math.cos(angle \* Math.PI / 180)) / 111,

latlngA.longitude + (distance \* Math.sin(angle \* Math.PI / 180)) / (111 \* Math.cos(latlngA.latitude \* Math.PI / 180))

);

}

private void OnReceivedRealDriverPos(String driverNameReal, double realPosLat, double realPosLng) {

if (destMarker != null) destMarker.remove();

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_GREEN));

otMarkerOptions.title("Driver");

otMarkerOptions.snippet(driverNameReal);

otMarkerOptions.position(new LatLng(realPosLat, realPosLng));

destMarker = aMap.addMarker(otMarkerOptions);

float distance = AMapUtils.calculateLineDistance(new LatLng(lat, lng), new LatLng(realPosLat, realPosLng));

Log.e("两人距离： ", distance + "");

if (circleReveal != null) circleReveal.remove();

double centerLat = (lat + realPosLat) / 2;

double centerLng = (lng + realPosLng) / 2;

if (distance < 1000) aMap.animateCamera(CameraUpdateFactory.zoomTo(16));

else if (distance < 3000) aMap.animateCamera(CameraUpdateFactory.zoomTo(15));

else if (distance < 5000) aMap.animateCamera(CameraUpdateFactory.zoomTo(14));

else if (distance < 7000) aMap.animateCamera(CameraUpdateFactory.zoomTo(13));

aMap.animateCamera(CameraUpdateFactory.changeLatLng(new LatLng(centerLat, centerLng)));

destMarker.showInfoWindow();

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.locationsource\_activity);

mapView = findViewById(R.id.map);

editTextLat = findViewById(R.id.x);

editTextLat.setCursorVisible(false);

editTextLat.setSelectAllOnFocus(true);

editTextLng = findViewById(R.id.y);

editTextLng.setCursorVisible(false);

editTextLng.setSelectAllOnFocus(true);

editTextRadius = findViewById(R.id.r);

editTextRadius.setCursorVisible(false);

editTextRadius.setSelectAllOnFocus(true);

buttonDo = findViewById(R.id.buttonDo);

destTextLat = findViewById(R.id.destx);

destTextLat.setCursorVisible(false);

destTextLat.setSelectAllOnFocus(true);

destTextLng = findViewById(R.id.desty);

destTextLng.setCursorVisible(false);

destTextLng.setSelectAllOnFocus(true);

destInfo = findViewById(R.id.destp);

buttonPick = findViewById(R.id.buttonpick);

isDestPicked = false;

destInfo.setSelected(true);

geocoderSearch = new GeocodeSearch(this);

geocoderSearch.setOnGeocodeSearchListener(this);

progressDialog = new ProgressDialog(LocationSourceActivity.this);

progressDialog.setTitle("数据处理中");

progressDialog.setMessage("请稍候……");

progressDialog.setCancelable(true);

mapView.onCreate(savedInstanceState);// 此方法必须重写

init();

LocalBroadcastManager.getInstance(this).registerReceiver(mMessageReceiver,

new IntentFilter("notify-to-refresh"));

buttonDo.setOnClickListener(this);

buttonPick.setOnClickListener(this);

}

/\*\*

\* 初始化AMap对象

\*/

private void init() {

if (aMap == null) {

aMap = mapView.getMap();

setUpMap();

}

}

/\*\*

\* 设置一些amap的属性

\*/

private void setUpMap() {

// 自定义系统定位小蓝点

aMap.setLocationSource(this);// 设置定位监听

aMap.setOnMapClickListener(this);

aMap.getUiSettings().setMyLocationButtonEnabled(true);// 设置默认定位按钮是否显示

aMap.setMyLocationEnabled(true);// 设置为true表示显示定位层并可触发定位，false表示隐藏定位层并不可触发定位，默认是false

AMap.OnMarkerDragListener markerDragListener = new AMap.OnMarkerDragListener() {

@Override

public void onMarkerDragStart(Marker arg0) {

Vibrator vib = (Vibrator) getApplicationContext().getSystemService(Service.VIBRATOR\_SERVICE);

if (vib.hasVibrator()) { //判断手机硬件是否有振动器

vib.vibrate(100);

}

editTextLat.clearFocus();

editTextLng.clearFocus();

editTextRadius.clearFocus();

}

@Override

public void onMarkerDragEnd(Marker arg0) {

lat = arg0.getPosition().latitude;

lng = arg0.getPosition().longitude;

aMap.clear();

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_BLUE));

otMarkerOptions.position(new LatLng(lat, lng));

otMarkerOptions.draggable(true);

CircleOptions circle = null;

if (editTextRadius.getText().toString().equals("")) {

editTextRadius.setText("4000");

circle = (new CircleOptions().center(new LatLng(lat, lng)).radius(4000)

.strokeColor(Color.argb(50, 1, 1, 1)).fillColor(Color.argb(50, 1, 1, 1)).strokeWidth(1));

} else {

circle = (new CircleOptions().center(new LatLng(lat, lng)).radius(Double.parseDouble(editTextRadius.getText().toString()))

.strokeColor(Color.argb(50, 1, 1, 1)).fillColor(Color.argb(50, 1, 1, 1)).strokeWidth(1));

}

aMap.addMarker(otMarkerOptions);

aMap.moveCamera(CameraUpdateFactory.changeLatLng(new LatLng(lat, lng)));

editTextLat.setText(lat + "");

editTextLng.setText(lng + "");

circleReveal = aMap.addCircle(circle);

final double latf = lat;

final double lngf = lng;

new Thread() {

@Override

public void run() {

Looper.prepare();

sendEncryptedMapData(latf, lngf);

Toast.makeText(getApplicationContext(), "位置信息加密完成！", Toast.LENGTH\_LONG).show();

Looper.loop();

}

}.start();

}

@Override

public void onMarkerDrag(Marker arg0) {

// TODO Auto-generated method stub

}

};

aMap.setOnMarkerDragListener(markerDragListener);

}

private void generateRandomDrivers() {//加入随机数量，随机的司机

ArrayList<ListDriver> allDrivers = new ArrayList<>();

ListDriver dr1 = new ListDriver("Latonia", "13145459090", R.drawable.car\_1);

allDrivers.add(dr1);

ListDriver dr2 = new ListDriver("Laurinda", "15676768989", R.drawable.car\_2);

allDrivers.add(dr2);

ListDriver dr3 = new ListDriver("Lavinia", "13142330909", R.drawable.car\_3);

allDrivers.add(dr3);

ListDriver dr4 = new ListDriver("Lisette", "13445678909", R.drawable.car\_4);

allDrivers.add(dr4);

ListDriver dr5 = new ListDriver("Madeleine", "15589352678", R.drawable.car\_5);

allDrivers.add(dr5);

ListDriver dr6 = new ListDriver("Mathilda", "17823456754", R.drawable.car\_6);

allDrivers.add(dr6);

ListDriver dr7 = new ListDriver("Maureen", "18821239090", R.drawable.car\_7);

allDrivers.add(dr7);

ListDriver dr8 = new ListDriver("Katrina", "13923456786", R.drawable.car\_8);

allDrivers.add(dr8);

Random rand = new Random();

int k = rand.nextInt(3) + 1;//总数量为1-3个

ArrayList<Integer> randlist = new ArrayList<>();

while (k > 0) {

int num = rand.nextInt(8);

if (randlist.contains(num)) continue;

else {

randlist.add(num);

k--;

}

}

for (int num : randlist) {

driverList.add(allDrivers.get(num));

}

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onResume() {

super.onResume();

mapView.onResume();

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onPause() {

super.onPause();

mapView.onPause();

deactivate();

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

mapView.onSaveInstanceState(outState);

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onDestroy() {

super.onDestroy();

mapView.onDestroy();

LocalBroadcastManager.getInstance(this).unregisterReceiver(mMessageReceiver);

}

public void closeProgressDialog() {

progressDialog.dismiss();

}

/\*\*

\* 定位成功后回调函数

\*/

@Override

public void onLocationChanged(final AMapLocation amapLocation) {

if (mListener != null && amapLocation != null) {

if (amapLocation != null

&& amapLocation.getErrorCode() == 0) {

lat = amapLocation.getLatitude();

lng = amapLocation.getLongitude();

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_BLUE));

otMarkerOptions.position(new LatLng(amapLocation.getLatitude(), amapLocation.getLongitude()));

otMarkerOptions.draggable(true);

aMap.addMarker(otMarkerOptions);

aMap.moveCamera(CameraUpdateFactory.zoomTo(12));

aMap.moveCamera(CameraUpdateFactory.changeLatLng(new LatLng(amapLocation.getLatitude(), amapLocation.getLongitude())));

circleReveal = aMap.addCircle(new CircleOptions().center(new LatLng(amapLocation.getLatitude(), amapLocation.getLongitude())).radius(4000)

.strokeColor(Color.argb(50, 1, 1, 1)).fillColor(Color.argb(50, 1, 1, 1)).strokeWidth(1));

new Thread() {

@Override

public void run() {

sendEncryptedMapData(amapLocation.getLatitude(), amapLocation.getLongitude());

Looper.prepare();

Toast.makeText(getApplicationContext(), "位置信息加密完成！", Toast.LENGTH\_LONG).show();

Looper.loop();

}

}.start();

mListener.onLocationChanged(amapLocation);

} else {

String errText = "定位失败," + amapLocation.getErrorCode() + ": " + amapLocation.getErrorInfo();

Log.e("AmapErr", errText);

}

}

}

/\*\*

\* 激活定位

\*/

@Override

public void activate(OnLocationChangedListener listener) {

mListener = listener;

if (mlocationClient == null) {

mlocationClient = new AMapLocationClient(this);

mLocationOption = new AMapLocationClientOption();

//设置定位监听

mlocationClient.setLocationListener(this);

//设置定位时间间隔

mLocationOption.setInterval(2000);

//设置为高精度定位模式

mLocationOption.setLocationMode(AMapLocationMode.Hight\_Accuracy);

mLocationOption.setOnceLocation(true);

//设置定位参数

mlocationClient.setLocationOption(mLocationOption);

// 此方法为每隔固定时间会发起一次定位请求，为了减少电量消耗或网络流量消耗，

// 注意设置合适的定位时间的间隔（最小间隔支持为2000ms），并且在合适时间调用stopLocation()方法来取消定位请求

// 在定位结束后，在合适的生命周期调用onDestroy()方法

// 在单次定位情况下，定位无论成功与否，都无需调用stopLocation()方法移除请求，定位sdk内部会移除

mlocationClient.startLocation();

}

}

/\*\*

\* 停止定位

\*/

@Override

public void deactivate() {

mListener = null;

if (mlocationClient != null) {

mlocationClient.stopLocation();

mlocationClient.onDestroy();

}

mlocationClient = null;

}

private void changeList() {

adapter.notifyDataSetChanged();

}

public void OnReceivedCaledCircleData(String EMData, String fromUserName, int fromUserId) {

StringBuffer stringBuffer = new StringBuffer(EMData);

QU\_AGRQ\_C qu\_agrq\_c = new QU\_AGRQ\_C();

boolean result = false;

try {

result = qu\_agrq\_c.QU\_AGRQ\_P\_QRR(stringBuffer.toString(), queryCircle);

Log.e("收到与圆计算的结果:", stringBuffer.toString());

Log.e("收到与圆计算的结果: ", result + "");

} catch (Exception e) {

Log.e("用户圆形范围车辆判断: ", "数据处理出错！");

e.printStackTrace();

}

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage respMsg = new TMessage();

respMsg.setMsgType(5);

respMsg.setDestData(destInfoStr);

respMsg.setFromUser(MyActivityManager.user);

User1 fromUser = new User1();

fromUser.setID(fromUserId);

fromUser.setUserName(fromUserName);

respMsg.setToUser(fromUser);

MapData mapData = new MapData();

if (result) mapData.setEncryptedMap(new StringBuffer("true"));

else mapData.setEncryptedMap(new StringBuffer("false"));

respMsg.setMapData(mapData);

oos.writeObject(respMsg);

} catch (IOException e) {

Log.e("ED: ", "读取输出流失败！");

e.printStackTrace();

}

}

public void OnSocketFailed() {

Dialog dialog = new AlertDialog.Builder(LocationSourceActivity.this).setTitle("操作结果")

.setItems(new String[]{"服务异常，请退出后重新登录 "}, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

}

}).setPositiveButton("Confirm", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

mapView.onDestroy();

LocalBroadcastManager.getInstance(getApplicationContext()).unregisterReceiver(mMessageReceiver);

finish();

}

}).create();

dialog.show();

}

/\*\*

\* 将加密的地图数据和某个位置进行计算

\*

\* @return TMessage类型的数据

\*/

public StringBuffer EncryptedCaledData(double latitude, double longitude) {

CoordinateConversion coor = new CoordinateConversion();

String coorResult = coor.latLon2UTM(latitude, longitude);

String[] coorResults = coorResult.split(" ");

Vertex vertex = new Vertex(Long.parseLong(coorResults[2]), Long.parseLong(coorResults[3]));

//这个Vertex是自身的位置，然后需要和84个加密的地图数据进行计算，得到一个mapdata的数据然后用Tmessage封装

String[] mapDatas = MapUtils.mapDatas;

MapData mapData = new MapData();

StringBuffer stringBuffer = new StringBuffer();

QU\_AGRQ\_P qu\_agrq\_p = new QU\_AGRQ\_P();

for (String str : mapDatas) {

try {

stringBuffer.append(qu\_agrq\_p.UF\_AGRQ\_P\_RDC(str, vertex) + ";");

} catch (Exception e) {

Log.e("EncryptedCaledData: ", "数据处理出错！");

e.printStackTrace();

}

}

return stringBuffer;

}

/\*\*

\* 通过给定的圆心和半径坐标对当前的圆形范围进行加密。

\*

\* @param distance 半径，单位为m

\* @param lat 维度

\* @param lng 经度

\* @return 加密后的StringBuffer类型数据

\*/

private StringBuffer encryptedCircleData(int distance, double lat, double lng) {

String[] posString = coor.latLon2UTM(lat, lng).split(" ");

queryCircle = new CircleArea(distance, Long.parseLong(posString[2]), Long.parseLong(posString[3]));

QU\_AGRQ\_C qu\_agrq\_c = new QU\_AGRQ\_C();

String queryMessage = qu\_agrq\_c.QU\_AGRQ\_C\_QDC(queryCircle);

StringBuffer stringBuffer = new StringBuffer();

stringBuffer.append(queryMessage);

return stringBuffer;

}

public void sendEncryptedMapData(double latitude, double longitude) {

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage tMessage = new TMessage();

tMessage.setMsgType(1);

MapData mapData = new MapData();

mapData.setEncryptedMap(EncryptedCaledData(latitude, longitude));

mapData.setDataType(MapData.ECPM);

tMessage.setMapData(mapData);

oos.writeObject(tMessage);

Log.e("sendEncryptedMapData: ", "写入数据完成！");

} catch (IOException e) {

Log.e("ccst init: ", "读取输出流失败！");

}

}

public void sendEncryptedData(StringBuffer stringBuffer, int type) {

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage tMessage = new TMessage();

tMessage.setMsgType(type);

MapData mapData = new MapData();

mapData.setEncryptedMap(stringBuffer);

if (type == 3) mapData.setDataType(MapData.ECSD);

else if (type == 2) mapData.setDataType(MapData.ECD);//Encrypted Circle Data

tMessage.setMapData(mapData);

oos.writeObject(tMessage);

Log.e("sendED: ", "写入数据完成！");

} catch (IOException e) {

Log.e("ED: ", "读取输出流失败！");

}

}

public void sendRealPos(String driverName) {

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage tMessage = new TMessage();

tMessage.setMsgType(7);

MapData mapData = new MapData();

mapData.setEncryptedMap(new StringBuffer());

tMessage.setMapData(mapData);

User1 tempUser = new User1();

tempUser.setUserName(driverName);

tMessage.setToUser(tempUser);

tMessage.setFromUser(MyActivityManager.user);

tMessage.setDestData(lat + "," + lng);//用这个参数设置真实位置

oos.writeObject(tMessage);

Log.e("sendED: ", "写入数据完成！");

} catch (IOException e) {

Log.e("ED: ", "读取输出流失败！");

}

}

public void showBuilder() {

LayoutInflater inflater = LayoutInflater.from(this);

View view = inflater.inflate(R.layout.listview\_display, null);

adapter = new DriverAdapter(getApplicationContext(), R.layout.car\_item, driverList);

ListView listView = view.findViewById(R.id.listview);

listView.setAdapter(adapter);

androidx.appcompat.app.AlertDialog.Builder builder = new androidx.appcompat.app.AlertDialog.Builder(this);

builder.setTitle("Cars nearby");

builder.setView(view);

final androidx.appcompat.app.AlertDialog dialog = builder.show();

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

driverName = ((TextView) view.findViewById(R.id.driver\_name)).getText().toString();

Toast.makeText(getApplicationContext(), "您选择了" + driverName + "的车", Toast.LENGTH\_LONG).show();

sendRealPos(driverName);

dialog.dismiss();

}

});

}

@Override

public void onClick(View view) {

switch (view.getId()) {

case R.id.buttonDo:

if (!isDestPicked)

Toast.makeText(getApplicationContext(), "请先选择目的地。", Toast.LENGTH\_LONG).show();

else {

driverList.clear();

//generateRandomDrivers();

Message msg = mHandler.obtainMessage(EVENT\_SHOW\_LIST);

mHandler.sendMessage(msg);//显示加载中和list

Toast.makeText(getApplicationContext(), "开始发送数据……", Toast.LENGTH\_LONG).show();

StringBuffer stringBuffer = new StringBuffer();

//以自身为圆点，distance为半径的圆，下方的LatLng值分别为圆的外接正方形与圆的切点坐标

int distance = editTextRadius.getText().toString().equals("") ? 4000 : Integer.parseInt(editTextRadius.getText().toString());

LatLng latLngTopCenter = getLatlng((float) distance / (float) 1000, new LatLng(lat, lng), 0);

LatLng latLngRightCenter = getLatlng((float) distance / (float) 1000, new LatLng(lat, lng), 90);

LatLng latLngBottomCenter = getLatlng((float) distance / (float) 1000, new LatLng(lat, lng), 180);

LatLng latLngLeftCenter = getLatlng((float) distance / (float) 1000, new LatLng(lat, lng), 270);

latLngLeftTop = new LatLng(latLngTopCenter.latitude, latLngLeftCenter.longitude);

latLngRightTop = new LatLng(latLngTopCenter.latitude, latLngRightCenter.longitude);

latLngRightBottom = new LatLng(latLngBottomCenter.latitude, latLngRightCenter.longitude);

latLngLeftBottom = new LatLng(latLngBottomCenter.latitude, latLngLeftCenter.longitude);

//使用四个线程完成四个不同点的加密，并最后将加密的地图数据发给服务器

MyThread1 thread1 = new MyThread1();

MyThread2 thread2 = new MyThread2();

MyThread3 thread3 = new MyThread3();

MyThread4 thread4 = new MyThread4();

thread1.start();

thread2.start();

thread3.start();

thread4.start();

try {

thread1.join();

thread2.join();

thread3.join();

thread4.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

stringBuffer.append(thread1.stringBuffer).append(thread2.stringBuffer).append(thread3.stringBuffer).append(thread4.stringBuffer);

sendEncryptedData(stringBuffer, 3);

Toast.makeText(getApplicationContext(), "模糊位置加密数据发送完毕……", Toast.LENGTH\_LONG).show();

Log.e("onClick: ",

distance + "m:" + lat + "," + lng + ";" + latLngTopCenter.latitude + "," + latLngLeftCenter.longitude + ";" +

+latLngTopCenter.latitude + "," + latLngRightCenter.longitude + ";"

+ latLngBottomCenter.latitude + "," + latLngRightCenter.longitude + ";"

+ latLngBottomCenter.latitude + "," + latLngLeftCenter.longitude);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*下面为发送的圆形数据，通过服务器发送给所有范围内的车主

StringBuffer stringBufferCircle = encryptedCircleData(distance, lat, lng);

sendEncryptedData(stringBufferCircle, 2);

}

break;

case R.id.buttonpick:

if (destInfo.getText().toString().equals("") && !isDestPicked) {//还没有点地图就想确定终点

Toast.makeText(getApplicationContext(), "请先点选地图作为目的地！", Toast.LENGTH\_LONG).show();

} else if (!destInfo.getText().equals("") && isDestPicked) {//之前选好终点了，现在要取消

isDestPicked = false;

Toast.makeText(getApplicationContext(), "原目的地已取消", Toast.LENGTH\_LONG).show();

destTextLat.setText("");

destTextLng.setText("");

destInfo.setText("");

buttonPick.setText("PICK");

} else {//点地图了，并且将这次点击视为终点

isDestPicked = true;

buttonPick.setText("CANCEL");

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

break;

}

}

@Override

public void onMapClick(LatLng latLng) {

if (isDestPicked) {

Toast.makeText(getApplicationContext(), "请先取消原目的地。", Toast.LENGTH\_LONG).show();

} else {

if (destMarker != null) destMarker.remove();

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_RED));

otMarkerOptions.position(new LatLng(latLng.latitude, latLng.longitude));

destMarker = aMap.addMarker(otMarkerOptions);

destTextLat.setText(latLng.latitude + "");

destTextLng.setText(latLng.longitude + "");

getAddress(new LatLonPoint(latLng.latitude, latLng.longitude));

}

}

public void getAddress(final LatLonPoint latLonPoint) {

progressDialog.show();

RegeocodeQuery query = new RegeocodeQuery(latLonPoint, 200,

GeocodeSearch.AMAP);// 第一个参数表示一个Latlng，第二参数表示范围多少米，第三个参数表示是火系坐标系还是GPS原生坐标系

geocoderSearch.getFromLocationAsyn(query);// 设置异步逆地理编码请求

}

@Override

public void onRegeocodeSearched(RegeocodeResult result, int rCode) {

progressDialog.dismiss();

if (rCode == AMapException.CODE\_AMAP\_SUCCESS) {

if (result != null && result.getRegeocodeAddress() != null

&& result.getRegeocodeAddress().getFormatAddress() != null) {

destInfoStr = result.getRegeocodeAddress().getFormatAddress()

+ "附近";

destInfo.setText(destInfoStr);

} else {

destInfoStr = "不明位置";

destInfo.setText(destInfoStr + ":" + destTextLat.getText().toString() + "," + destTextLng.getText().toString());

}

} else {

Toast.makeText(getApplicationContext(), "获取位置信息出错", Toast.LENGTH\_LONG).show();

destInfoStr = "不明位置";

destInfo.setText(destInfoStr + ":" + destTextLat.getText().toString() + "," + destTextLng.getText().toString());

}

}

@Override

public void onGeocodeSearched(GeocodeResult geocodeResult, int i) {

}

class MyThread1 extends Thread {

private StringBuffer stringBuffer;

@Override

public void run() {

stringBuffer = EncryptedCaledData(latLngLeftTop.latitude, latLngLeftTop.longitude);

}

}

class MyThread2 extends Thread {

private StringBuffer stringBuffer;

@Override

public void run() {

stringBuffer = EncryptedCaledData(latLngRightTop.latitude, latLngRightTop.longitude);

}

}

class MyThread3 extends Thread {

private StringBuffer stringBuffer;

@Override

public void run() {

stringBuffer = EncryptedCaledData(latLngRightBottom.latitude, latLngRightBottom.longitude);

}

}

class MyThread4 extends Thread {

private StringBuffer stringBuffer;

@Override

public void run() {

stringBuffer = EncryptedCaledData(latLngLeftBottom.latitude, latLngLeftBottom.longitude);

}

}

}

package com.alvis.trace.activity;

import android.app.Activity;

import android.app.ActivityManager;

import android.app.ProgressDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.os.Handler;

import android.os.Looper;

import android.os.Message;

import android.os.StrictMode;

import android.preference.PreferenceManager;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.Window;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.TextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.localbroadcastmanager.content.LocalBroadcastManager;

import com.alvis.trace.R;

import com.alvis.trace.model.Client;

import com.alvis.trace.model.ClientConServerThread;

import com.alvis.trace.model.MessageType;

import com.alvis.trace.model.User1;

import com.alvis.trace.util.MapUtils;

import com.alvis.trace.util.MyActivityManager;

import com.amap.api.location.AMapLocationClient;

public class LoginActivity extends Activity {

public Button btnLogin;

private EditText accountEt, passwordEt;

private TextView appnameTv;

private RadioButton rbCustomer, rbDriver;

private CheckBox checkBox;

private SharedPreferences pref;

private SharedPreferences.Editor editor;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

requestWindowFeature(Window.FEATURE\_NO\_TITLE);//窗体无头部

setContentView(R.layout.activity\_login);

pref = PreferenceManager.getDefaultSharedPreferences(this);

// MyActivityManager.addr = pref.getString("ipaddr", "").equals("") ? "10.42.93.143" : pref.getString("ipaddr", "");

initView();

btnLogin.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

if (accountEt.getText().toString().trim().equals("") || passwordEt.getText().toString().trim().equals("")) {

Toast.makeText(LoginActivity.this, "账号或密码不能为空！", Toast.LENGTH\_SHORT).show();

} else if (!rbCustomer.isChecked() & !rbDriver.isChecked()) {

Toast.makeText(LoginActivity.this, "请选择身份！", Toast.LENGTH\_SHORT).show();

} else {

int roleid = rbCustomer.isChecked() ? 1 : 2;

login(accountEt.getText().toString(), passwordEt.getText().toString(), roleid);

}

}

});

appnameTv.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

showBuilder();

}

});

}

public void showBuilder() {

LayoutInflater inflater = LayoutInflater.from(this);

final View view = inflater.inflate(R.layout.addr\_input, null);

androidx.appcompat.app.AlertDialog.Builder builder = new androidx.appcompat.app.AlertDialog.Builder(this);

EditText editText1 = (EditText) view.findViewById(R.id.et1);

EditText editText2 = (EditText) view.findViewById(R.id.et2);

EditText editText3 = (EditText) view.findViewById(R.id.et3);

EditText editText4 = (EditText) view.findViewById(R.id.et4);

String[] ipfrags = pref.getString("ipaddr", "").split("\\.");

if (ipfrags.length == 4) {

editText1.setText(ipfrags[0]);

editText2.setText(ipfrags[1]);

editText3.setText(ipfrags[2]);

editText4.setText(ipfrags[3]);

}

builder.setTitle("Setting");

builder.setView(view);

final EditText et1 = editText1;

final EditText et2 = editText2;

final EditText et3 = editText3;

final EditText et4 = editText4;

builder.setPositiveButton("Confirm", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

//Toast.makeText(getApplicationContext(),"点击了确定",Toast.LENGTH\_LONG).show();

String addr = et1.getText().toString() + "."

+ et2.getText().toString() + "." + et3.getText().toString() + "." + et4.getText().toString();

MyActivityManager.addr = addr;

Toast.makeText(getApplicationContext(), addr, Toast.LENGTH\_LONG).show();

}

});

final androidx.appcompat.app.AlertDialog dialog = builder.show();

}

public void initView() {

MapUtils mapUtils = new MapUtils(getApplicationContext());

btnLogin = (Button) findViewById(R.id.btn\_login);

accountEt = (EditText) findViewById(R.id.et\_userName);

passwordEt = (EditText) findViewById(R.id.et\_password);

appnameTv = (TextView) findViewById(R.id.tv\_title);

rbCustomer = (RadioButton) findViewById(R.id.rb\_customer);

rbDriver = (RadioButton) findViewById(R.id.rb\_driver);

checkBox = (CheckBox) findViewById(R.id.cb\_checkbox);

boolean isRemember = pref.getBoolean("rp", false);

if (isRemember) {

accountEt.setText(pref.getString("account", ""));

passwordEt.setText(pref.getString("psd", ""));

RadioGroup radioGroup = (RadioGroup) findViewById(R.id.roleGroup);

if (pref.getInt("roleid", 1) == 1) {

radioGroup.check(rbCustomer.getId());

} else radioGroup.check(rbDriver.getId());

checkBox.setChecked(true);

}

}

/\*\*

\* 在此处处理登录请求

\*

\* @param account 用户名

\* @param psd 密码

\* @param roleid 角色身份

\*/

public void login(String account, String psd, int roleid) {

if (android.os.Build.VERSION.SDK\_INT > 9) {

StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();

StrictMode.setThreadPolicy(policy);

}

User1 user = new User1();

user.setUserName(account);

user.setPassword(psd);

user.setRoleId(roleid);

user.setMsgType(MessageType.LOGIN);

Log.e("account", account + "");

Log.e("psd", psd + "");

Log.e("roleid", roleid + "");

boolean isLoginSuc = new Client(this).sendLoginInfo(user);

// Log.e("isLoginSuc",isLoginSuc+"");

if (isLoginSuc) {

// if(true){

MyActivityManager.locationClient = new AMapLocationClient(this.getApplicationContext());

editor = pref.edit();

if (checkBox.isChecked()) {

editor.putBoolean("rp", true);

editor.putString("account", account);

editor.putString("psd", psd);

editor.putInt("roleid", roleid);

editor.putString("ipaddr", MyActivityManager.addr);

} else {

editor.clear();

}

editor.commit();

if (roleid == 1) startActivity(new Intent(this, LocationSourceActivity.class));

else startActivity(new Intent(this, DriverActivity.class));

} else {

Toast.makeText(this, "登录失败，请重新登录", Toast.LENGTH\_SHORT).show();

}

}

/\*\*

\* 添加menu

\*/

public boolean onCreateOptionsMenu(Menu menu) {

menu.add(Menu.NONE, 1, 0, "退出");

return true;

}

/\*

\* 添加menu事件

\* @see android.app.Activity#onOptionsItemSelected(android.view.MenuItem)

\*/

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case 1:

for (int i = 0; i < MyActivityManager.ActivityList.size(); i++) {

if (MyActivityManager.ActivityList.get(i) != this) {

MyActivityManager.ActivityList.get(i).finish();

}

}

this.finish();

System.exit(0);

break;

}

return true;

}

}

package com.alvis.trace.activity;

import android.animation.ValueAnimator;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.IntentFilter;

import android.graphics.Color;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.view.Window;

import android.view.WindowManager;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.localbroadcastmanager.content.LocalBroadcastManager;

import com.alvis.trace.common.CircleArea;

import com.alvis.trace.common.QU\_AGRQ\_C;

import com.alvis.trace.common.QU\_AGRQ\_P;

import com.alvis.trace.common.Vertex;

import com.alvis.trace.model.MapData;

import com.alvis.trace.model.TMessage;

import com.alvis.trace.util.CoordinateConversion;

import com.alvis.trace.util.MapUtils;

import com.alvis.trace.util.MyActivityManager;

import com.amap.api.location.AMapLocation;

import com.amap.api.location.AMapLocationClient;

import com.amap.api.location.AMapLocationClientOption;

import com.amap.api.location.AMapLocationListener;

import com.alvis.trace.R;

import com.amap.api.maps2d.AMap;

import com.amap.api.maps2d.CameraUpdateFactory;

import com.amap.api.maps2d.LocationSource;

import com.amap.api.maps2d.MapView;

import com.amap.api.maps2d.UiSettings;

import com.amap.api.maps2d.model.BitmapDescriptorFactory;

import com.amap.api.maps2d.model.Circle;

import com.amap.api.maps2d.model.CircleOptions;

import com.amap.api.maps2d.model.LatLng;

import com.amap.api.maps2d.model.Marker;

import com.amap.api.maps2d.model.MarkerOptions;

import com.amap.api.maps2d.model.MyLocationStyle;

import java.io.IOException;

import java.io.ObjectOutputStream;

public class MainActivity extends Activity implements LocationSource, AMapLocationListener {

private AMap aMap;

private MapView mapView;

private Circle circle;

private EditText edtX;

private EditText edtY;

private EditText edtR;

private TextView tv;

private LinearLayout llo;

private Context context;

double Lon, Lat;

public AMapLocationClient mLocationClient = null;

public OnLocationChangedListener mListener = null;

public AMapLocationClientOption mLocationOption = null;

public MyLocationStyle myLocationStyle = null;

public int radius;

public CoordinateConversion coor = new CoordinateConversion();

private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String message = intent.getStringExtra("message");

Log.d("receiver", "Got message: " + message);

if (message.equals("SocketFailed")) {

OnSocketFailed();

//Toast.makeText(getApplicationContext(), "hahaha", Toast.LENGTH\_LONG).show();

}

}

};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

requestWindowFeature(Window.FEATURE\_NO\_TITLE);//窗体无头部

setContentView(R.layout.abroadmap\_activity);

mapView = (MapView) findViewById(R.id.map);

mapView.onCreate(savedInstanceState);// 此方法必须重写

init();

//初始化定位

mLocationClient = new AMapLocationClient(getApplicationContext());

//设置定位回调监听

mLocationOption = new AMapLocationClientOption();

mLocationOption.setLocationMode(AMapLocationClientOption.AMapLocationMode.Device\_Sensors);

mLocationOption.setOnceLocation(true);

mLocationClient.setLocationListener(this);

mLocationClient.setLocationOption(mLocationOption);

mLocationClient.startLocation();

myLocationStyle = new MyLocationStyle();

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_LOCATE);

aMap.setMyLocationStyle(myLocationStyle);

edtX = (EditText) findViewById(R.id.x);

edtY = (EditText) findViewById(R.id.y);

edtR = (EditText) findViewById(R.id.r);

tv = (TextView) findViewById(R.id.s);

llo = (LinearLayout) findViewById(R.id.ll);

LocalBroadcastManager.getInstance(this).registerReceiver(mMessageReceiver,

new IntentFilter("notify-to-refresh"));

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_BLUE));

otMarkerOptions.position(new LatLng(Lat, Lon));

aMap.addMarker(otMarkerOptions);

if (edtR.getText().toString().equals("")) {

radius = 1000;

edtR.setText("1000");

} else radius = Integer.parseInt(edtR.getText().toString());

circle = aMap.addCircle(new CircleOptions().center(new LatLng(Lat, Lon)).radius(radius)

.strokeColor(Color.argb(50, 1, 1, 1)).fillColor(Color.argb(50, 1, 1, 1)).strokeWidth(1));

AMap.OnMarkerDragListener markerDragListener = new AMap.OnMarkerDragListener() {

@Override

public void onMarkerDragStart(Marker marker) {

}

@Override

public void onMarkerDrag(Marker marker) {

edtX.setText(marker.getPosition().latitude + "");

edtY.setText(marker.getPosition().longitude + "");

}

@Override

public void onMarkerDragEnd(Marker marker) {

}

};

aMap.setOnMarkerDragListener(markerDragListener);

}

/\*\*

\* 初始化AMap对象

\*/

private void init() {

if (aMap == null) {

aMap = mapView.getMap();

aMap.setMapLanguage("en");

//setUpMap();

aMap.setLocationSource(this);

UiSettings settings = aMap.getUiSettings();

settings.setMyLocationButtonEnabled(true);

aMap.setMyLocationEnabled(true);

}

}

/\*\*

\* 对地图添加onMapIsAbroadListener

\*/

private void setUpMap() {

aMap.clear();

MarkerOptions otMarkerOptions = new MarkerOptions();

otMarkerOptions.icon(BitmapDescriptorFactory

.defaultMarker(BitmapDescriptorFactory.HUE\_BLUE));

otMarkerOptions.position(new LatLng(Lat, Lon));

Toast.makeText(getApplicationContext(), Lat + " " + Lon, Toast.LENGTH\_LONG).show();

aMap.addMarker(otMarkerOptions);

aMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(Lat, Lon), 15));

circle = aMap.addCircle(new CircleOptions().center(new LatLng(Lat, Lon)).radius(1000)

.strokeColor(Color.argb(50, 1, 1, 1)).fillColor(Color.argb(50, 1, 1, 1)).strokeWidth(1));

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onResume() {

super.onResume();

mapView.onResume();

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onPause() {

super.onPause();

mapView.onPause();

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

mapView.onSaveInstanceState(outState);

}

/\*\*

\* 方法必须重写

\*/

@Override

protected void onDestroy() {

super.onDestroy();

mapView.onDestroy();

LocalBroadcastManager.getInstance(this).unregisterReceiver(mMessageReceiver);

}

/\*\*

\* 发送外接正方形的四个点的数据

\*/

public void sendCaledSquareData() {

Thread th = new Thread() {

@Override

public void run() {

super.run();

//这里根据读取的半径生成四个点的坐标，然后与加密的地图数据分别进行84次计算然后封装到Tmessage里面

//最后的反馈信息在ClientConServer中处理

}

};

th.start();

}

public void sendEncryptedMapData(double latitude, double longitude) {

ObjectOutputStream oos = null;

try {

oos = new ObjectOutputStream(MyActivityManager.curClientThread.getSocket().getOutputStream());

TMessage tMessage = new TMessage();

tMessage.setMsgType(1);

MapData mapData = new MapData();

mapData.setEncryptedMap(EncryptedCaledData(Lat, Lon));

mapData.setDataType(MapData.ECPM);

tMessage.setMapData(mapData);

oos.writeObject(tMessage);

Log.e("sendEncryptedMapData: ", "写入数据完成！");

} catch (IOException e) {

Log.e("ccst init: ", "读取输出流失败！");

}

}

/\*\*

\* 将加密的地图数据和某个位置进行计算

\*

\* @return TMessage类型的数据

\*/

public StringBuffer EncryptedCaledData(double latitude, double longitude) {

CoordinateConversion coor = new CoordinateConversion();

String coorResult = coor.latLon2UTM(latitude, longitude);

String[] coorResults = coorResult.split(" ");

Vertex vertex = new Vertex(Long.parseLong(coorResults[2]), Long.parseLong(coorResults[3]));

//这个Vertex是自身的位置，然后需要和84个加密的地图数据进行计算，得到一个mapdata的数据然后用Tmessage封装

String[] mapDatas = MapUtils.mapDatas;

MapData mapData = new MapData();

StringBuffer stringBuffer = new StringBuffer("");

QU\_AGRQ\_P qu\_agrq\_p = new QU\_AGRQ\_P();

for (String str : mapDatas) {

try {

stringBuffer.append(qu\_agrq\_p.UF\_AGRQ\_P\_RDC(str, vertex) + ";");

} catch (Exception e) {

Log.e("EncryptedCaledData: ", "数据处理出错！");

e.printStackTrace();

}

}

return stringBuffer;

}

private void dimBackground(final float from, final float to) {

final Window window = getWindow();

ValueAnimator valueAnimator = ValueAnimator.ofFloat(from, to);

valueAnimator.setDuration(100);

valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {

@Override

public void onAnimationUpdate(ValueAnimator animation) {

WindowManager.LayoutParams params = window.getAttributes();

params.alpha = (Float) animation.getAnimatedValue();

window.setAttributes(params);

}

});

valueAnimator.start();

}

/\*\*

\* 处理Socket连接失败的问题

\*/

public void OnSocketFailed() {

Dialog dialog = new AlertDialog.Builder(MainActivity.this).setTitle("操作结果")

.setItems(new String[]{"服务异常，请退出后重新登录 "}, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

}

}).setPositiveButton("确定", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

}

}).create();

dialog.show();

}

@Override

public void onLocationChanged(AMapLocation aMapLocation) {

if (aMapLocation != null) {

if (aMapLocation.getErrorCode() == 0) {

//设置缩放级别

aMap.moveCamera(CameraUpdateFactory.zoomTo(17));

//将地图移动到定位点

aMap.moveCamera(CameraUpdateFactory.changeLatLng(new LatLng(aMapLocation.getLatitude(), aMapLocation.getLongitude())));

//点击定位按钮 能够将地图的中心移动到定位点

mListener.onLocationChanged(aMapLocation);

Lat = aMapLocation.getLatitude();

Lon = aMapLocation.getLongitude();

edtX.setText(Lat + "");

edtY.setText(Lon + "");

Toast.makeText(getApplicationContext(), aMapLocation.getLatitude() + ":" + aMapLocation.getLongitude(), Toast.LENGTH\_LONG).show();

} else {

Toast.makeText(getApplicationContext(), "定位失败", Toast.LENGTH\_LONG).show();

Log.e("AmapError", "location Error, ErrCode:"

+ aMapLocation.getErrorCode() + ", errInfo:"

+ aMapLocation.getErrorInfo());

}

}

}

@Override

public void activate(OnLocationChangedListener onLocationChangedListener) {

mListener = onLocationChangedListener;

}

@Override

public void deactivate() {

}

}

package com.ping.thingsjournalclient.server;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Toolkit;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import javax.swing.JButton;

import javax.swing.JFrame;

import com.alvis.trace.gui.ProgressFrame;

import com.alvis.trace.gui.TraceServer;

import com.alvis.trace.model.MessageType;

import com.alvis.trace.model.TMessage;

import com.alvis.trace.model.User1;

import com.ping.thingsjournalclient.dao.UserDao1;

public class Server1 extends JFrame {

private ServerSocket ss;

private Socket socket;

private static int SERVER\_PORT = 6100;

public static TraceServer serverGui;

public Server1() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getContentPane().setBackground(Color.LIGHT\_GRAY);

setForeground(Color.LIGHT\_GRAY);

Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();

int width = dimension.width;

int height = dimension.height;

setResizable(false);

getContentPane().setFont(new Font("微软雅黑", Font.PLAIN, 12));

setVisible(true);

setTitle("安心出行服务端");

setSize(325, 300);

setLocation((width - 325) / 2, (height - 300) / 2);

getContentPane().setLayout(null);

JButton btnLaunch\_1 = new JButton("启动");

btnLaunch\_1.setFont(new Font("Microsoft JhengHei UI", Font.BOLD, 14));

btnLaunch\_1.setBackground(new Color(31, 186, 214));

btnLaunch\_1.setBounds(105, 112, 93, 23);

getContentPane().add(btnLaunch\_1);

JButton btnQuit\_1 = new JButton("退出");

btnQuit\_1.setFont(new Font("Microsoft JhengHei UI", Font.BOLD, 14));

btnQuit\_1.setBackground(new Color(31, 186, 214));

btnQuit\_1.setForeground(Color.BLACK);

btnQuit\_1.setBounds(105, 157, 93, 23);

getContentPane().add(btnQuit\_1);

btnQuit\_1.addMouseListener(new MouseListener() {

@Override

public void mouseClicked(MouseEvent e){

// TODO Auto-generated method stub

System.exit(0);

}

@Override

public void mousePressed(MouseEvent e){

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e){

// TODO Auto-generated method stub

}

@Override

public void mouseEntered(MouseEvent e){

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e){

// TODO Auto-generated method stub

}

});

btnLaunch\_1.addMouseListener(new MouseListener() {

@Override

public void mouseClicked(MouseEvent e){

dispose();

ProgressFrame progress = new ProgressFrame();

progress.setVisible(true);

new Thread(() -> {

serverGui = new TraceServer(progress);

}).start();

new Thread(() -> {

try {

ss = new ServerSocket(SERVER\_PORT);

System.out.println("加密的地图数据：\n" + StatusList.singleMapData + StatusList.singleMapData1);

while (serverGui == null) {

try {

Thread.sleep(1000);

} catch (InterruptedException e1) {

e1.printStackTrace();

}

}

serverGui.appendLog(

new StringBuffer("加密的地图数据：\n" + StatusList.singleMapData + StatusList.singleMapData1));

while (true) {

socket = ss.accept();

socket.setSoTimeout(1000 \* 60 \* 20);

new Thread(new DataProcessThread(socket)).start();

}

} catch (IOException e1) {

e1.printStackTrace();

}

}).start();

}

@Override

public void mousePressed(MouseEvent e){

}

@Override

public void mouseReleased(MouseEvent e){

}

@Override

public void mouseEntered(MouseEvent e){

}

@Override

public void mouseExited(MouseEvent e){

}

});

}

public static void main(String[] args){

new Server1();

}

class DataProcessThread implements Runnable {

private Socket socket;

private ObjectInputStream ois;

private ObjectOutputStream oos;

private User1 user;

private UserDao1 userDao;

public DataProcessThread(Socket socket) {

super();

this.socket = socket;

}

/\*\*

\* 预处理

\*/

public void BlackSheepWall(){

try {

ois = new ObjectInputStream(this.socket.getInputStream());

oos = new ObjectOutputStream(this.socket.getOutputStream());

user = (User1) ois.readObject();

userDao = new UserDao1();

} catch (ClassNotFoundException e) {

System.out.println("输入流找不到对象！");

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@Override

public void run(){

this.BlackSheepWall();

// while(true) {//这个while究竟有没有必要呢？应该没有

if (user.getMsgType() == MessageType.LOGIN) {

/\*

System.out.println(user.getUserName());

System.out.println(user.getPassword());

System.out.println(user.getRoleId());

\*/

User1 userlog = userDao.UserLog(user.getUserName(), user.getPassword(), user.getRoleId());

TMessage tmsg = new TMessage();

if (userlog != null) {// 登录成功, userlog为成功登录后的用户

System.out.println("用户：" + user.getUserName() + "成功登录！");

serverGui.appendLog(new StringBuffer("用户：" + user.getUserName() + "成功登录！"));

ServerConClientThread scct = new ServerConClientThread(this.socket, userlog);

StatusList.put(userlog.getID(), scct);

scct.start();

tmsg.setMsgType(MessageType.LOGIN\_SUC);

tmsg.setToUser(userlog);

} else {// 登录失败

tmsg.setMsgType(MessageType.LOGIN\_FAIL);

}

try {

oos.writeObject(tmsg);

} catch (IOException e) {

System.out.println("登录反馈消息写入失败！");

e.printStackTrace();

}

} else {// 如果消息类型为别的,暂时没别的，主动抛出异常

System.out.println(user.getUserName());

System.out.println(user.getPassword());

System.out.println(user.getRoleId());

System.out.println(user.getMsgType());

throw new RuntimeException("未知消息内容！");

}

// }

}

}

}

package com.ping.thingsjournalclient.server;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.util.ArrayList;

import com.alvis.trace.model.MapData;

import com.alvis.trace.model.MessageType;

import com.alvis.trace.model.TMessage;

import com.alvis.trace.model.User1;

import com.ping.thingsjournalclient.util.DbUtil;

import common.QU\_AGRQ\_P;

public class ServerConClientThread extends Thread {

private Socket socket;

private User1 user;

private DbUtil dbUtil = new DbUtil();

private Connection conn = dbUtil.getCon();

private QU\_AGRQ\_P p\_func = new QU\_AGRQ\_P();

private final int[][] a = { { 21, 22, 25, 26, 37, 38, 41, 42 }, { 23, 24, 27, 28, 39, 40, 43, 44 },

{ 29, 30, 33, 34, 45, 46, 49, 50 }, { 31, 32, 35, 36, 47, 48, 51, 52 }, { 53, 54, 57, 58, 69, 70, 73, 74 },

{ 55, 56, 59, 60, 71, 72, 75, 76 }, { 61, 62, 65, 66, 77, 78, 81, 82 },

{ 63, 64, 67, 68, 79, 80, 83, 84 } };

public ServerConClientThread(Socket socket, User1 user) {

//System.out.println("进入线程了！");

this.socket = socket;

this.user = user;

}

public Socket getSocket() {

return socket;

}

public void setSocket(Socket socket) {

this.socket = socket;

}

public User1 getUser() {

return user;

}

public void setUser(User1 user) {

this.user = user;

}

private void insertLogSQL(String locData, int uid, String operation, String result) {

String sql = " insert into t\_position (loc, user\_id, operation, result, timestamp) values (?, ?, ?, ?, ?)";

PreparedStatement pstmt = null;

try {

conn = new DbUtil().getCon();

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, locData);

pstmt.setInt(2, uid);

pstmt.setString(3, operation);

pstmt.setString(4, result);

pstmt.setLong(5, System.currentTimeMillis());

pstmt.executeUpdate();

conn.close();

} catch (Exception e) {

System.out.println("插入数据失败");

e.printStackTrace();;

}

}

/\*\*

\* 对收到的包含位置计算的地图数据进行处理并找到所在块

\*

\* @param mapData 这里在客户端要做一次处理使得按照顺序传送

\* @return 返回的是一个包含具体块号的ArrayList

\*/

public ArrayList<Integer> findPos(MapData mapData) {

StringBuffer strb = mapData.getEncryptedMap();

ArrayList<Integer> pos = new ArrayList<Integer>();

String[] ems = strb.toString().split(";");

System.out.print("ems:");

System.out.println(ems[0]);

Server1.serverGui.appendLog(new StringBuffer(ems[0]));

for (int i = 1; i <= ems.length; i++) {

try {

if (p\_func.QU\_AGRQ\_P\_QRR(ems[i - 1])) {

pos.add(i % 84 == 0 ? 84 : i % 84);// 数据库中的地图序号按照84来的，

}

} catch (Exception e) {

System.out.println("地图数据错误！！！");

e.printStackTrace();

}

}

if(pos.size()!=1) System.out.println("地图数据错误！！！");

System.out.println("数组长度：" + ems.length);

System.out.println("-------------------");

String status = null;

boolean inMap = false;

for (int j = 0; j < pos.size(); j++) {

if (pos.get(j) < 21) {

pos.remove(j);

j--;

} else {

inMap = true;

}

}

if (pos.size() == 1) {

System.out.println("用户：“" + user.getUserName() + "” 所在地图块号:" + pos.get(0));

String loc = "";

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

if (a[i][j] == pos.get(0)) {

loc = j + "" + i;

}

}

}

Server1.serverGui.userLocation.put(user, loc);

Server1.serverGui.appendLog(new StringBuffer("用户：“" + user.getUserName() + "” 所在地图块号:" + pos.get(0)));

} else

for (int j = 0; j < pos.size(); j++) {

System.out.println("外接正方形顶点所在地图块号：" + pos.get(j));

Server1.serverGui.appendLog(new StringBuffer("外接正方形顶点所在地图块号：" + pos.get(j)));

}

if (!inMap) {

System.out.println("用户：“" + user.getUserName() + "” 当前不在地图中。");

Server1.serverGui.appendLog(new StringBuffer("用户：“" + user.getUserName() + "” 当前不在地图中。"));

}

solve(pos);

String loc\_data = strb.toString();

System.out.println("uid:" + user.getID() + "loc\_data:" + loc\_data + "time:" + System.currentTimeMillis() + "\n");

System.out.println(loc\_data.length());

System.out.println("-------------------");

// System.out.println("块号数量:" + pos.size());

if(pos.size()==0) pos.add(-1);

return pos;

}