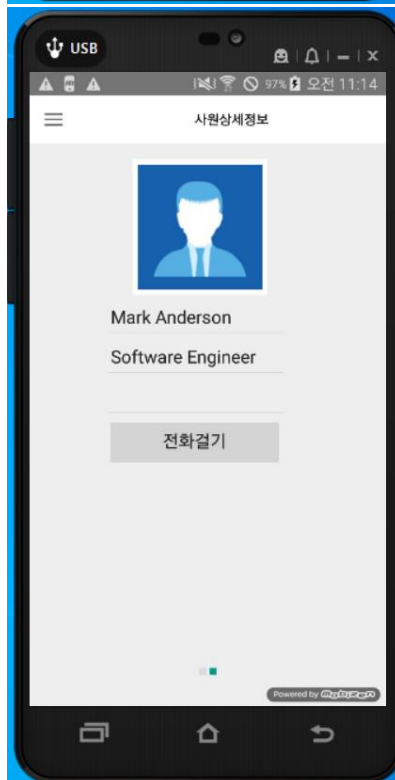
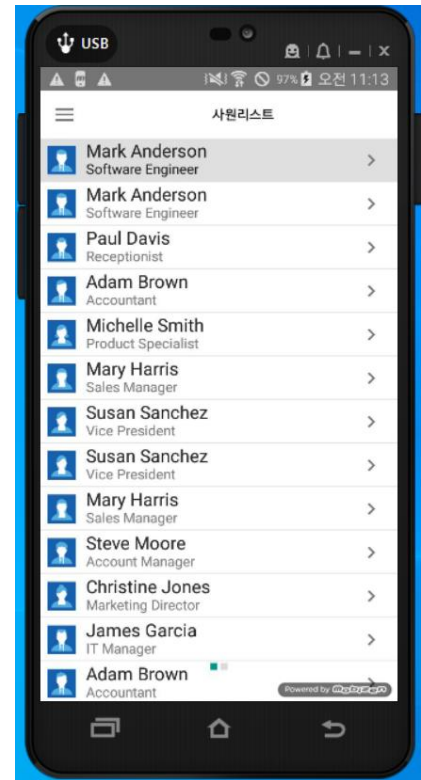
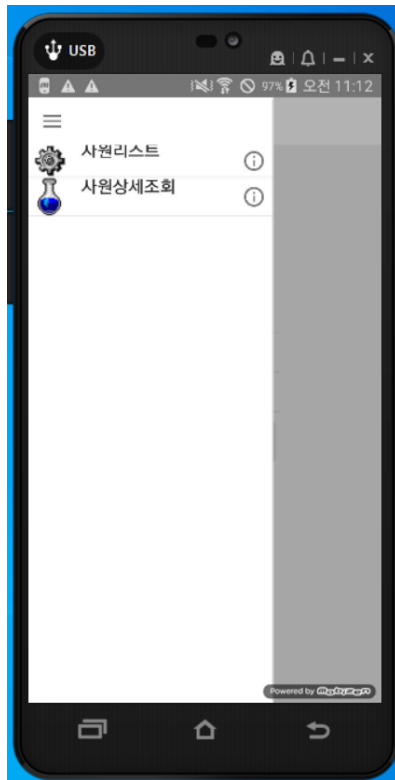


사원관리 앱 만들기

모바일 앱에서 가장 기본이 되는 컴포넌트를 사용하여 마스터-디테일 화면을 작성해 본다.

[완성된 화면]



실습목적

- 모바일에서 많이 사용하는 컴포넌트를 사용한 화면 인터페이스, 마스터-디테일 화면 작성
- 터치, 제스처 사용하여 탭 간 이동
- 라이브바인딩을 이용하여 데이터를 연결하기
- 컴포넌트가 아닌 직접 인터페이스(전화걸기)가 지원하는 메소드 호출하기
- 안드로이드 전화걸기 권한 체크

사용 컴포넌트

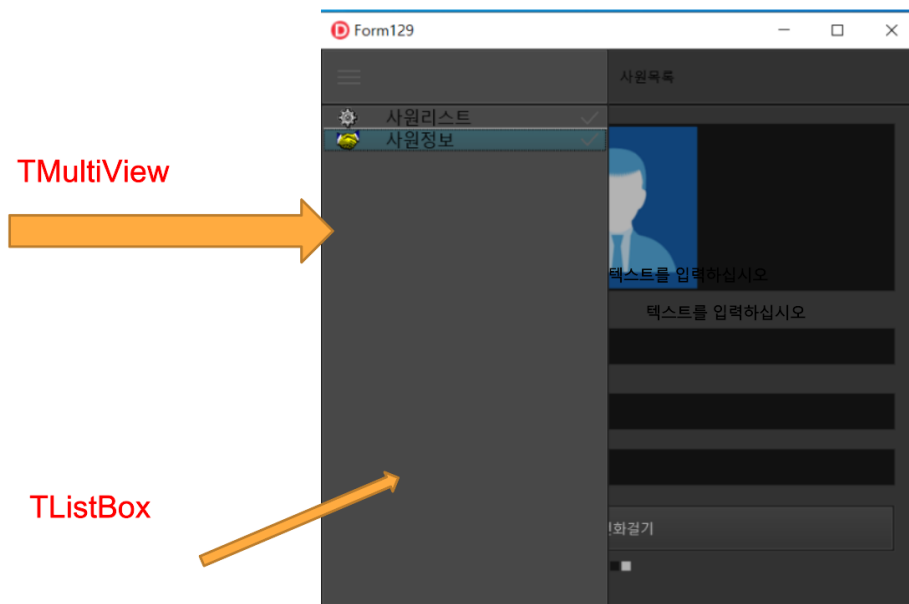
- TStyleBook
- TMultiView
- TLayout
- TToolBar
- TTabControl
- TTabItem
- TListView
- TEdit
- TButton
- TActionList
- TGestureManager
- TProtoTypeBindSouece

인터페이스

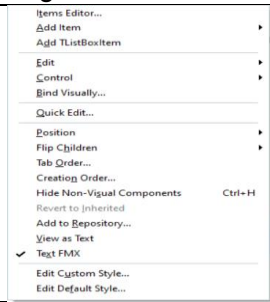
- IFMXPhoneDialerService

[따라하기]

1. File < New < Multi Device Application - Delphi을 이용하여 어플리케이션을 시작한다.
2. File < Save All 혹은 단축키 Shift + Ctrl + S을 선택하여 USawonList.pas로, 프로젝트의 이름은 PSawonList.dpr로 프로젝트를 저장한다.
3. 생성한 폼에 TMultiView 컴포넌트를 내려놓습니다.



4. MultiView 컴포넌트 위에 다음과 같이 컴포넌트를 내려놓고 속성을 지정한다.

컴포넌트	속성	값
TToolBar	Align	Top
TButton	Align	MostLeft
	StyleLookUp	drawertoolbutton
TListBox	Align	Client
		오른쪽 마우스를 클릭하여 TListBoxItem을 2개 추가하여 각각의 속성을 설정합니다.
TListBoxItem	ItemData.Text	각각을 '사원리스트', 사원상세조회'

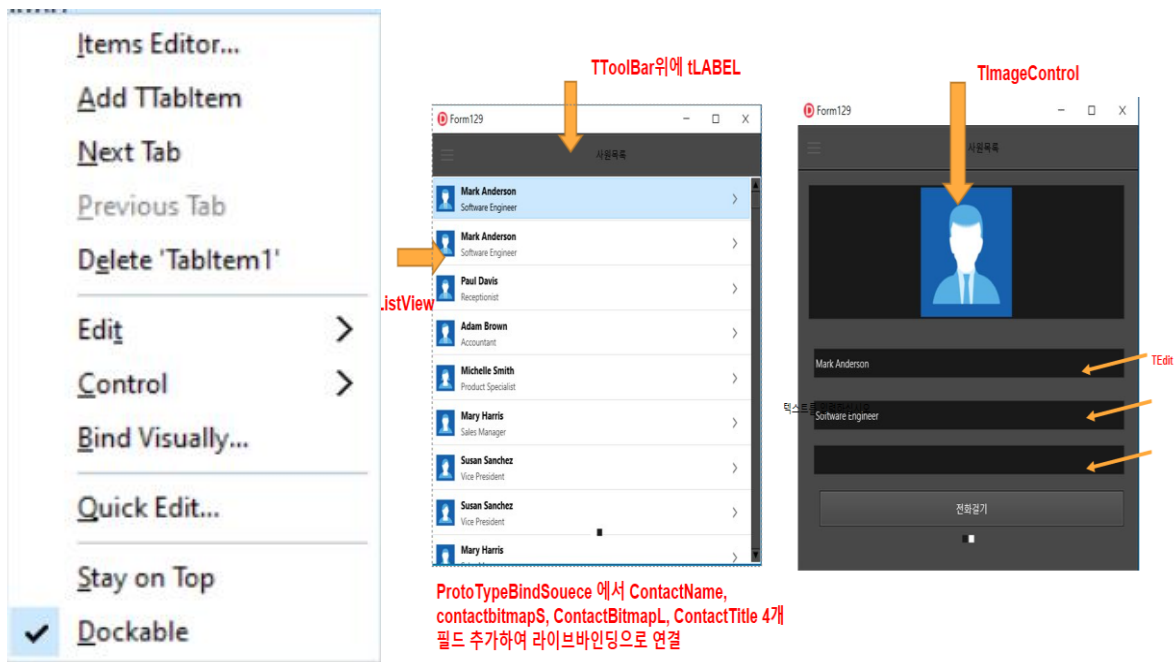
	ItemData.Accessory	aMore
	ItemData.Bitmap	원하는 이미지
	StyleLookUp	listboxitembottomdetail
TMultiView	Mode	Drawer

5. 폼에 TLayout 컴포넌트를 내려놓고 Align 속성을 Client 라고 지정한다.

6. 그 위에 다음과 같은 순서로 컴포넌트들을 내려놓고 속성을 지정한다.

컴포넌트	속성	값
TToolBar	Align	Top
툴바위에 TLabel	Text	사원리스트
툴바위에 TButton	Align	MostLeft
	StyleLookUp	drawertoolbutton

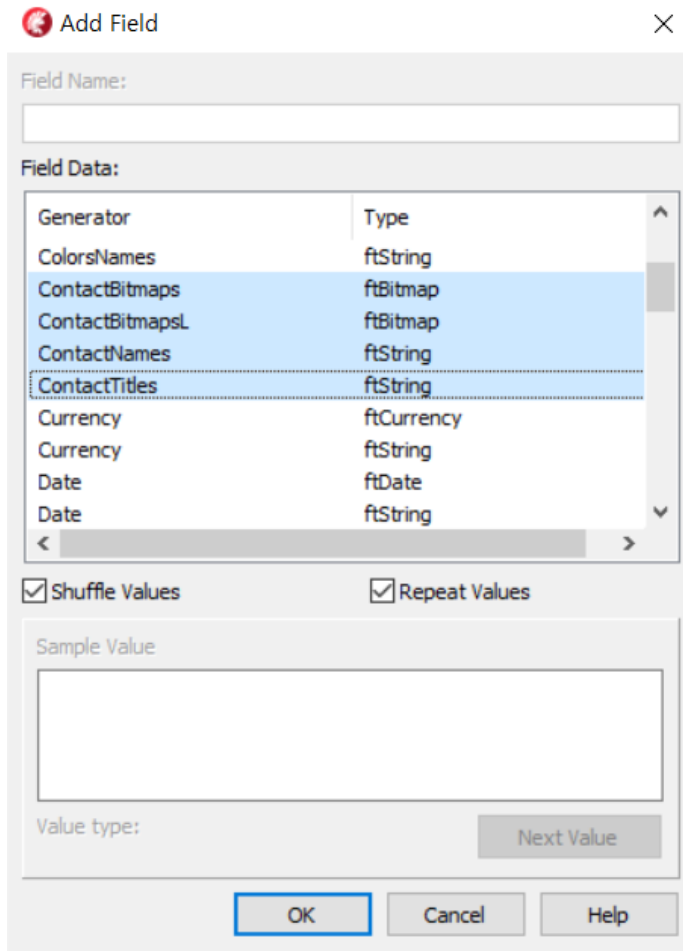
7. 폼에 TTabControl 컴포넌트를 내려놓고 Add TTabItem을 2개 추가한다. TabPosition 속성을 Dots로 지정한다.



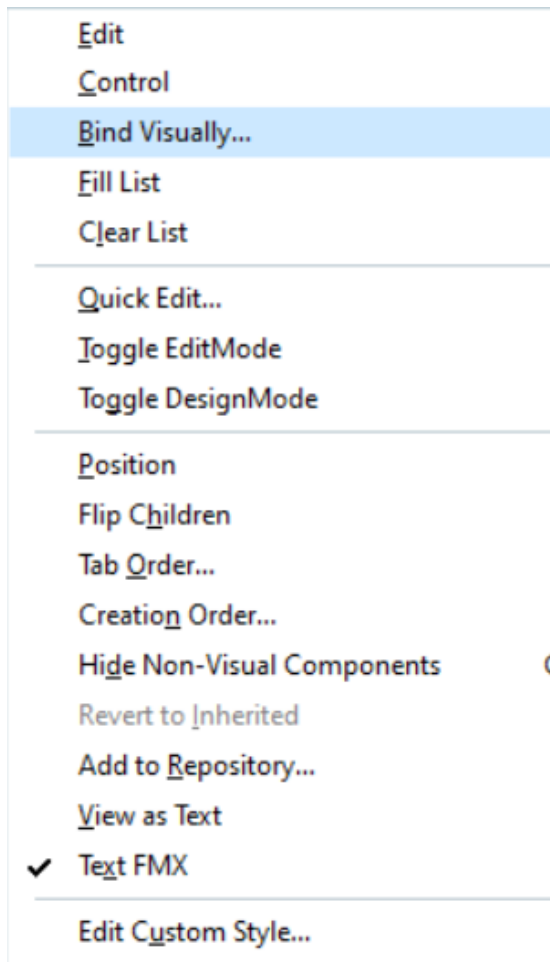
8. 각 TabItem에 위와 같이 컴포넌트를 위치하고 속성을 지정한다.

컴포넌트	속성	값
TabItem1에 TListView 컴포넌트를 내려놓은다	Align	Client
	ItemAppreance. ItemAppreance	ImageListItemBottomDetail
TabItem2에 TImageControl		
TEdit 3개		
TButton	Text	전화걸기

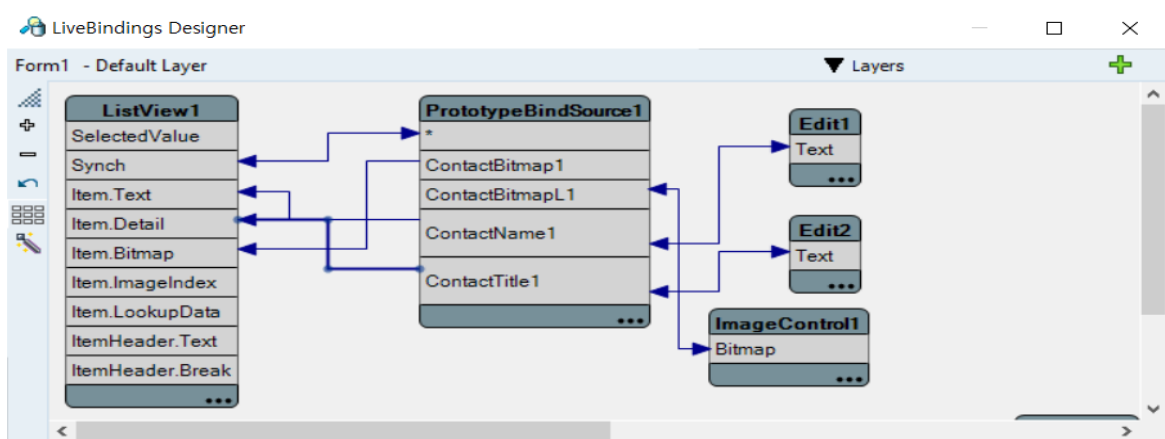
9. TPrototypeBindSouce 컴포넌트를 내려놓고 오른쪽 마우스를 클릭하여 Add Fields..를 선택하여 아래 그림과 같이 필드(4개)를 추가한다.



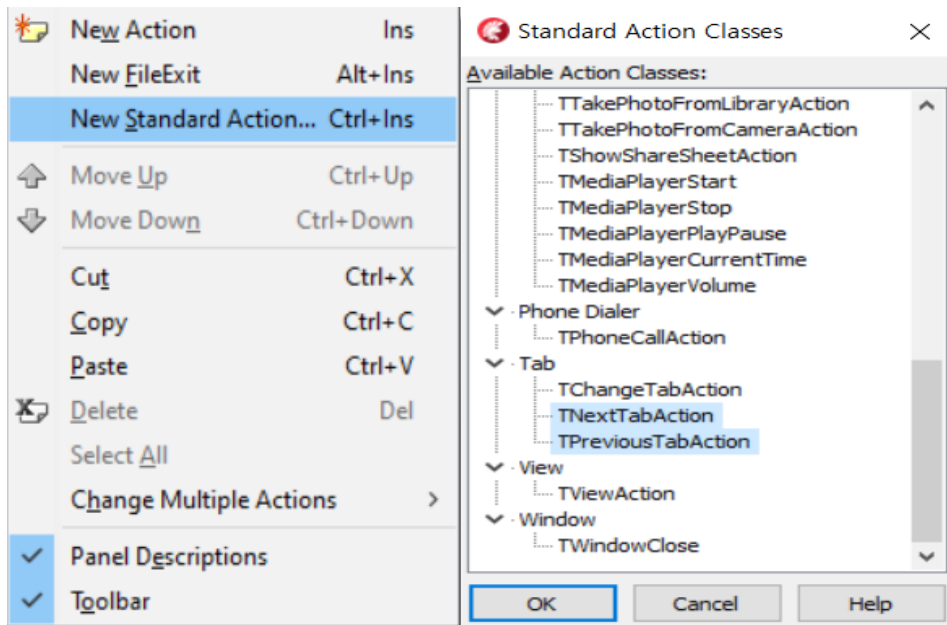
10. 폼 위에서 오른쪽 마우스를 클릭하여 아래와 같이 Bind Visually...를 선택한다.



11. 아래 그림과 같이 드래그&드롭을 이용하여 연결합니다.(Synch --> * 연결 필수)



12. TActionList 컴포넌트를 내려 놓고 오른쪽 마우스를 클릭하여 New Action > New Standard Action..>Tab의 TNextTabAction, TPreviousTabAction을 선택하여 추가한다. 각 액션의 TabControl 속성을 TabControl1으로 지정한다.



13. 탭간 이동하기 위하여 TGestureManager 컴포넌트를 내려놓고 다음과 속성을 지정한다.

TToolBar	Touch.GestureManager	GestureManager1
Ttouch.GestureManager.standard 에서 left , Right를 선택 하고	Action	각각 NextTabAction, PreviousTabAction 으로 연결 한다.
TTabControl 에도 동일하게 적용하다.		

15. 윈도우에서 실행하여 데이터가 정확히 표시되고 탭간에 이동이 잘 이루어지는지 확인 한다.

16. MultiView상의 TButton의 클릭 이벤트 핸들러를 다음과 같이 작성합니다..

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Multiview1.HideMaster;
end;
```

17. MultiView상의 TListBox의 OnChange 이벤트 핸들러를 다음과 같이 작성한다.

```
procedure TForm1.ListBox1Change(Sender: TObject);
begin
    case ListBox1.ItemIndex of
        0: begin
            TabControl1.ActiveTab := Tabitem1;
            Text1.text := '사원리스트';
        end;
        1: begin
            TabControl1.ActiveTab := Tabitem2;
            Text1.text := '사원상세정보';
        end;
    end;
    MultiView1.HideMaster;
end;
```

18. TToolBar2(TLayout에 있는)의 TButton의 클릭 이벤트 핸들러를 다음과 같이 작성합니다.


```

procedure TForm1.Button2Click(Sender: TObject);
begin
    multiview1.ShowMaster;
end;

```

19. 전화걸기(권한 체크)를 구현해 보도록 하겠습니다. (어려우신 분들은 참조만 해주셔도 됩니다. 이 부분은 소스 부분을 복사해서 붙여 사용합니다.)

- 폼의 **Private** 부분에 다음과 같은 루틴들을 선언합니다.

```

FPhoneDialerService: IFMXPhoneDialerService;
FCallPhonePermission: string;
procedure DisplayRationale(Sender: TObject; const APermissions: TArray<string>; const
APostRationaleProc: TProc);
procedure MakePhoneCallPermissionRequestResult(Sender: TObject; const APermissions:
TArray<string>; const AGrantResults: TArray<TPermissionStatus>);

```

- Implementation 부분의 **uses**절에 아래와 같이 추가합니다.

```

implementation
uses
    {$IFDEF ANDROID}
        Androidapi.Helpers,
        Androidapi.JNI.JavaTypes,
        Androidapi.JNI.Os,
    {$ENDIF}
    FMX.DialogService, FMX.Platform;
{$R *.fmx}

```

위에 선언된 루틴들을 **ctrl + Shift + c** 를 눌러 블록안에 다음과 같이 코드를 구현합니다.

```

procedure TForm1.DisplayRationale(Sender: TObject;
const APermissions: TArray<string>; const APostRationaleProc: TProc);
begin
    TDialogService.ShowMessage('The app needs to be able to support your making phone calls',
    procedure(const AResult: TModalResult)
    begin
        APostRationaleProc;
    end)
end;

procedure TForm1.MakePhoneCallPermissionRequestResult(Sender: TObject;
const APermissions: TArray<string>;
const AGrantResults: TArray<TPermissionStatus>);
begin
    if (Length(AGrantResults) = 1) and (AGrantResults[0] = TPermissionStatus.Granted) then
        FPhoneDialerService.Call(Edit3.Text)
    else
        TDialogService.ShowMessage('Cannot make a phone call because the required permission

```

```
has not been granted');  
end;
```

- Interface uses 절에 FMX.PhoneDialer, System.Permissions을 추가합니다.
- Form의 OnCreate 핸들러를 아래와 같이 작성합니다.

```
procedure TForm1.FormCreate(Sender: TObject);  
begin  
{IFDEF ANDROID}  
FCallPhonePermission :=  
JStringToString(TJManifest_permission.JavaClass.CALL_PHONE);  
{ENDIF}  
  { test whether the PhoneDialer services are supported }  
end;
```

- 전화걸기 버튼의 onClick 이벤트 핸들러를 작성합니다.

```
procedure TForm1.Button3Click(Sender: TObject);  
begin  
if Edit3.text = '' then exit;  
  
If  
TPlatformServices.Current.SupportsPlatformService(IFMXPhoneDialerService, IInterface(FPhoneDialerService)) then  
  
PermissionsService.RequestPermissions([FCallPhonePermission], MakePhoneCallPermissionRequestResult, DisplayRationale)  
else  
  TDialogService.ShowMessage('전화 걸기가 지원이 안됨');  
end;
```