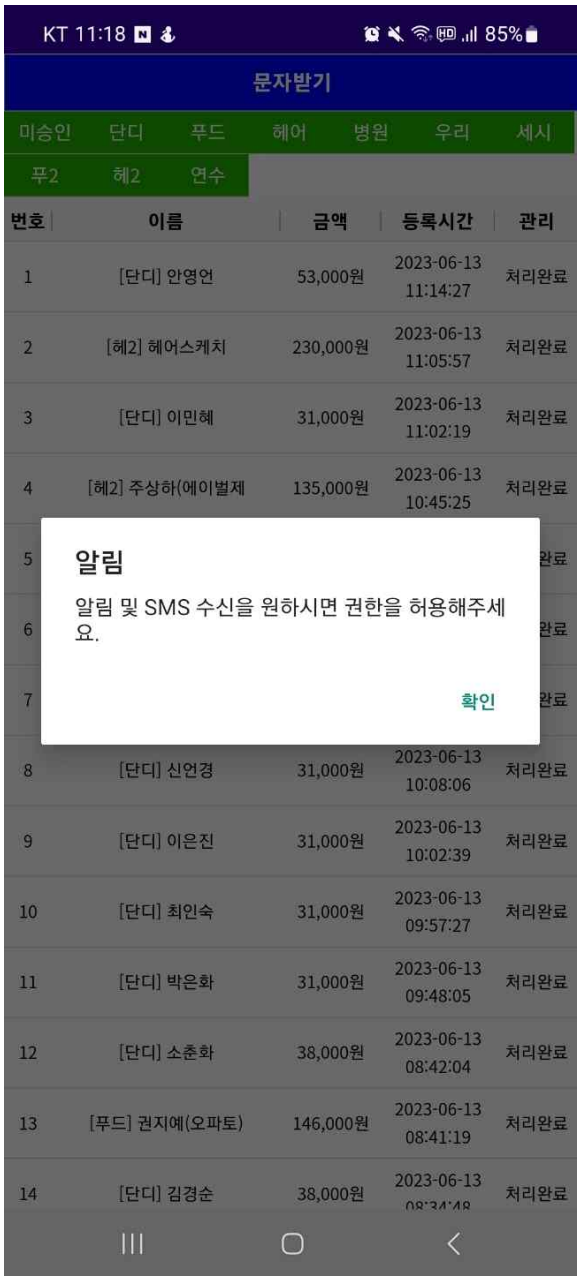
1. 우리매니저 결제관리 앱 최초 설치시 권한을 요청하기전 권한에 대한 설명을 팝업창에 띄움



해당 부분 코드 (테드 퍼미션 라이브러리 사용)

- setRationalTitle(), setRationalMessage() 메소드를 사용하여 팝업창의 제목과 내용을 설정함

```
TedPermission.create().setPermissionListener(new PermissionListener() {
    @Override
    public void onPermissionGranted() {
        sendRegisterDataToServer();
    }

    @Override
    public void onPermissionDenied(List<String> deniedPermissions) {
        sendRegisterDataToServer();
        String[] permissionsList = deniedPermissions.toArray(new String[0]);
        ActivityCompat.requestPermissions( activity: MainActivity.this, permissionsList, requestCode: 1);
    }
})      .setPermissions(POST_NOTIFICATIONS, READ_PHONE_STATE, READ_SMS, RECEIVE_SMS)
        .setRationaleTitle("알림").setRationaleMessage("알림 및 SMS 수신을 원하시면 권한을 허용해주세요.")
        .setDeniedTitle("알림").setDeniedMessage("일부 권한이 거부되어 있습니다.\n[설정] > [권한]에서 권한을 허용해주세요.")
        .check();
```

```
.setRationaleTitle("알림").setRationaleMessage("알림 및 SMS 수신을 원하시면 권한을 허용해주세요.")
```

1-1 앱 최초 설치시 파이어베이스 토큰이 생성되며 서버로 토큰데이터를 보낸다

'FirebaseMessagingService'를 상속받는 'MessagingService' 클래스는 onNewToken() 콜백 함수를 오버라이드해서 앱을 최초 설치해서 새 토큰이 생성되면 sendRegistrationToServer() 메소드를 호출해서 서버로 토큰 데이터를 보낸다

```
public class MessagingService extends FirebaseMessagingService {
    private final ServerManager serverManager = new ServerManager();
    private int NOTIFICATION_ID = 0, REQUEST_CODE = 0;

    /**
     * There are two scenarios when onNewToken is called:
     * 1) When a new token is generated on initial app startup
     * 2) Whenever an existing token is changed
     * Under #2, there are three scenarios when the existing token is
     * A) App is restored to a new device
     * B) User uninstalls/reinstalls the app
     * C) User clears app data
     */
    @Override
    public void onNewToken(@NonNull String token) { // 새 토큰이 생성될
        super.onNewToken(token);
        Log.d( tag: "FCM_NewToken", msg: "Refreshed token : " + token);

        /**
         * If you want to send messages to this application instance
         * manage this apps subscriptions on the server side, send th
         * FCM registration token to your app server.
         */
        sendRegistrationToServer(token);
    }

    private void sendRegistrationToServer(String token) {
        serverManager.registerDataToServer(token);
    }
```

서버로 보내진 데이터

- 아래의 'reg_id'는 토큰에 해당하는 변수고 'device_id'는 기기ID에 해당하는 변수인데 안드로이드 API33에서는 해당사항이 권장되지않아 생략되었다 'phone'는 사용자 전화번호에 해당하는 변수이다

```
reg_id : ensxNMKLQbuXJOIrwZo2Fw:APA91bGWmOaIrMesOmzQe0mx5LMu9oSExnFpF60r45EMX-Zv_S1cr5PTBM5OtI-
dU_xmyJdfE9F_8P_VBX5mZHNPpTcet_N5lYo0Pi4Ob2S_VhhhglJyDPANQAlNsTVt_YzBiJInqWhH || device_id : || phone :
```

2. 팝업창에서 확인을 눌렀을 때 권한을 요청 (FCM 푸쉬 알림, 전화번호, 문자)

해당 부분 코드 (테드 퍼미션 라이브러리 사용)

- setPermissions() 메소드를 사용하여 사용자에게 요청할 권한을 매개변수로 넘겨줌

```
TedPermission.create().setPermissionListener(new PermissionListener() {
    @Override
    public void onPermissionGranted() {
        sendRegisterDataToServer();
    }

    @Override
    public void onPermissionDenied(List<String> deniedPermissions) {
        sendRegisterDataToServer();
        String[] permissionsList = deniedPermissions.toArray(new String[0]);
        ActivityCompat.requestPermissions( activity: MainActivity.this, permissionsList,  requestCode: 1);
    }
})      .setPermissions(POST_NOTIFICATIONS, READ_PHONE_STATE, READ_SMS, RECEIVE_SMS)
        .setRationaleTitle("알림").setRationaleMessage("알림 및 SMS 수신을 원하시면 권한을 허용해주세요.")
        .setDeniedTitle("알림").setDeniedMessage("일부 권한이 거부되어 있습니다.\n[설정] > [권한]에서 권한을 허용해주세요.")
        .check();
```

```
.setPermissions(POST_NOTIFICATIONS, READ_PHONE_STATE, READ_SMS, RECEIVE_SMS)
```

2-1 요청 권한을 모두 승인한 경우
테드퍼미션의 onPermissionGranted() 메소드가 호출되며 내부적으로 sendRegisterDataToServer() 메소드를 호출한다

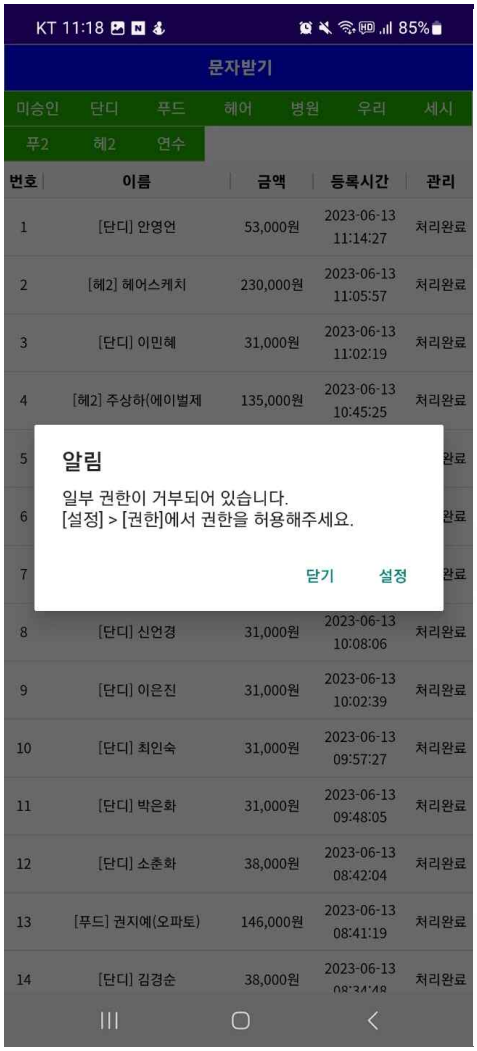
```
public void sendRegisterDataToServer() {
    FirebaseMessaging.getInstance().getToken().addOnCompleteListener(task -> {
        try {
            serverManager.registerDataToServer(task.getResult(), getPhoneNumber());
        } catch (IllegalArgumentException e) {
            Log.e( tag: "Exception", e.getMessage());
        }
    });
}
```

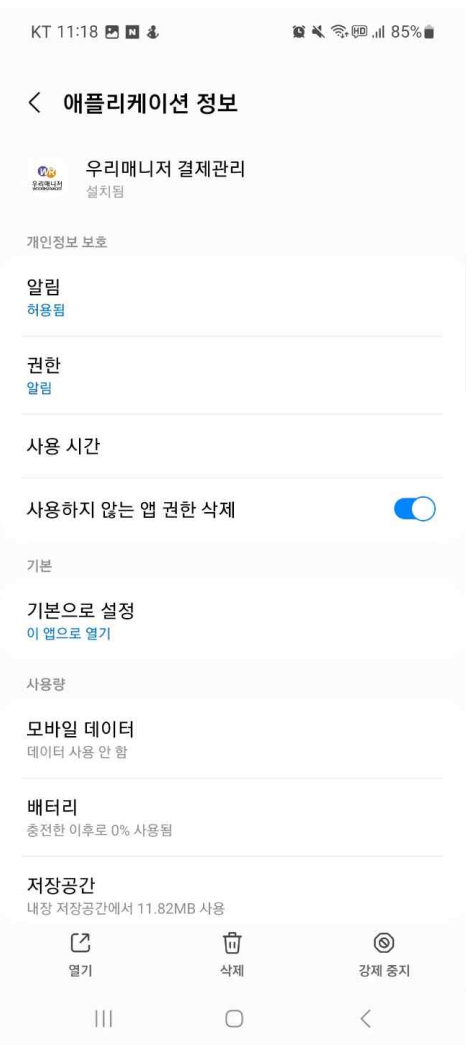
파이어베이스에서 토큰값을 가져오는 작업은 비동기 작업이기 때문에 addOnCompleteListener() 콜백 함수를 연결해서 task 작업이 완료되면 'ServerManager'클래스의 registerDataToServer() 메소드를 호출해서 서버에 토큰과 전화번호 데이터를 서버로 보낸다

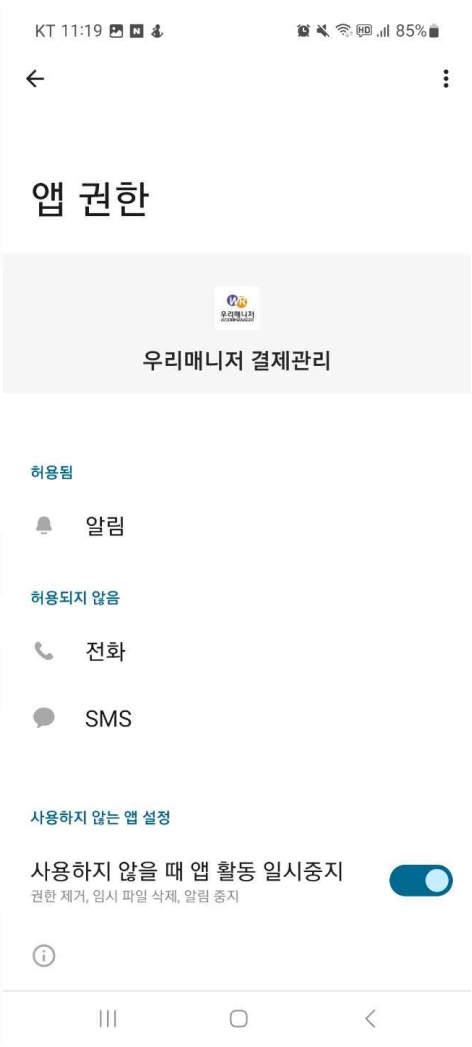
서버로 보내진 데이터
- 아래의 'reg_id'는 토큰에 해당하는 변수고 'device_id'는 기기ID에 해당하는 변수인데 안드로이드 API33에서는 해당사항이 권장되지않아 생략되었다 'phone'는 사용자 전화번호에 해당하는 변수이다

reg_id : dfnphubySW2dwkQuvB2riS:APA91bGhT8uluvKIZKZm3LyMrKQlvwj18lNRwpNhGLOreDfBJsjlB0wm7sCT2pSovwHBkPa7SBPU9IzpsiNkQoz7BleTjca_FNzt-QaReE9wt5RbuSSZDL_ExlV9Oj5bePrj9O5e3pz6 || device_id : || phone : 821022052119	821022052119

2-2 요청한 권한중에서 일부 권한을 허용하지 않았을 경우
'설정' 버튼을 누르면 앱 권한을 설정하는 화면으로 넘어간다







해당 부분 코드 (테드 퍼미션 라이브러리 사용)

- setDeniedTtile(), setDeniedMessage() 메소드를 사용하여 사용자가 권한을 거절했을 때 팝업창의 제목과 내용을 설정함
- onPermissionDenied() 메소드의 매개변수인 List<String> deniedPermissions 변수에 거절된 권한들에 대한 데이터가 저장되있고 String 배열로 바꾼뒤 requestPermissions() 메소드의 매개변수로 넘겨줌으로서 거절된 권한은 다시 한번 권한을 재요청하게됨

```
TedPermission.create().setPermissionListener(new PermissionListener() {
    @Override
    public void onPermissionGranted() {
        sendRegisterDataToServer();
    }

    @Override
    public void onPermissionDenied(List<String> deniedPermissions) {
        sendRegisterDataToServer();
        String[] permissionsList = deniedPermissions.toArray(new String[0]);
        ActivityCompat.requestPermissions( activity: MainActivity.this, permissionsList, requestCode: 1);
    }
})      .setPermissions(POST_NOTIFICATIONS, READ_PHONE_STATE, READ_SMS, RECEIVE_SMS)
        .setRationaleTitle("알림").setRationaleMessage("알림 및 SMS 수신을 원하시면 권한을 허용해주세요.")
        .setDeniedTitle("알림").setDeniedMessage("일부 권한이 거부되어 있습니다.\n[설정] > [권한]에서 권한을 허용해주세요.")
        .check();
```

3. 웹뷰(WebView) 화면

번호	이름	금액	등록시간	관리
1	[헤2] 이강민	230,000원	2023-06-13 11:18:05	처리완료
2	[헤2] 타마시헤어수원성	85,000원	2023-06-13 11:17:02	승인필요
3	[단디] 안영언	53,000원	2023-06-13 11:14:27	처리완료
4	[헤2] 헤어스케치	230,000원	2023-06-13 11:05:57	처리완료
5	[단디] 이민혜	31,000원	2023-06-13 11:02:19	처리완료
6	[헤2] 주상하(에이벌제	135,000원	2023-06-13 10:45:25	처리완료
7	[단디] 변기연	38,000원	2023-06-13 10:44:49	처리완료
8	[헤2] 송호준	127,700원	2023-06-13 10:42:04	처리완료
9	[헤2] 신강철	85,000원	2023-06-13 10:30:38	처리완료
10	[단디] 신언경	31,000원	2023-06-13 10:08:06	처리완료
11	[단디] 이은진	31,000원	2023-06-13 10:02:39	처리완료
12	[단디] 최인숙	31,000원	2023-06-13 09:57:27	처리완료
13	[단디] 박은화	31,000원	2023-06-13 09:48:05	처리완료
14	[단디] 소춘화	38,000원	2023-06-13 08:42:04	처리완료

메인액티비티 코드중 일부

- 이너클래스인 'JavaScriptService'를 웹뷰의 addJavascriptInterface() 메소드를 사용해서 연결함으로서 웹뷰 상단의 문자받기 파란색버튼을 누르게 될 경우 'JavaScriptService'클래스의 smsUpdate() 메소드가 실행됨 smsUpdate() 메소드는 내부적으로 smsUpdateCheck() 메소드를 호출하도록 구현
- 결과적으로 웹뷰의 문자받기 버튼을 누르면 안드로이드의 smsUpdateCheck() 메소드가 호출됨

```
webView = binding.webView;
webView.getSettings().setJavaScriptEnabled(true);
webView.addJavascriptInterface(new JavaScriptService(),  name: "android");
```

```
public class JavaScriptService {
    @JavascriptInterface
    public void smsUpdate() {
        smsUpdateCheck();
    }
}
```

getPhoneNumber()

- 제어문 if를 통해 checkSelfPermission() 메소드를 통해 권한을 체크하고 권한이 승인되었을 경우 'TelephonyManager' 객체의 getLine1Number() 메소드를 통해 전화번호 데이터를 가져온 뒤 자바 String 클래스의 replace() 메소드를 통해 데이터를 재가공 한뒤 전화번호가 '82'로 시작하지않고 '0'으로 시작한다면 다시한번 마무리 정제과정을 거쳐 전화번호 값을 반환하고 권한이 승인되지 않았을 경우 null을 반환한다

```
public String getPhoneNumber() {
    if (ActivityCompat.checkSelfPermission( context: this, READ_PHONE_STATE) == PERMISSION_GRANTED &&
        ActivityCompat.checkSelfPermission( context: this, READ_SMS) == PERMISSION_GRANTED) {
        TelephonyManager tm = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
        String phoneNumber = tm.getLine1Number().replace( target: "+", replacement: "");
        return !phoneNumber.startsWith("82") && phoneNumber.startsWith("0") ? "82" + phoneNumber.substring(1) : phoneNumber;
    }
    return null;
}
```

smsUpdateCheck()

- getPhoneNumber() 메소드의 반환값에 따라 권한 여부를 파악하고 권한이 없다면 그대로 return 함으로서 해당 부분 아래의 로직을 실행하지않고 메소드를 끝마치고 권한이 있다면 'Cursor'객체와 해당기기의 SMS(문자메시지)에 접근할 수 있는 uri 정보가 있는 'Uri'객체를 이용해서 최신날짜순으로 30개의 문자메시지를 while문을 통해서 데이터를 추출하고 StringBuilder 타입의 'messageBody' 변수에 append() 메소드와 String.format() 메소드를 통해 서버에 데이터를 보내기전 알맞은 형식의 데이터로 가공한 뒤 'ServerManager' 클래스의 messageToServer() 메소드를 호출해서 서버로 문자내용을 보낸다

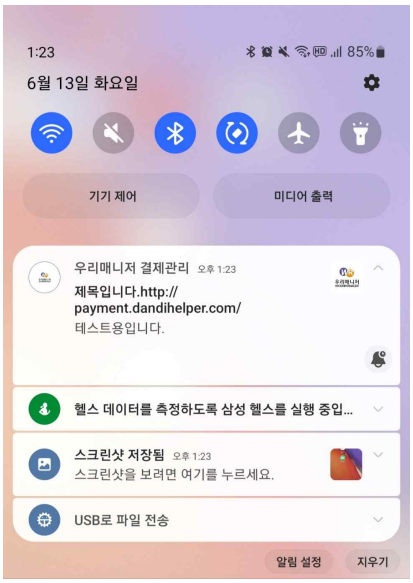
```
public void smsUpdateCheck() {
    if (getPhoneNumber() == null) {
        Toast.makeText( context: this, text: "전화번호 정보가 없습니다.\n전화 및 SMS 권한을 허용해주세요.", Toast.LENGTH_SHORT).show();
        return;
    }
    boolean checkFirstMessage = true;
    StringBuilder messageBody = new StringBuilder();
    Cursor cursor = getContentResolver().query(Uri.parse("content://sms"),
                    new String[]{"_id", "thread_id", "address", "person", "date", "body"}, selection: null, selectionArgs: null, sortOrder: "date DESC limit 0, 30");

    while (cursor.moveToNext()) {
        long messageId = cursor.getLong( columnIndex: 0); // "_id"
        String address = cursor.getString( columnIndex: 2); // "address"
        String body = cursor.getString( columnIndex: 5); // "body"
        messageBody.append(String.format("%s@@%s", address, body)).append("||");

        if (checkFirstMessage) {
            firstMessageId = messageId;
            checkFirstMessage = false;
        }
    } cursor.close();

    if (compareMessageId != firstMessageId) {
        compareMessageId = firstMessageId;
        try {
            serverManager.messageToServer(messageBody.toString(), getPhoneNumber());
        } catch (Exception e) {
            Log.e( tag: "Exception", e.getMessage());
        }
    }
}
```

4. Firebase Cloud Messaging 푸쉬 알림



FCM에서는 애플리케이션이 서버에서 Push 메시지를 받을 때, 이를 Broadcast 메시지로 전환하여 시스템에 발송

```
@Override // FCM에서는 애플리케이션이 서버에서 Push 알림 메시지를 받을 때, 이를 Broadcast 메시지로 전환하여 시스템에 발송
public void onMessageReceived(@NonNull RemoteMessage message) {
    super.onMessageReceived(message);
    if (NotificationManagerCompat.from(this).areNotificationsEnabled()) {
        NOTIFICATION_ID++; REQUEST_CODE++;
        String CHANNEL_ID = "WooriManager", CHANNEL_NAME = "WooriPayment";
        String title = !message.getData().isEmpty() ? message.getData().get("Notice") : message.getNotification().getTitle();
        String body = !message.getData().isEmpty() ? message.getData().get("Text") : message.getNotification().getBody();
        String url = !message.getData().isEmpty() ? message.getData().get("Link") : "http://payment.dandihelper.com/";

        Intent intent = new Intent( packageContext: this, MainActivity.class).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP | Intent.FLAG_ACTIVITY_CLEAR_TOP).putExtra( name
        PendingIntent pendingIntent = PendingIntent.getActivity( context: this, REQUEST_CODE, intent, flags: PendingIntent.FLAG_IMMUTABLE | PendingIntent.FLAG_ONE_SHOT

        /*안드로이드 오레오(API 26) 이상부터는 모든 알림에 채널을 할당해야한다 채널을 할당하지 않으면 알림이 오지 않는다
        NotificationManager의 createNotificationChannel()를 이용해서 등록, 알림이 올 때마다 만들 필요가 없고 딱 한번만 만들면 된다
        채널은 알림마다 설정해서 채널을 통해 알림을 분류하고 채널별로 설정을 다르게 지정할 수 있다*/
        NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
        NotificationCompat.Builder builder = null;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            if (notificationManager.getNotificationChannel(CHANNEL_ID) == null) {
                NotificationChannel channel = new NotificationChannel(CHANNEL_ID, CHANNEL_NAME, NotificationManager.IMPORTANCE_HIGH);
                notificationManager.createNotificationChannel(channel);
            }
            builder = new NotificationCompat.Builder( context: this, CHANNEL_ID);
        } else builder = new NotificationCompat.Builder( context: this);

        builder.setSmallIcon(R.drawable.woorimanager_logo_icon)
                .setLargeIcon(BitmapFactory.decodeResource(getResources(), R.drawable.woorimanager_logo_icon))
                .setContentTitle(title)
                .setContentText(body)
                .setStyle(new NotificationCompat.BigTextStyle().bigText(body))
                .setContentIntent(pendingIntent)
                .setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION))
                .setAutoCancel(true);

        Notification notification = builder.build();
        notificationManager.notify(NOTIFICATION_ID, notification);
    }
}
```

푸쉬 알림이 올 경우 onMessageReceived() 콜백 함수가 호출되며 권한이 승인 되 있을 경우 아래의 로직을 실행하며 안드로이드 API 26 이상부터는 모든 알림에 채널을 할당해야만 알림을 수신할 수 있다 또한 채널마다의 설정을 다르게 설정할 수 있다

4-1. 푸쉬알림을 눌렀을 경우의 앱 내 웹뷰 화면 이동

onMessageReceived() 콜백 함수 중의 일부

```
Intent intent = new Intent( packageContext: this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP | Intent.FLAG_ACTIVITY_CLEAR_TOP);
        intent.putExtra( name: "url", url);
PendingIntent pendingIntent = PendingIntent.getActivity( context: this, REQUEST_CODE, intent,  flags: PendingIntent.FLAG_IMMUTABLE | PendingIntent.FLAG_ONE_SHOT);
```

```
builder.setSmallIcon(R.drawable.woorimanager_logo_icon)
        .setLargeIcon(BitmapFactory.decodeResource(getResources(), R.drawable.woorimanager_logo_icon))
        .setContentTitle(title)
        .setContentText(body)
        .setStyle(new NotificationCompat.BigTextStyle().bigText(body))
        .setContentIntent(pendingIntent)
        .setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION))
        .setAutoCancel(true);
```

'Intent'객체를 생성 putExtra() 메소드를 사용해서 "url"이라는 키 값에 onMessageReceived() 콜백 함수의 매개변수로 넘어온 'RemoteMessage'객체에서 추출한 url 데이터값을 저장하고 이러한 데이터가 저장되있는 'Intent'객체를 실행하기 위해서 'PendingIntent'객체를 생성해서 매개변수에 'Intent'객체의 참조변수를 넣어주고 플래그를 설정해준다 최종적으로 NotificationCompat.Builder의 setContentIntent() 메소드에 'PendingIntent'객체를 넘겨줌으로서 푸쉬알림 클릭시 'Intent'객체가 넘어가게된다

메인액티비티의 onNewIntent() 메소드 오버라이드

```
@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    setIntent(intent);

    String messageUrl = intent.getStringExtra( name: "url");
    if (messageUrl != null) {
        setContentView(binding.getRoot());
        webView.loadUrl(messageUrl);
    }
}
```

메인액티비티의 onCreate() 메소드 중 일부

```
String messageUrl = getIntent().getStringExtra( name: "url");
if (messageUrl != null) webView.loadUrl(messageUrl);
else webView.loadUrl(getString(R.string.defalut_url));
```

FCM 푸쉬 알림 클릭시 앱의 상태(포그라운드 / 백그라운드)에 따라 앱이 실행중인 경우 onNewIntent() 메소드가 호출되고 앱이 종료되서 꺼진 경우 onCreate() 메소드가 호출되서 'Intent'객체에서 getStringExtra() 메소드를 호출 "url"이라는 키 값에 저장된 데이터를 String 타입의 'messageUrl' 변수에 저장한뒤 웹뷰의 loadUrl() 메소드의 매개변수로 넘겨줘서 해당 URL의 웹뷰 화면을 띄울 수 있게 된다

5. SMS(문자메시지) 서버로 전송

```java
public class SMSReceiver extends BroadcastReceiver {
    private final ServerManager serverManager = new ServerManager();

    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(SMS_RECEIVED_ACTION)) {
            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                Object[] pdus = (Object[]) bundle.get("pdus");
                String format = bundle.getString( key: "format");
                /** PDU = Protocol Data Unit ...*/
                if (pdus != null) {
                    for (Object pdu : pdus) {
                        SmsMessage smsMessage = SmsMessage.createFromPdu((byte[]) pdu, format);
                        String address = smsMessage.getOriginatingAddress();
                        String body = smsMessage.getMessageBody();

                        StringBuilder messageBody = new StringBuilder();
                        messageBody.append(String.format("%s@@%s", address, body)).append("||");

                        String phoneNumber = getPhoneNumber(context);
                        try {
                            serverManager.messageToServer(messageBody.toString(), phoneNumber);
                        } catch (Exception e) {
                            Log.e( tag: "Exception", e.getMessage());
                        }
```

제어문으로 Intent객체가 SMS 수신했을시 Intent객체의 getExtras() 메소드를 통해 Bundle객체를 반환한다 (Bundle객체는 키-값 쌍으로 이루어진 데이터 컨테이너이며 putExtras() 메소드와 getExtras() 메소드를 사용해서 Intent에 데이터를 첨부하거나 가져올 수 있다)
이때 SMS 메시지는 PDU(Protocol Data Unit)이라는 형식으로 전달되므로 Bundle객체의 get() 메소드를 사용 “pdus”이라는 키를 사용하여 Bundle에서 해당 데이터를 가져올 수 있게 되며 “format”이라는 키를 사용하여 SMS 메시지의 형식(*‘3gpp’ = 3rd Generation Partnership Project 이동 통신 시스템과 표준을 개발하는 국제 표준화 단체이며 ‘3gpp’형식은 3GPP에서 정의한 SMS PDU 형식을 의미 SMS 메시지를 해석하는데 사용*)을 가져올 수 있다
이렇게 생성된 ‘PDU’데이터가 들어있는 ‘pdus’오브젝트 배열을 반복문 for를 통해 SmsMessage객체의 createFromPdu() 메소드의 매개변수에 ‘PDU’오브젝트와 “format”이라는 키 값에서 추출한 SMS 메시지 형식을 넘겨줌으로서 SmsMessage 객체를 반환 getOriginationAddress() 메소드와 getMessageBody() 메소드를 사용해서 발신자번호와 문자메시지를 추출할 수 있게 된다 추출한 데이터는 각각 String 타입의 ‘address’변수와 ‘body’변수에 바인딩하고 StringBuilder 객체의 append() 메소드와 String.format() 메소드를 통해 알맞은 형식으로 데이터를 재가공 한 뒤 서버로 SMS 문자메시지 데이터를 보내게 된다

6. ‘Retrofit’을 사용한 서버 통신

‘RetrofitClient’ 클래스

```
public class RetrofitClient {
    private static final String USER_CODE = "0100";
    private static final String USER_AGENT = PACKAGE_NAME.substring(PACKAGE_NAME.lastIndexOf( str: ".") + 1) + "_" + USER_CODE;
    private static final String BASE_URL = "http://payment.dandihelper.com/";
    private static Retrofit retrofit = null;

    private RetrofitClient() { }

    public static Retrofit getInstance() {
        return RetrofitSingletonHolder.getClient();
    }
```

```
private static class RetrofitSingletonHolder {
    public static Retrofit getClient() {
        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                    .baseUrl(BASE_URL)
                    .addConverterFactory(GsonConverterFactory.create())
                    .client(getHttpClient())
                    .build();
        }
        return retrofit;
    }
}
```

‘RetrofitClient’클래스는 기본적으로 싱글톤(홀더기법)을 사용해서 ‘Retrofit’인스턴스를 생성해서 사용하고 있으며 HTTP 헤더의 ‘User Agent’값을 변경하기 위해 (해당 앱을 통해 접속하는 사용자인지 구분하기위해) OkHttpClient 인스턴스를 생성 addInterceptor() 메소드를 사용해서 ‘Interceptor’인터페이스(익명)를 연결 intercept() 콜백 함수를 오버라이드해서 서버에 HTTP요청을 하기전 헤더의 “User-Agent” 키 값을 변경한 뒤 서버에 Request하게 된다

```
private static OkHttpClient getHttpClient() {
    OkHttpClient.Builder httpClient = new OkHttpClient().newBuilder();
    httpClient.addInterceptor(chain -> {
        Request requestWithUserAgent = chain.request().newBuilder().header( name: "User-Agent", USER_AGENT).build();
        return chain.proceed(requestWithUserAgent);
    });
    return httpClient.build();
}
```

‘RetrofitService’ 인터페이스

```
public interface RetrofitService {
    @FormUrlEncoded
    @POST("application/add_test.php")
    Call<ResponseBody> sendToken(@Field("reg_id") String token);

    @FormUrlEncoded
    @POST("application/add_test.php")
    Call<ResponseBody> sendTokenAndPhone(@Field("reg_id") String token, @Field("phone") String phone);

    @FormUrlEncoded
    @POST("application/add_send_conv.php")
    Call<ResponseBody> sendMessage(@Field("text") String message, @Field("phone") String phone);
```

최종적으로 서버에 POST방식으로 데이터를 전송해야되기 때문에 데이터를 전송할 상세 주소와 POST 어노테이션을 명시해줌으로써 인터페이스에서 선언한 메소드를 호출하면 서버로 데이터가 전송이 된다

6-1. ‘ServerManager’ 클래스

클래스 코드 중 일부

```java
public class ServerManager {
    private final String TAG = this.getClass().getSimpleName();
    private final Retrofit retrofit = RetrofitClient.getInstance();
    private final RetrofitService service = retrofit.create(RetrofitService.class);
    private Call<ResponseBody> call;

    public void registerDataToServer(String token) {
        tokenToServer(token);
    }
    public void registerDataToServer(String token, String phone) throws IllegalArgumentException {
        if (!isNumber(phone)) {
            tokenToServer(token);
            throw new IllegalArgumentException("registerDataToServer(String token, String phone) => Parameter String 'phone' is not number format.");
        }
        tokenAndPhoneToServer(token, phone);
    }

    public void messageToServer(String message, String phone) throws NullPointerException, IllegalArgumentException {
        if (phone == null) {
            throw new NullPointerException("messageToServer(String message, String phone) => Parameter String 'phone' is null.");
        } else if (!isNumber(phone)) {
            throw new IllegalArgumentException("messageToServer(String message, String phone) => Parameter String 'phone' is not number format.");
        }
        call = service.sendMessage(message, phone);
        call.enqueue(new Callback<ResponseBody>() {
            @Override
            public void onResponse(@NonNull Call<ResponseBody> call, @NonNull Response<ResponseBody> response) {
                if (response.isSuccessful()) {
                    Log.d(TAG,  msg: "Message sent to server succeeded and response received good!");
                } else {
                    Log.e(TAG,  msg: "Message sent to server succeeded and response received fail!");
                }
            }
```

앞서 구현한 ‘RetrofitClient’클래스와 ‘RetrofitService’인터페이스를 이용해서 ‘RetrofitClient’의 getInstance() 메소드를 사용 Retrofit 인스턴스를 생성하고 create() 메소드를 사용해서 ‘RetrofitService’인터페이스를 연결해서 인터페이스에서 선언한 메소드들을 사용해서 서버에 연결을 한다 이때 ‘Call’ 인터페이스를 통해 비동기적으로 HTTP 요청을 실행하고 응답을 처리할 수 있으며 enqueue() 메소드를 사용해서 서버의 응답이 수신되었을 경우의 로직을 처리 할 수 있다