

embarcadero[®] Developer Camp

【T2】Delphiテクニカルセッション

こんなに簡単！ Delphi によるiOS/Android
業務アプリケーション開発！

株式会社ミガロ.
吉原 泰介

ミガロ.について



株式会社ミガロ.

<http://www.migaro.co.jp/>

会社情報

所在地: 本社 大阪市浪速区湊町2-1-57
難波サンケイビル13F

東京事業所 東京都港区麻布台1-4-3
エグゼクティブタワー麻布台11F

事業内容

IBM i 向けのソフトウェア・ツール販売および技術サポート

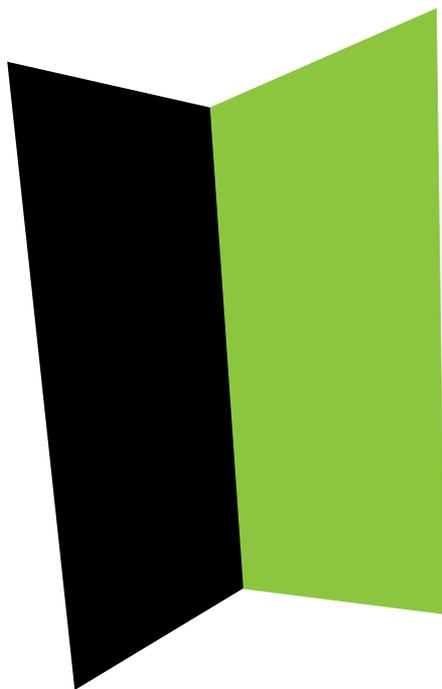
【開発ツール】

- ・Delphi/400
- ・JC/400

【スマートデバイス向けツール】

- ・Business4Mobile

- Delphi/400
- DelphiをIBM i (AS/400)に完全対応させたミドルウェア
- 国内約700社、全世界約6,000社の導入実績



アジェンダ

アジェンダ

1. 企業導入が進むスマートデバイス

2. スマートデバイスアプリの種類

3. ネイティブアプリの開発

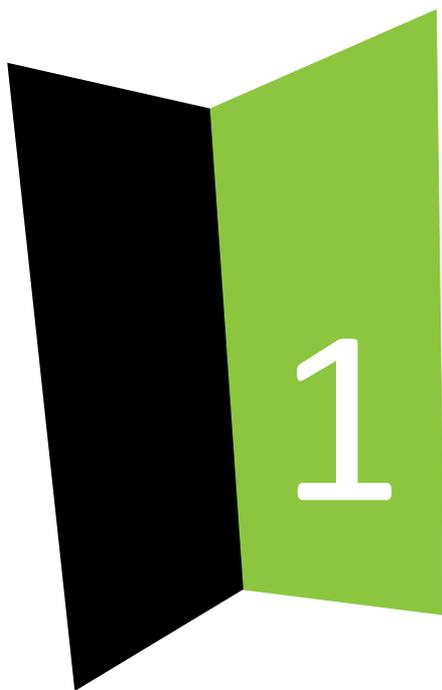
3-1. ネイティブアプリの開発環境

3-2. 簡単なネイティブアプリの開発

3-3. DBに接続するネイティブアプリの開発

3-4. ネイティブアプリの配布

4. まとめ



企業導入が進む スマートデバイス

1. 企業導入が進むスマートデバイス

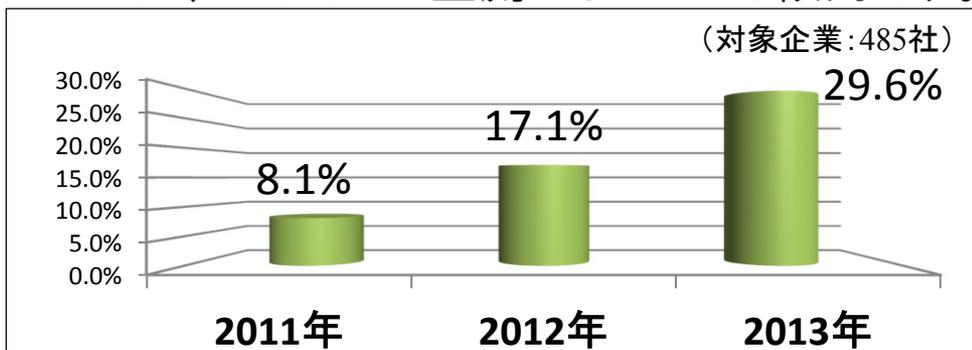
- スマートデバイス企業導入率と導入OSの傾向

この1~2年でスマートデバイスの法人への導入が急速に進んでいきます。

また企業で導入されるスマートデバイスはiOS、Androidが主流になっている傾向です。

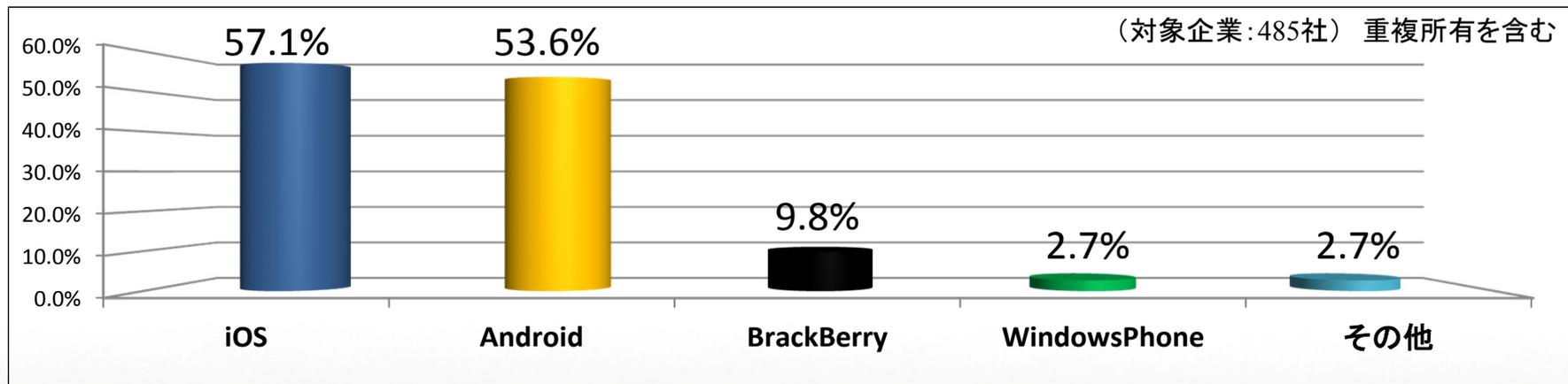
スマートデバイス企業導入率の遷移

2011年に比べると2013年では
導入率が3.5倍以上に増加



企業導入スマートデバイス比率(2013年)

スマートデバイスを導入している企業の大半がiOS、Androidを採用



1. 企業導入が進むスマートデバイス

- スマートデバイスアプリの業務利用

スマートデバイス導入企業の多くは「業務効率化」を導入目的としています。

→ 今後はPCアプリケーション(Windows)だけではなく、
スマートデバイスアプリも業務利用が増加

PCアプリケーションとスマートデバイスアプリケーションの特徴の違い



PCアプリの特徴

細かいキーボード入力ができる

画面が大きく見やすい

ネットワークが安定している

携帯性が低い

スマートデバイスアプリの特徴

どこでもすぐに使用できる

タッチで感覚的に操作できる

カメラ、GPS、センサー等が利用できる

細かいキーボード入力は不向き



スマートデバイスアプリ の種類

2. スマートデバイスアプリの種類

- スマートデバイスで利用されるアプリケーション
スマートデバイス(iOS、Android)で利用することができるアプリは、大きく2種類のアプリケーションに分かれます。
(ハイブリッドアプリケーションは折衷形式)

ネイティブアプリケーション

Webアプリケーション

2.スマートデバイスアプリの種類

ネイティブアプリケーション

ネイティブアプリケーションは、スマートデバイス端末上で動作してデバイス機能と連携ができるアプリケーションです。

App Store や PlayStore といったストアや、社内向けに公開したWebサーバ等からインストールして利用することができます。

一般的特徴

- ・デバイス機能(カメラ、GPS等)が利用できる
- ・レスポンスが良い
- ・オフラインでも利用できる

開発言語例

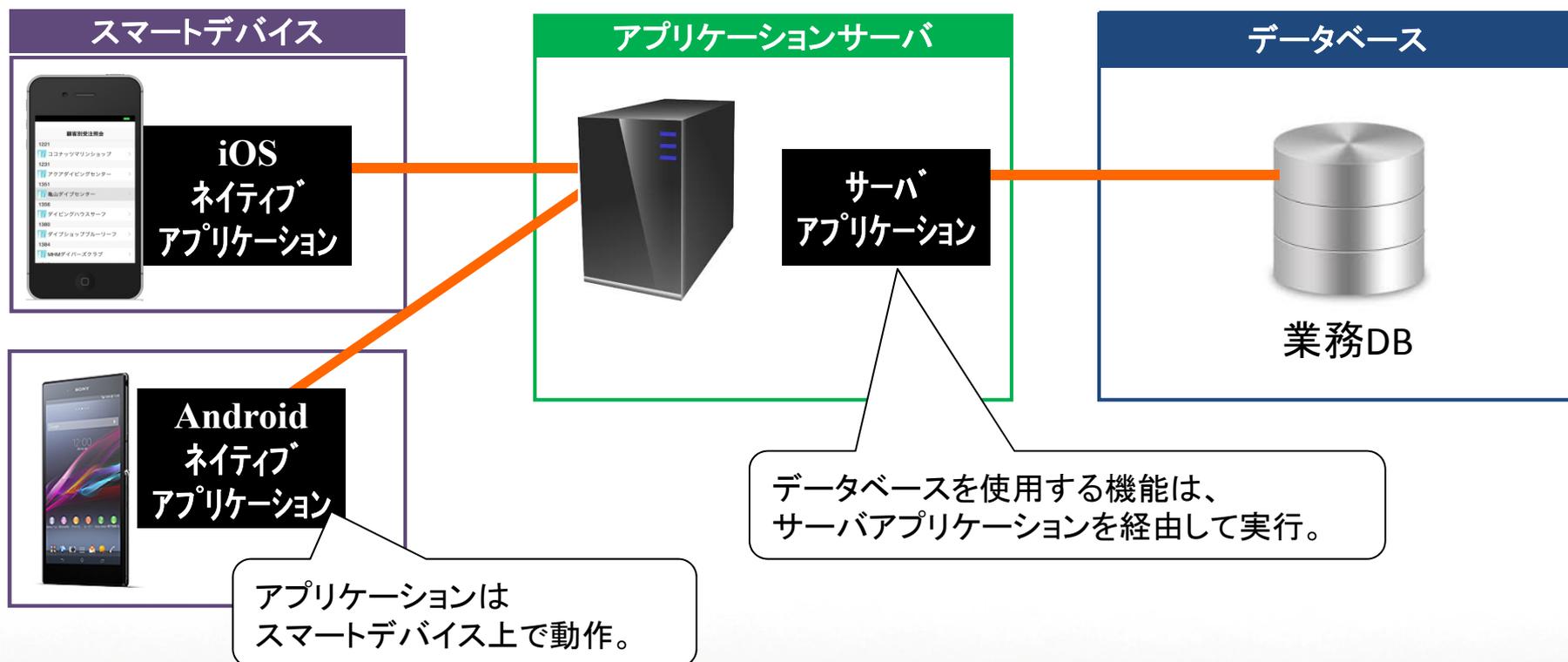
- ・iOS: Objective-C
- ・Android: Java等



2.スマートデバイスアプリの種類

ネイティブアプリケーション

ネイティブアプリケーションの環境構成



2.スマートデバイスアプリの種類

Webアプリケーション

Webアプリケーションは、Webサーバ上で動作するプログラムをPC同様にブラウザから利用できるアプリケーションです。スマートデバイス端末にアプリケーションはインストールされないため、ブラウザのブックマーク等を使って利用することができます。

一般的特徴

- ・ブラウザで実行するため、プラットフォームを問わず汎用的に開発・利用できる。
- ・インストールが不要なため、利用が容易。

開発言語例

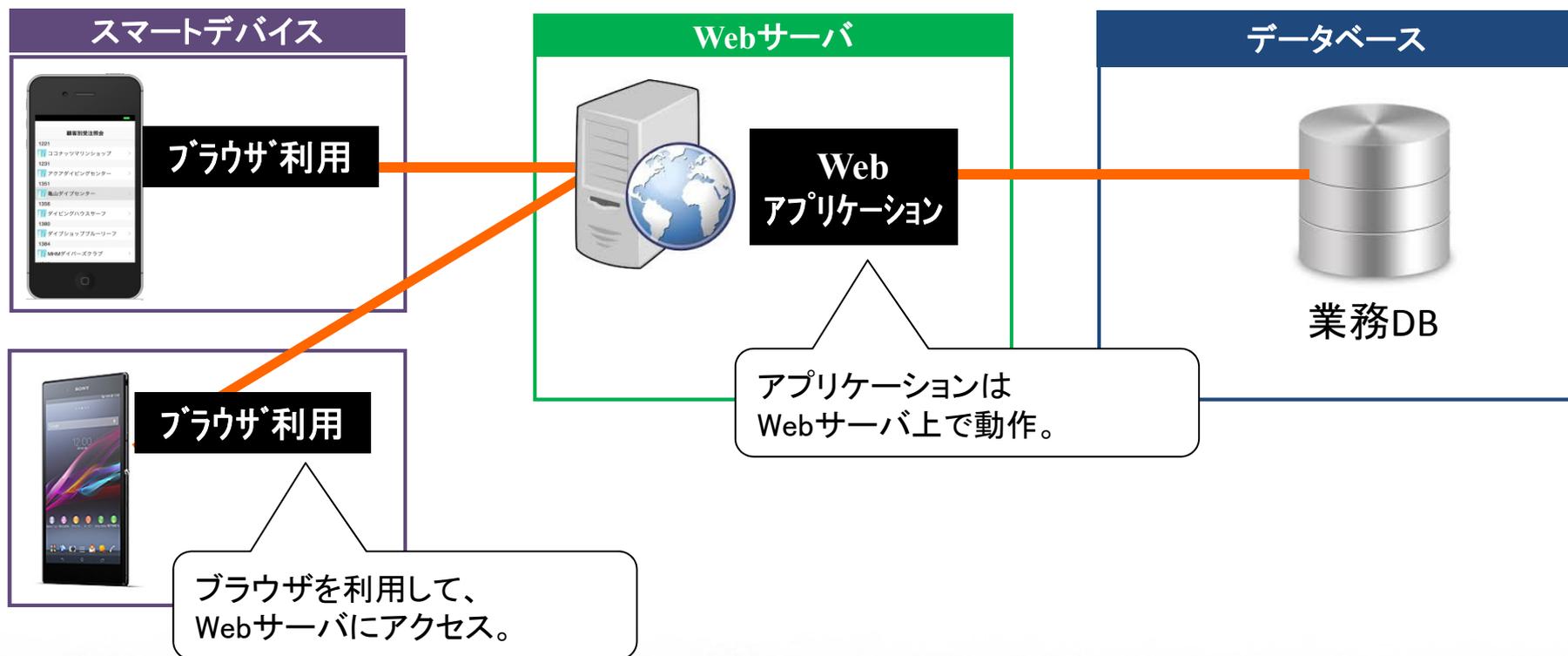
- ・HTML5、JavaScript、PHP、java等



2. スマートデバイスアプリの種類

Webアプリケーション

Webアプリケーションの環境構成



2.スマートデバイスアプリの種類

- ネイティブ / Webアプリの特徴

	ネイティブ	Web
開発言語	iOS:Objective-C Android:Java	HTML javascript等
開発生産性	△	○
デバイス機能	◎	△
パフォーマンス	◎	○
オフライン動作	◎	×
配布	△	◎

多くの開発言語習得が必要

言語によって開発環境も煩雑になるため、大変

2.スマートデバイスアプリの種類

- ネイティブ / Webアプリの特徴(Delphi)

	ネイティブ	Web	
開発言語	Delphi	Delphi	開発言語をDelphiで統一できる
開発生産性	◎	◎	Delphiの開発機能は生産性が高い
デバイス機能	◎	△	
パフォーマンス	◎	○	
オフライン動作	◎	×	
配布	△	◎	

2.スマートデバイスアプリの種類

・Delphiネイティブアプリケーションの強み

Delphiスキルで iOS / Android ネイティブ開発ができる

開発言語はDelphiだけで iOS / Android のネイティブ開発ができます。
またコンパイルの設定切り替えだけで、1つのプログラムから
iOS / Android 両方のデバイスに対応できます。
XE7では、デバイス毎に細かい画面変更管理も可能です。(FireUI)

従来と同じ手法でネイティブ開発ができる

コンポーネントで画面設計して、イベントでプログラムコーディングする
従来の開発手法でネイティブアプリケーションが開発できます。

デバイス連携機能を簡単に開発することができる

スマートデバイス連携機能(カメラやGPS等)を専用コンポーネントで、
簡単に開発することができます。

2. スマートデバイスアプリの種類

- ネイティブアプリのデバイス機能連携

ネイティブアプリケーションではカメラ連携、バーコード連携、GPS連携といったデバイス連携機能をアプリケーションへ簡単に実装できます。

カメラ撮影



録音・再生



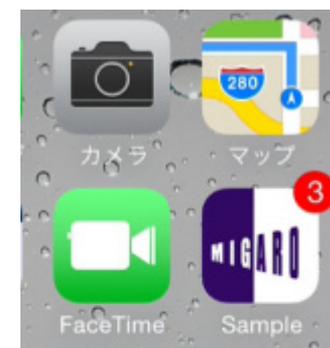
バーコード読取



GPS位置情報取得

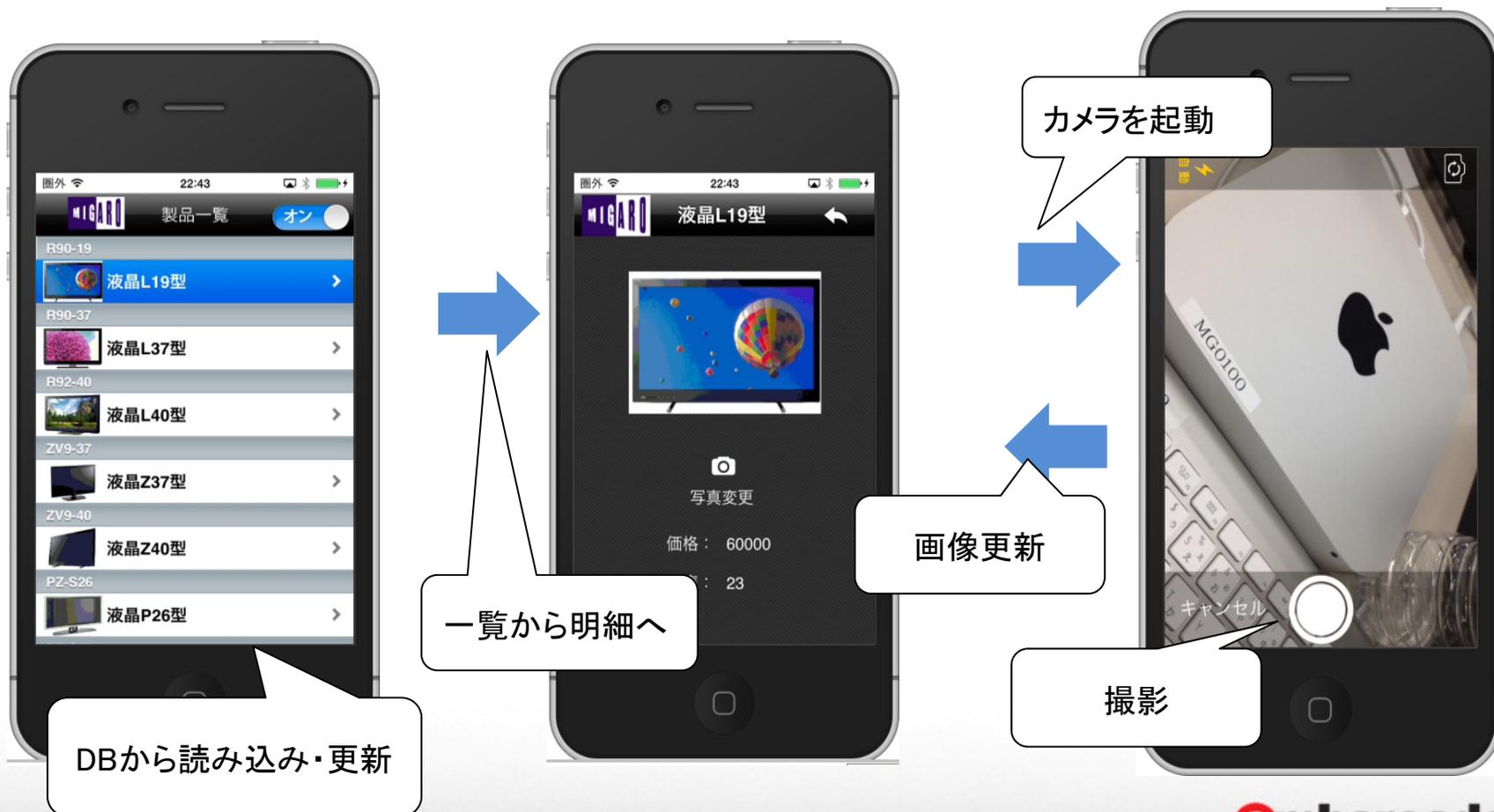


通知etc



2.スマートデバイスアプリの種類

- ネイティブアプリのデバイス機能連携例1
カメラ機能を連携したネイティブアプリケーション



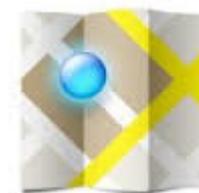
2.スマートデバイスアプリの種類

- ネイティブアプリのデバイス機能連携例2
バーコード、QRコード読取りを活用したネイティブアプリ



2. スマートデバイスアプリの種類

- ネイティブアプリのデバイス機能連携例3
GPSを使って地図連携を活用したネイティブ



現在位置から
GoogleMapを呼び出す

DBへ位置情報を
登録する

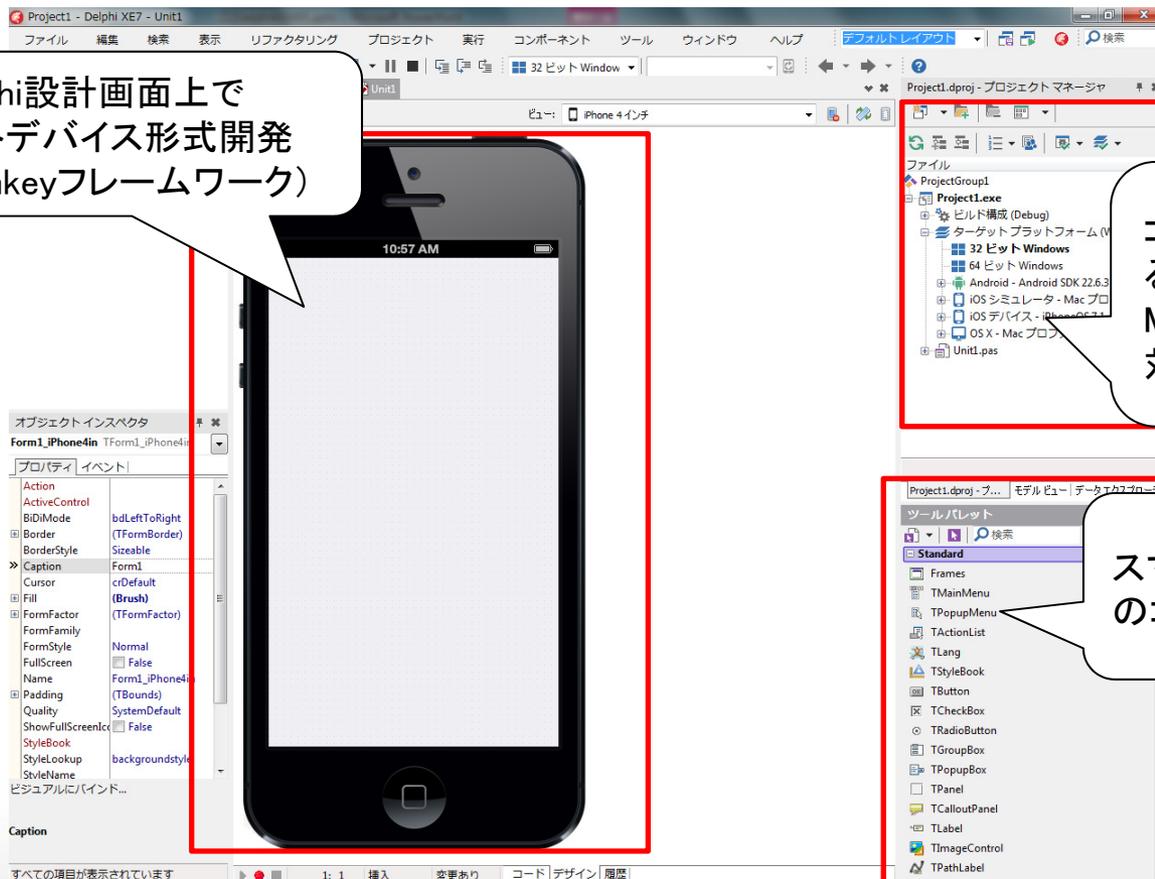


ネイティブアプリ の開発

3-1. ネイティブアプリの開発環境

- Delphi のネイティブアプリケーション開発機能

Delphi では、従来のWindowsアプリケーション向けVCL開発機能に加え、FireMonkeyフレームワークを利用することで、Mac、iOS、Androidなどのマルチデバイス開発が可能です。



Delphi設計画面上で
スマートデバイス形式開発
(FireMonkeyフレームワーク)

コンパイルを切り替えることで、Windows、Mac、iOS、Androidに対応

スマートデバイス向けのコンポーネントも充実

3-1. ネイティブアプリの開発環境

- iOSネイティブアプリ開発に必要な環境

- Windows端末 (Delphi)
- Mac端末 (OSX 10.8~10.9)
- iOS Developer Program (Xcode, 配布)
- iOS実機 (iPhone、iPad等 iOS7~7.1)

8はupdate1?



Mac上にWindows仮想環境を作成して、1台で構成することも可能です

開発

コンパイル

インストール・実行

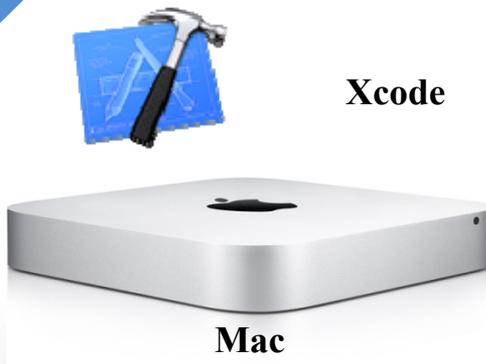


Windows



PA Server(付属)

Xcode



Mac



iPhone



iPad

3-1. ネイティブアプリの開発環境

- Androidネイティブアプリ開発に必要な環境
 - Windows端末 (Delphi)
 - Android実機
(Android 2.3.3以降のARM7 + NEON対応デバイス)

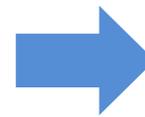
開発

コンパイル

インストール・実行



Windows



Android
Phone/Tablet

3-2. 簡単なネイティブアプリの開発

- 簡単なネイティブアプリの開発

この部分だけを開発します

スマートデバイス



iOS
ネイティブ
アプリケーション



Android
ネイティブ
アプリケーション

アプリケーションサーバ



サーバ
アプリケーション

DB



3-2. 簡単なネイティブアプリの開発

- デモで開発するネイティブアプリケーション
スマートデバイス機能(カメラ)を連携したアプリケーションを開発

iPhone



Android



ボタンをタッチするとカメラで撮影が行え
撮影した写真を画面に取り込みます。

3-2. 簡単なネイティブアプリの開発

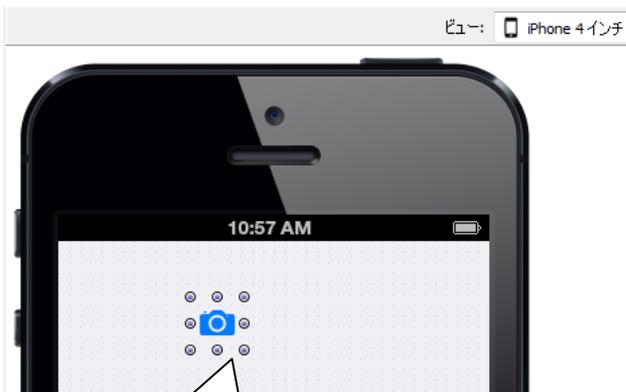
- ネイティブアプリケーション開発の流れ

ネイティブアプリケーションの開発は、従来のC/S、Web開発手法と同じ

① コンポーネントの配置

② イベントにプログラミング

③ コンパイル
/インストール



コンポーネントを使って
画面設計

```
implementation
{$R *.fmx}
{$R *.iPhone4in.fmx IOS}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Text := 'クリックしました!';
end;
end.
```

Delphi言語で
プログラミング

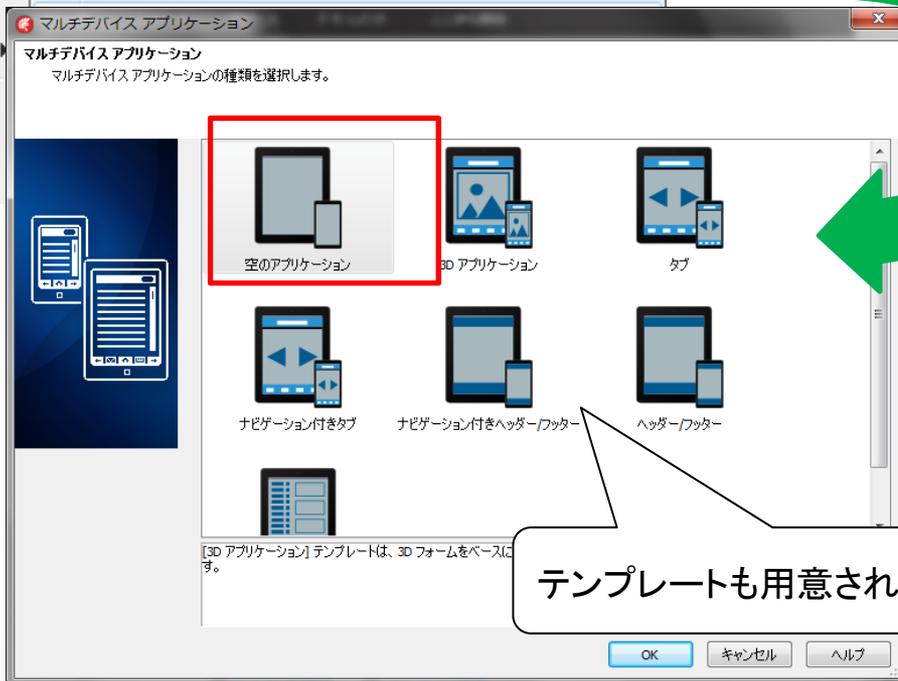
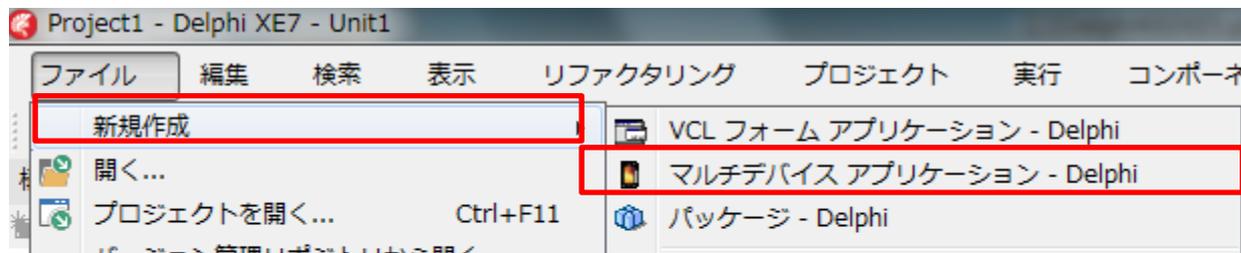


ターゲットのスマートデバイスに
コンパイルしてインストール

3-2. 簡単なネイティブアプリの開発

• ネイティブアプリ開発手順1

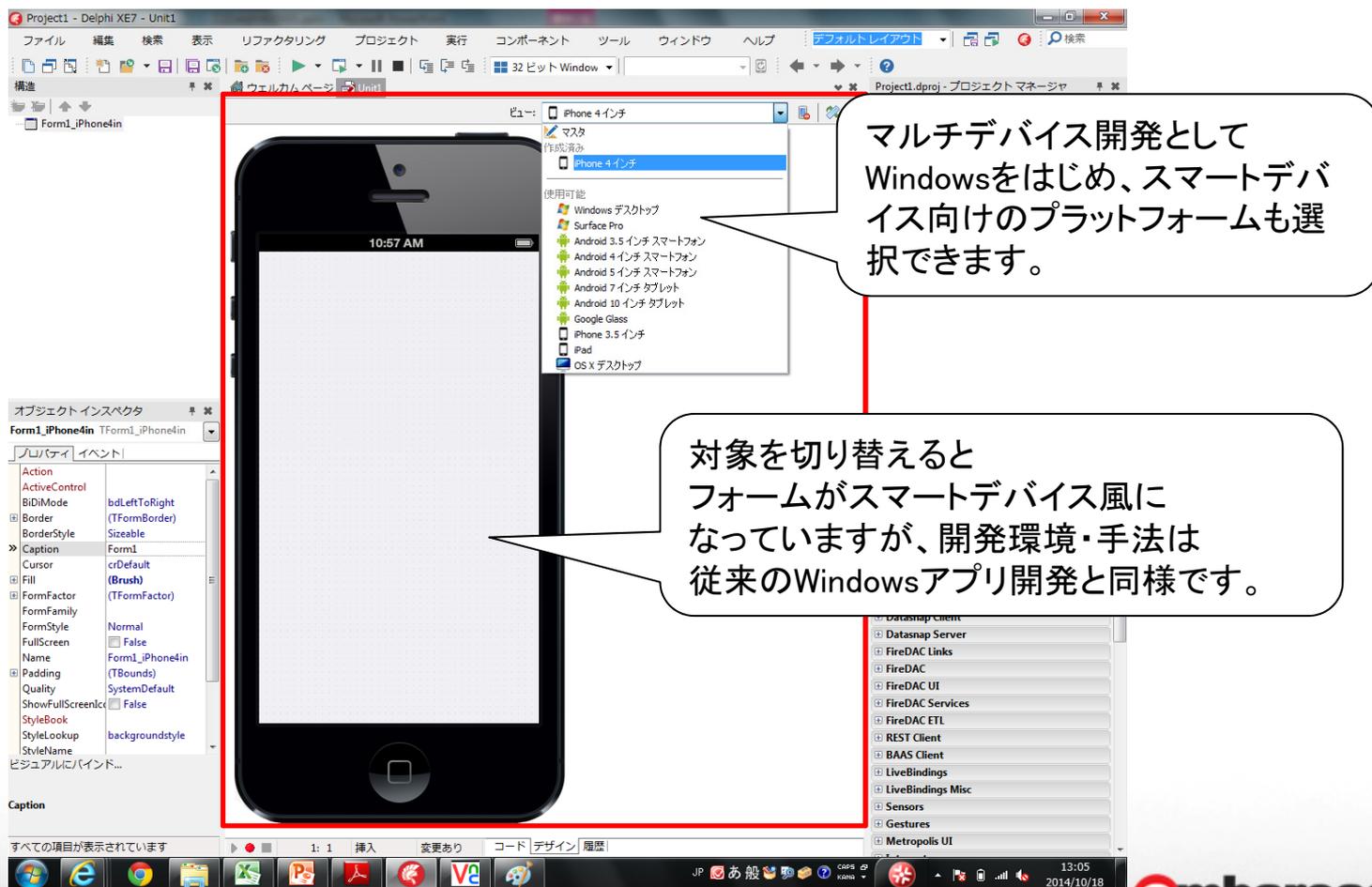
新規作成よりマルチデバイスアプリケーションを選択



3-2. 簡単なネイティブアプリの開発

• ネイティブアプリ開発手順2

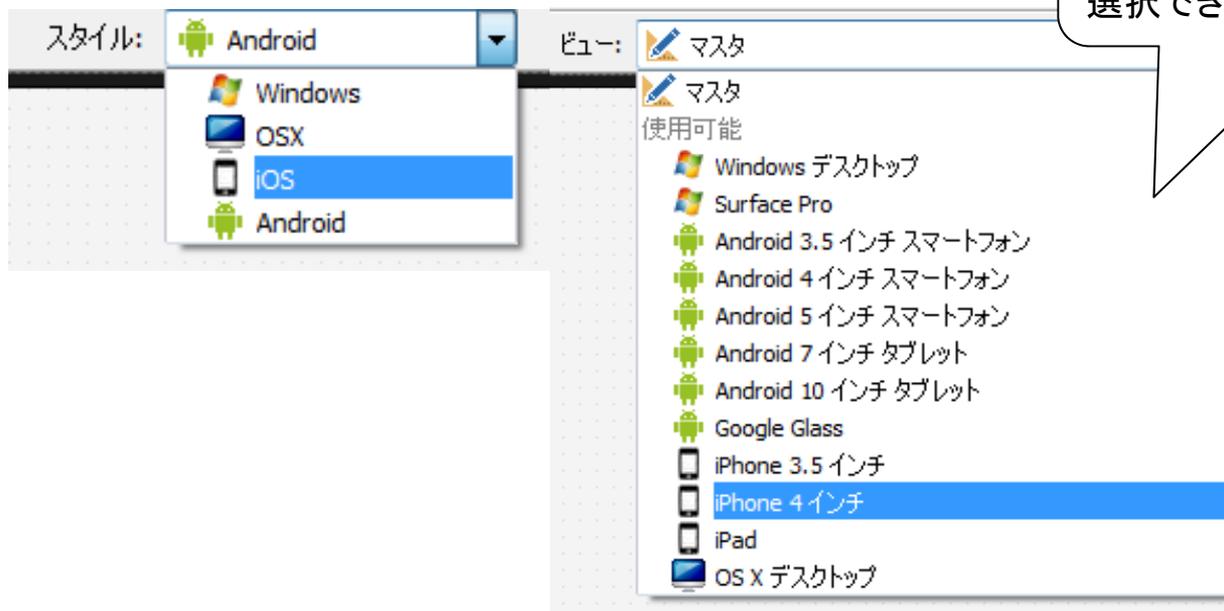
ネイティブアプリ開発画面



3-2. 簡単なネイティブアプリの開発

• ネイティブアプリ開発手順3

フォームスタイルを選択

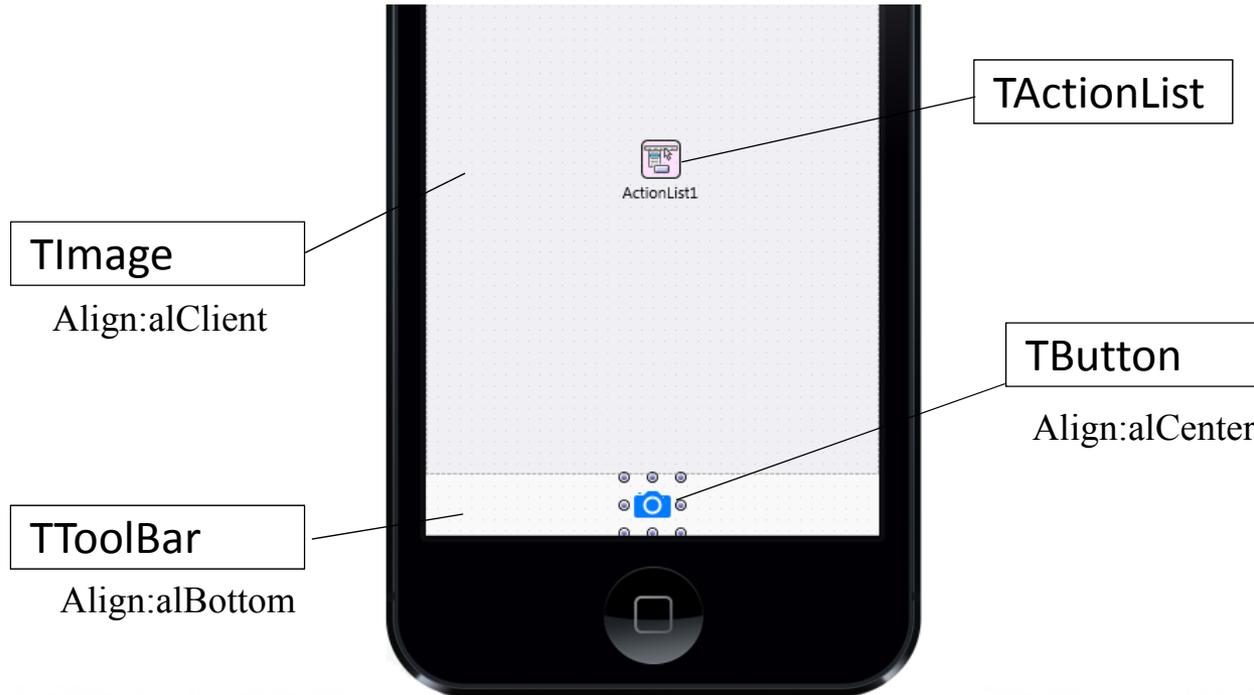


3-2. 簡単なネイティブアプリの開発

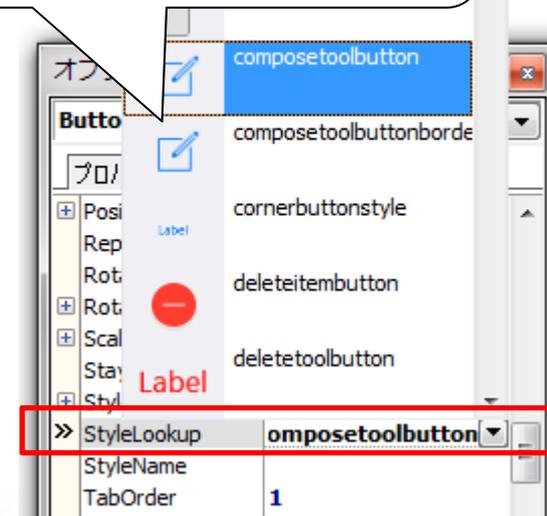
- ネイティブアプリ開発手順4

フォームに次のコンポーネントを配置

TToolBar、TButton、TImage、TActionList



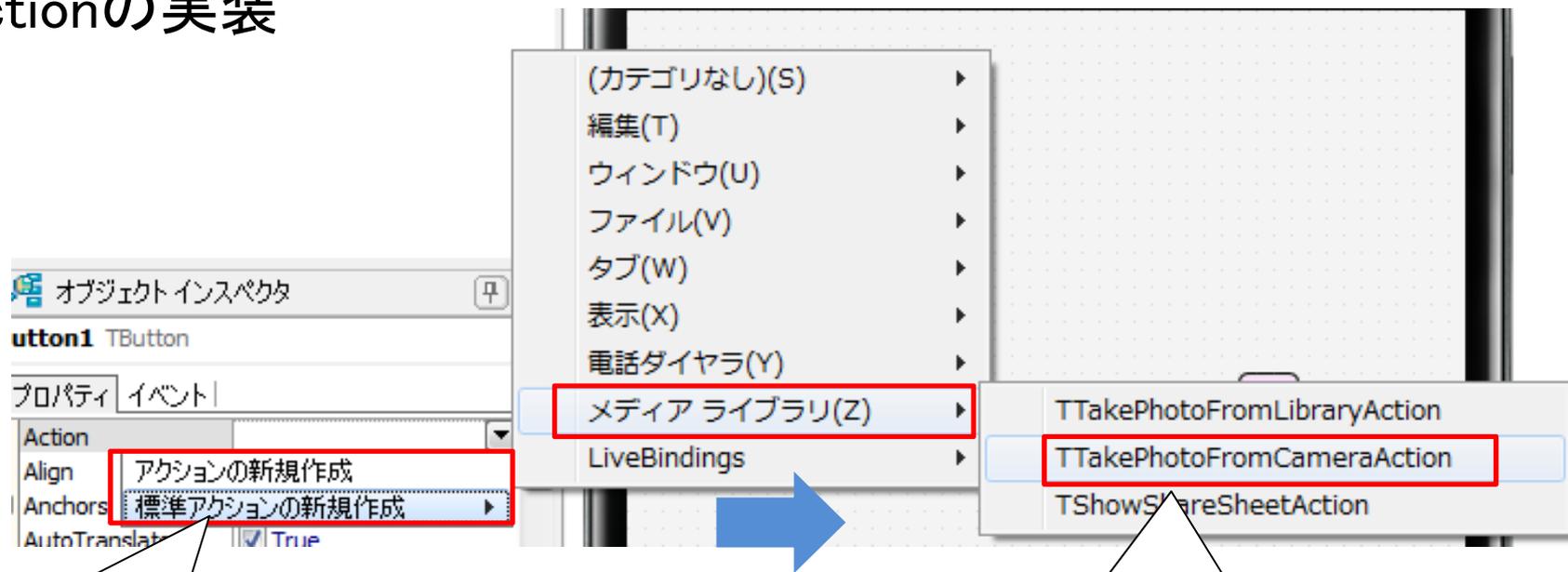
ボタンの見た目は
StyleLookupプロパティに
豊富に用意されています。



3-2. 簡単なネイティブアプリの開発

• ネイティブアプリ開発手順5

Actionの実装



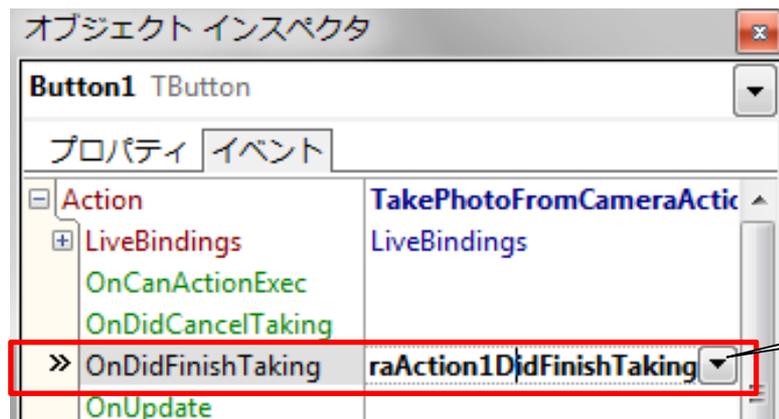
ButtonのActionプロパティで
標準アクションの新規追加

TTakePhotoFromCameraAction
を選択

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順5

Actionのイベントにプログラムを実装



OnDidFinishTakingイベントを作成

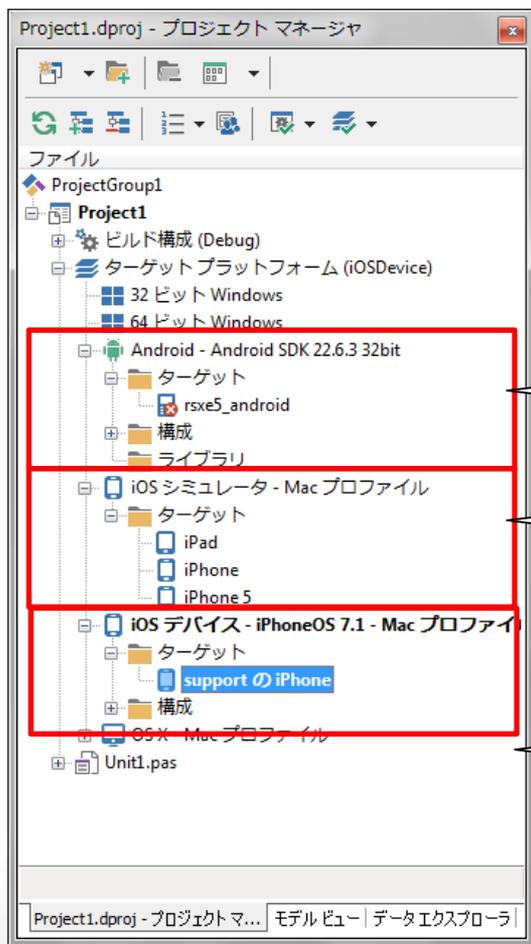
OnDidFinishTaking処理(カメラ撮影終了処理)

```
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);  
begin  
    Image1.Bitmap.Assign(Image);  
end;
```

3-2. 簡単なネイティブアプリの開発

• ネイティブアプリ開発手順6

iOS実機向けにコンパイル



Android向けのコンパイル

iOSシミュレータ向けのコンパイル

iOS実機向けコンパイル

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順7

iOSアプリケーションの実行

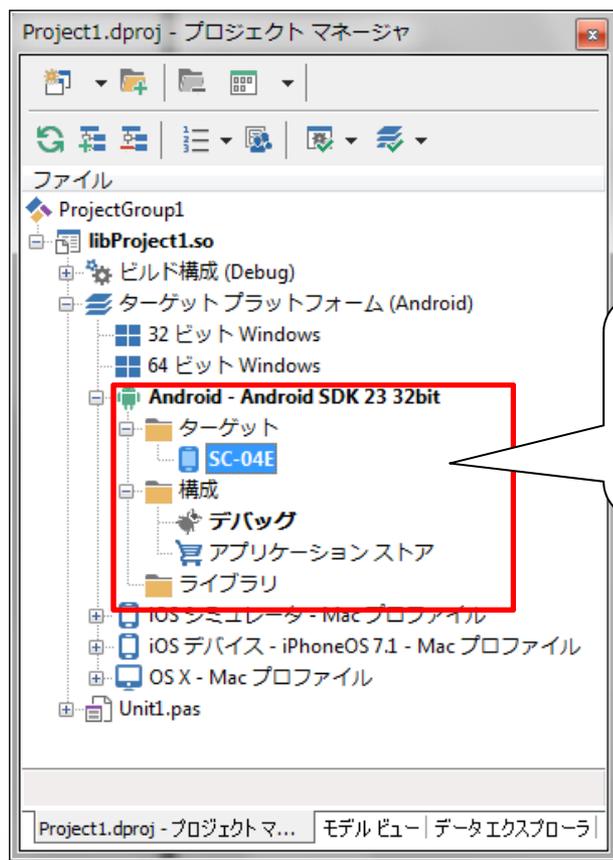


スマートデバイスの機能を使えば、カメラ撮影を連携したアプリケーションもPCやWebのアプリケーションと比べて、簡単に実現ができます。

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順8

Android実機向けにコンパイル



同じプログラムを
Android実機向けに
コンパイル

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順9

Androidアプリケーションの実行



1つのプログラムからiOS、Androidのネイティブアプリケーションを開発できます。

3-2. 簡単なネイティブアプリの開発

- 補足:iOSとAndroidの違い1

ハードウェアキーの違い

Androidには「戻るボタン」や「メニューボタン」が物理的に存在しますが、iOSには「ホームボタン」しかありません。

例えばiOSで「戻るボタン」が前提のアプリを作成してしまうと意図した画面遷移操作が行えなくなります。そのため、OS・ハードの違いを把握した画面設計は非常に重要となってきます。



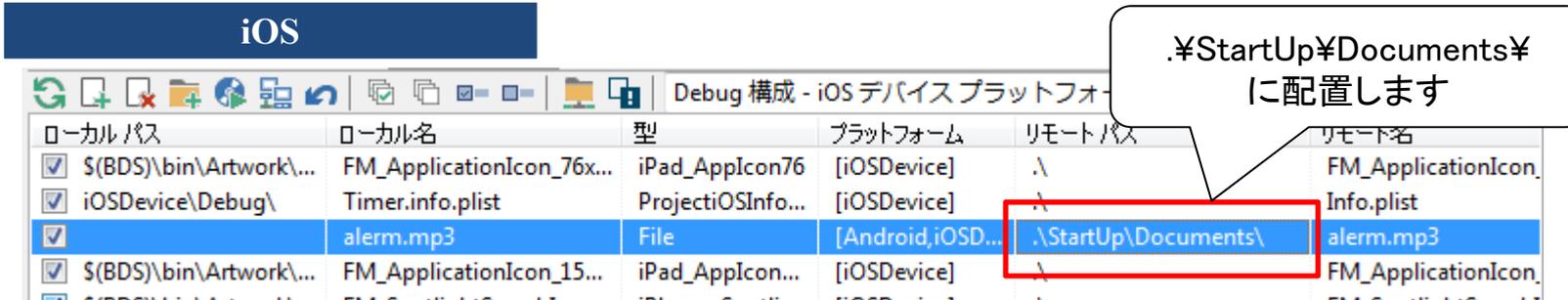
3-2. 簡単なネイティブアプリの開発

- 補足: iOSとAndroidの違い2

ファイル配置の違い(プロジェクト|配置から設定)

音源ファイルや動画ファイルなど、アプリケーション内で固有で持ちたい場合、配置(保存)先のパスはプラットフォームによって異なります。

iOS



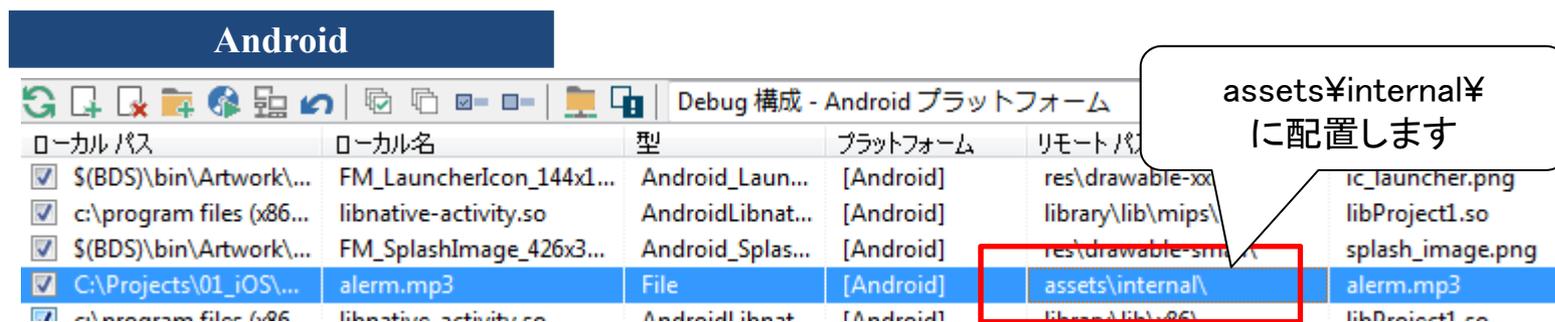
ローカルパス	ローカル名	型	プラットフォーム	リモートパス	リモート名
\$(BDS)\bin\Artwork\...	FM_ApplicationIcon_76x...	iPad_AppIcon76	[iOSDevice]	.\	FM_ApplicationIcon...
iOSDevice\Debug\	Timer.info.plist	ProjectiOSInfo...	[iOSDevice]	.\	Info.plist
alarm.mp3	alarm.mp3	File	[Android,iOSD...	.\Startup\Documents\	alarm.mp3
\$(BDS)\bin\Artwork\...	FM_ApplicationIcon_15...	iPad_AppIcon...	[iOSDevice]	.\	FM_ApplicationIcon...
\$(BDS)\bin\Artwork\...	FM_SpotlightSearchIcon	iPhone_Spotli...	[iOSDevice]	.\	FM_SpotlightSearchI...

プログラム上でのネイティブファイルパス指定例(iOS)

GetHomePath + PathDelim + ' Documents ' + PathDelim + ' ファイル名 '

3-2. 簡単なネイティブアプリの開発

- 補足: iOSとAndroidの違い2
ファイル配置の違い(プロジェクト|配置から設定)



プログラム上でのネイティブファイルパス指定例(Android)

`TPath.GetDocumentsPath + 'ファイル名'`

補足) SDカードなど外部ストレージで扱う場合

`TPath.GetSharedDocumentsPath + 'ファイル名'`

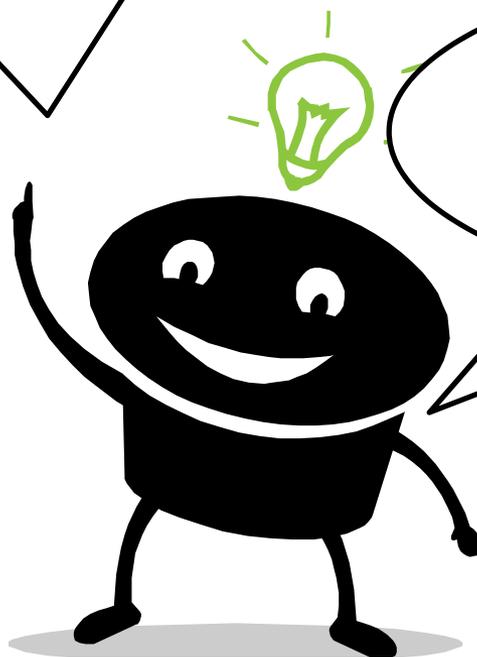


外部ストレージに保存しているファイルは
PCとのUSB転送などで便利です

3-3. DBに接続するネイティブアプリの開発

Delphiで簡単に
スマホアプリが作れた！

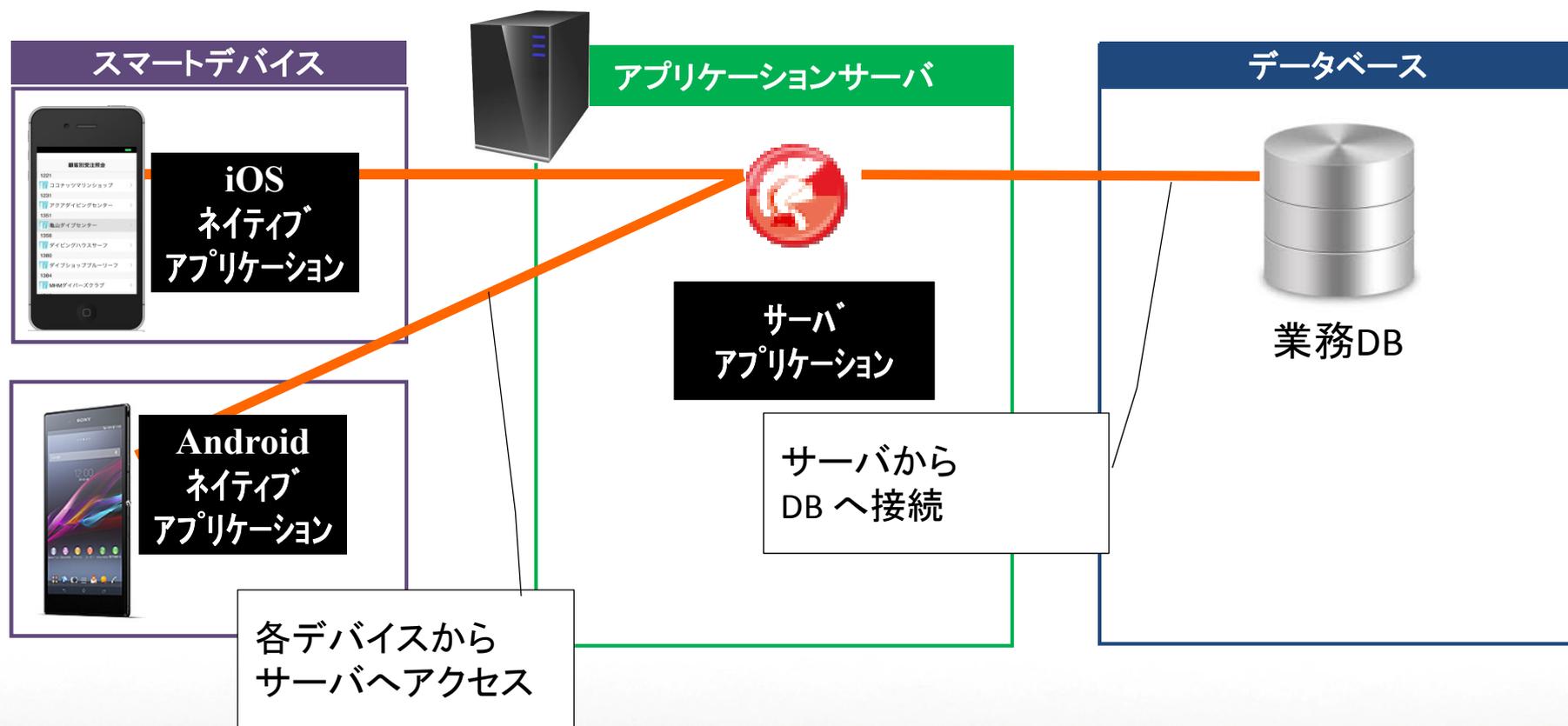
でも業務アプリは
社内のDBを使いたい！



3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリの仕組み

ネイティブアプリケーションからデータベースに接続する仕組みは、Webアプリケーションに近い、サーバを経由した3階層方式になります。



3-3. DBに接続するネイティブアプリの開発

- サーバアプリケーションとは？

アプリケーションサーバ(中間サーバ)からデータベースに接続、処理を行うアプリケーションです。

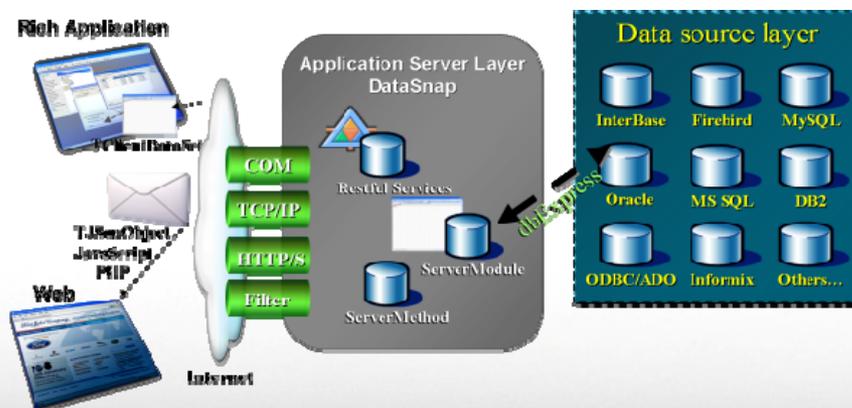
ネイティブアプリケーションは、サーバアプリケーションを経由してデータベースにアクセスすることができます。

Delphiで開発するサーバアプリケーション

Delphiではサーバアプリケーションを『DataSnap』で簡単に開発できます。

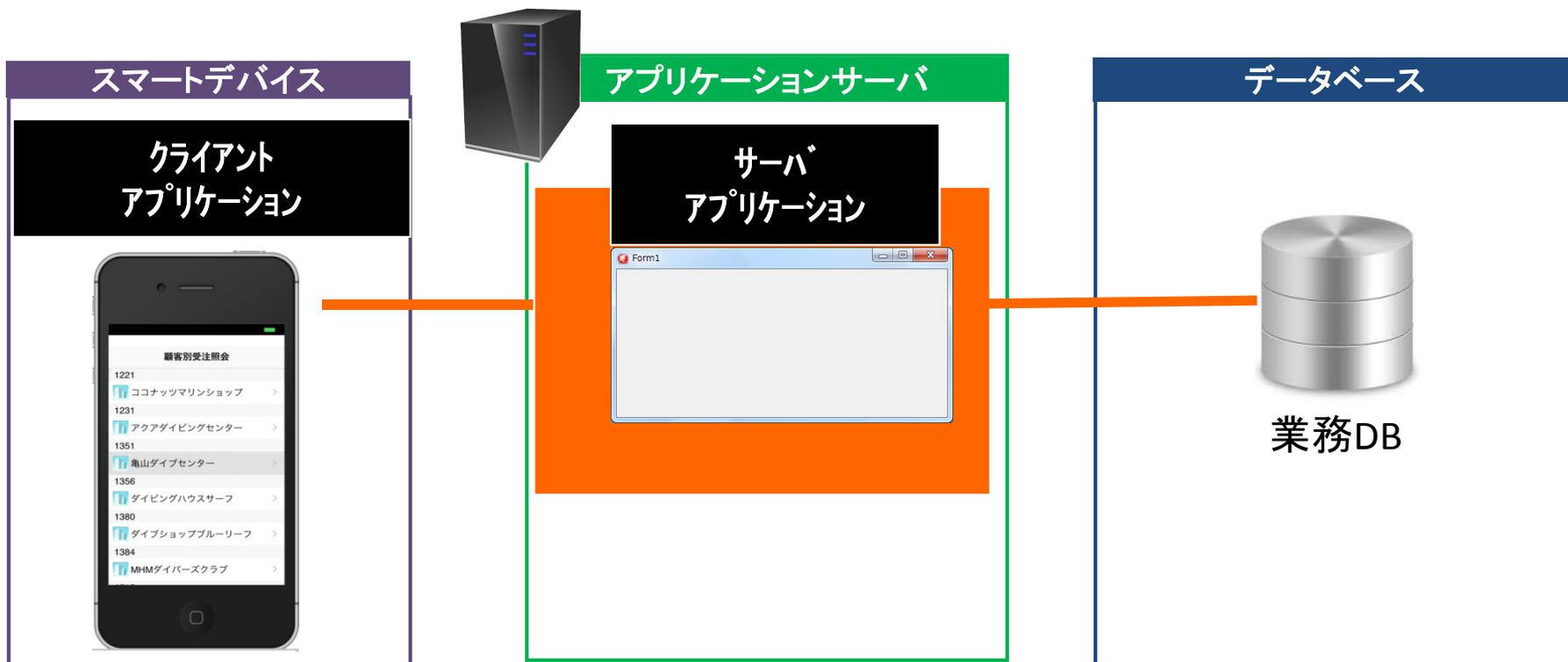
『DataSnap』はサーバアプリケーション専用の開発機能です。

サーバアプリケーションは、FireDac、dbExpress等のDBコンポーネントを設定したり、関数をプログラミングすることで機能を実装できます。



3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発

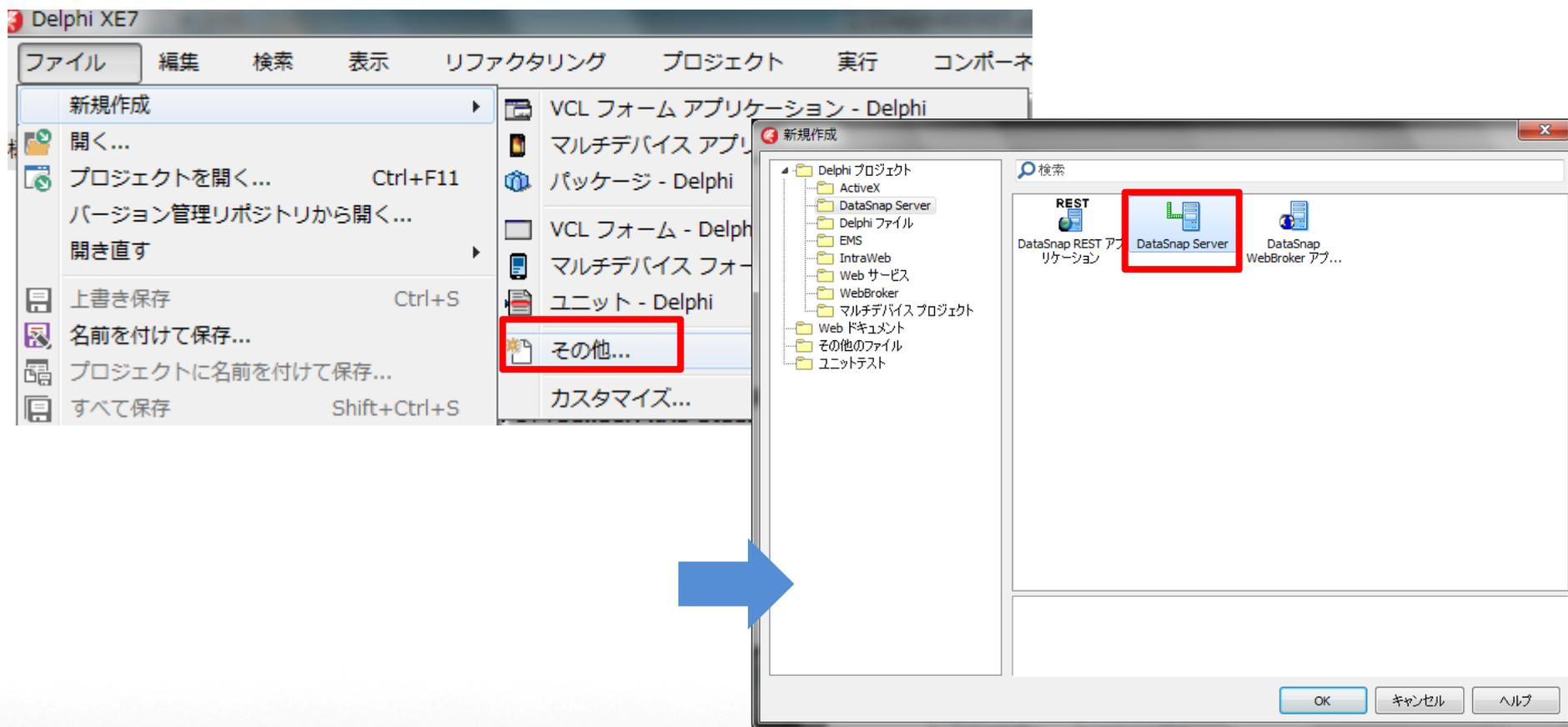


3-3. DBに接続するネイティブアプリの開発

• サーバアプリの開発手順1



メニューの[ファイル|新規作成|その他]から『DataSnapServer』を選択



3-3. DBに接続するネイティブアプリの開発

• サーバアプリの開発手順2

DataSnapServerの形式を選択



サーバ上でどのように実行するかを選択します。
サーバに常駐で起動させる場合はサービスアプリケーションを使います。

VCL、FireMonkeyどちらでも開発可能です。

3-3. DBに接続するネイティブアプリの開発

• サーバアプリの開発手順3

通信プロトコルを設定



接続通信の設定
標準はTCP/IPを使います。
HTTP経由での通信も可能です。

3-3. DBに接続するネイティブアプリの開発

• サーバアプリの開発手順4

使用するポート番号を指定

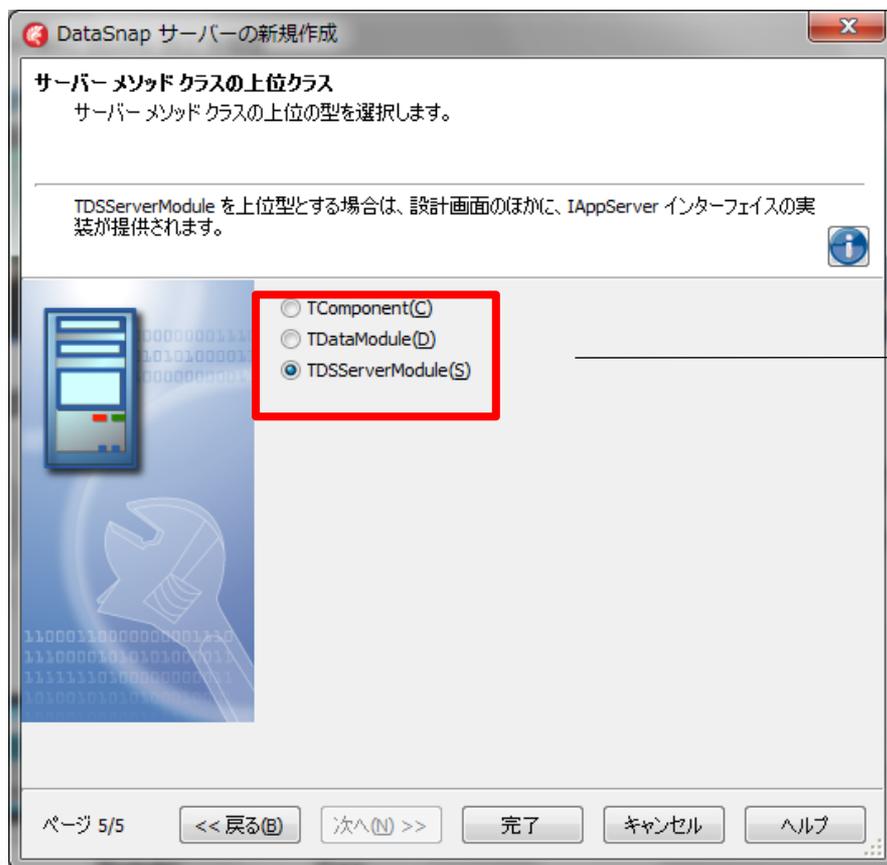


サーバ接続に使用するポート番号を指定します。(デフォルト211)

3-3. DBに接続するネイティブアプリの開発

• サーバアプリの開発手順5

サーバーメソッドの上位クラスを選択

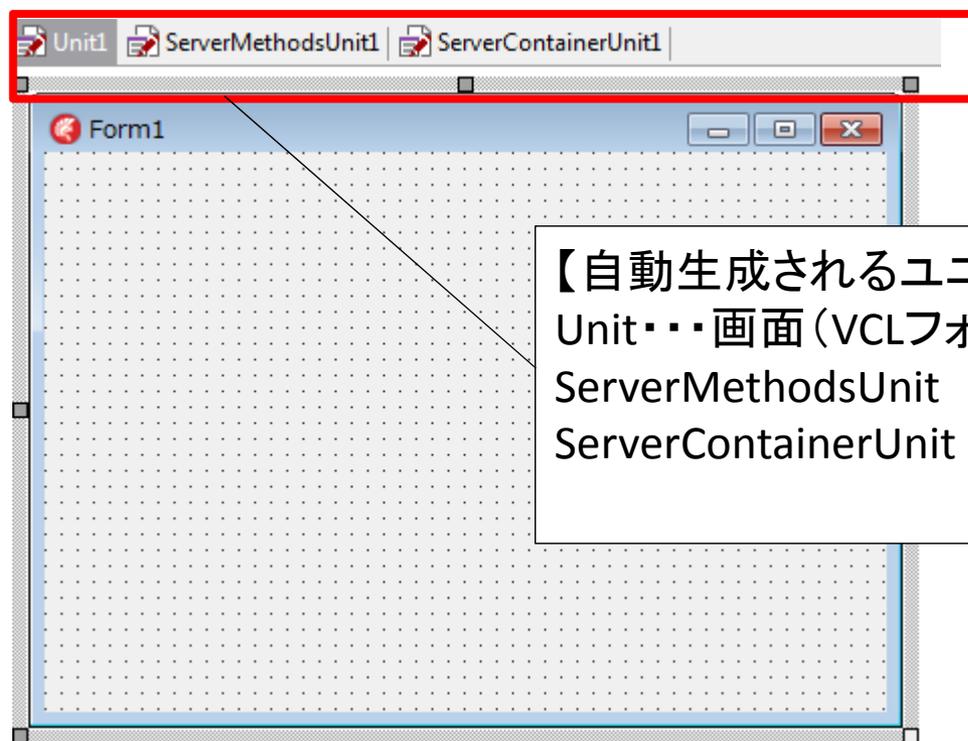


データセットを公開する場合は TDSServerModule を選択します。

3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順7

選択した構成によってモジュールが生成



【自動生成されるユニット】

Unit・・・画面（VCLフォームアプリケーション時のみ）

ServerMethodsUnit・・・サーバが提供する機能

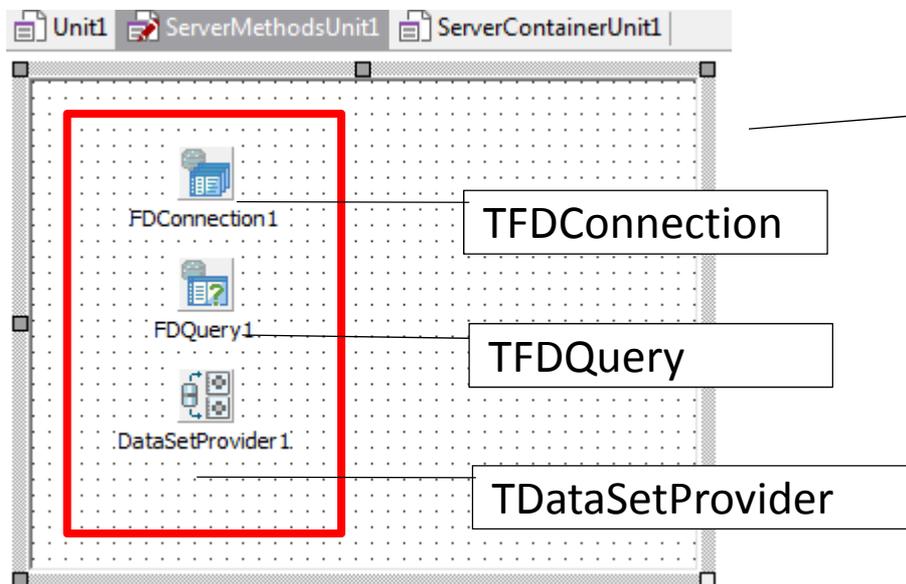
ServerContainerUnit・・・通信を制御する機能

3-3. DBに接続するネイティブアプリの開発

• サーバアプリの開発手順8

ServerMethodsに次のコンポーネントを配置

TFDConnection、TFDQuery、TDataSetProvider



今回はInterBaseへSQL実行するサーバアプリケーションの機能だけ実装しています。
(dbExpressでも可能です)

3-3. DBに接続するネイティブアプリの開発

• サーバアプリの開発手順9

TFDConnectionコンポーネントの設定

右クリックから接続エディタを起動して、DB接続情報を設定

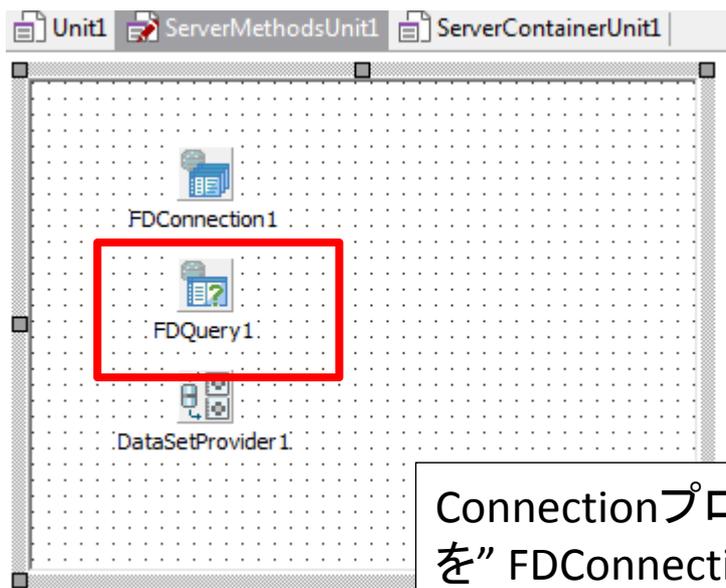
loginpromptプロパティはFalseに設定

パラメータ	値	デフォルト
DriverID	IB	IB
Pooled	False	False
Database	localhost:C:\ProgramData#Embarcadero#Inte	
User_Name	sysdba	
Password	masterkey	
MonitorBy		
OSAuthent		
Protocol	Local	Local
Server		
Port		
SQLDialect	3	3
RoleName		
CharacterSet	NONE	NONE
ExtendedMetadata	False	False

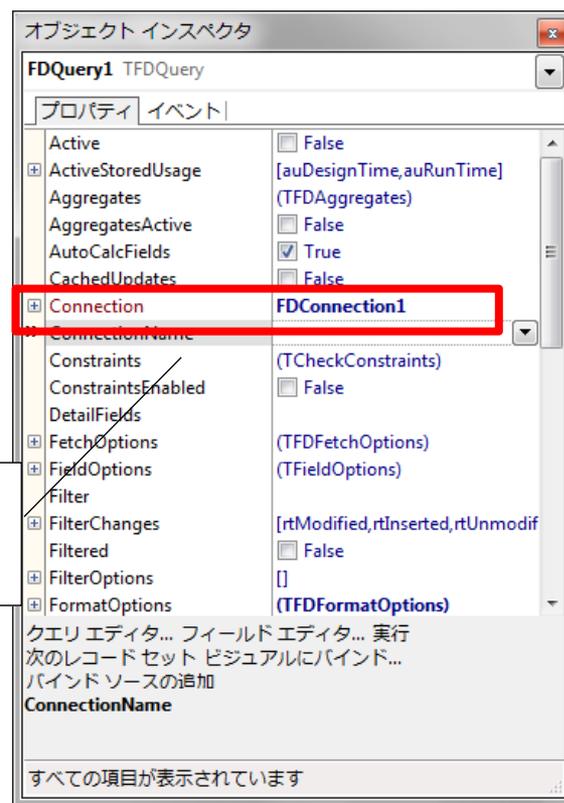
3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順10

TFDQueryコンポーネントの設定



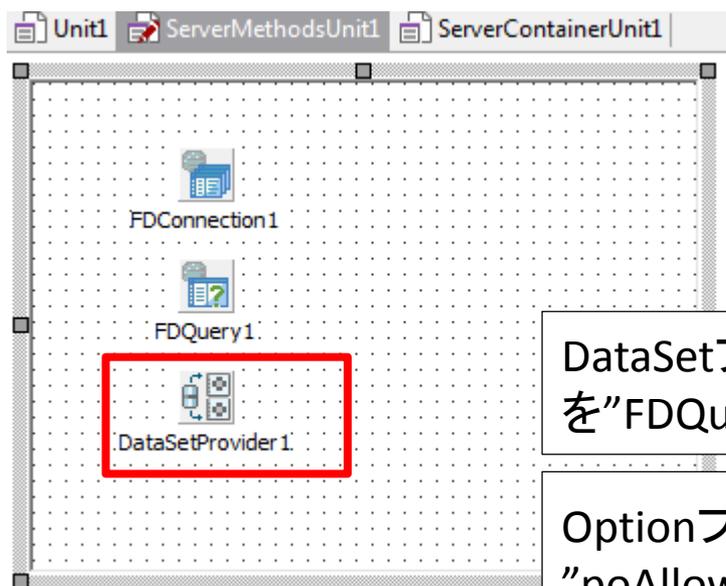
Connectionプロパティ
を” FDConnection1”に設定



3-3. DBに接続するネイティブアプリの開発

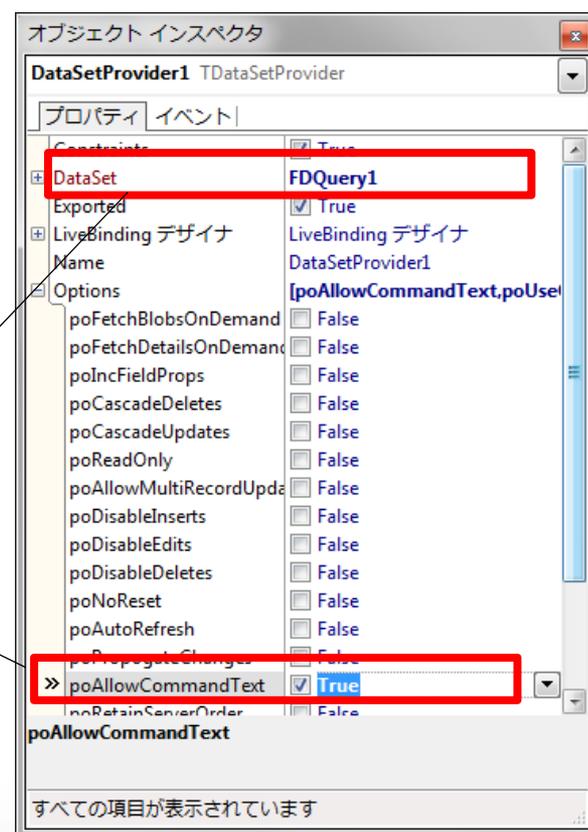
• サーバアプリの開発手順11

TDataSetProviderコンポーネントの設定



DataSetプロパティ
を"FDQuery1"に設定

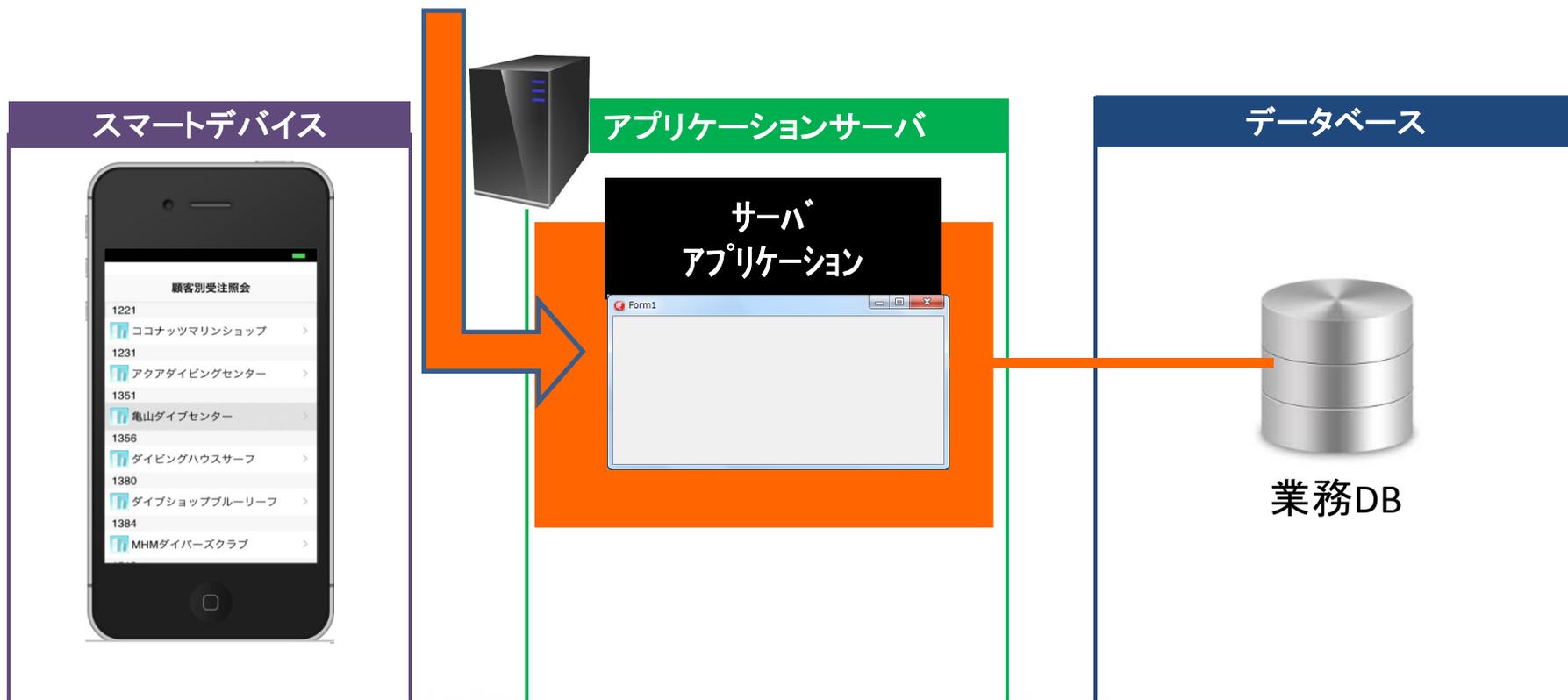
Optionプロパティの
"poAllowCommandText"を
"True"に設定



3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順12

完成したらコンパイルして、アプリケーションサーバ上で起動しておきます。



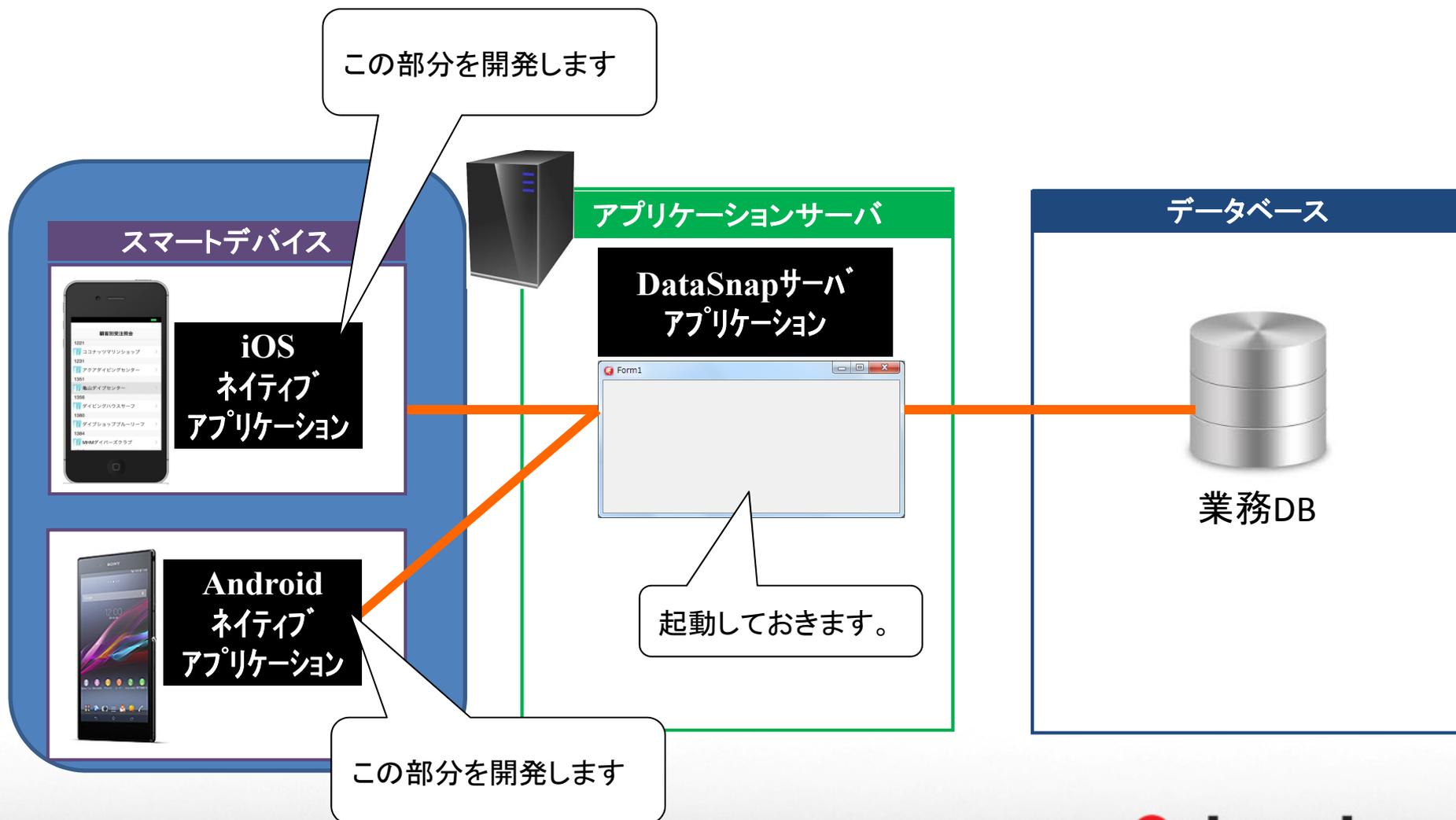
3-3. DBに接続するネイティブアプリの開発

- デモで開発するネイティブアプリケーション



3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ



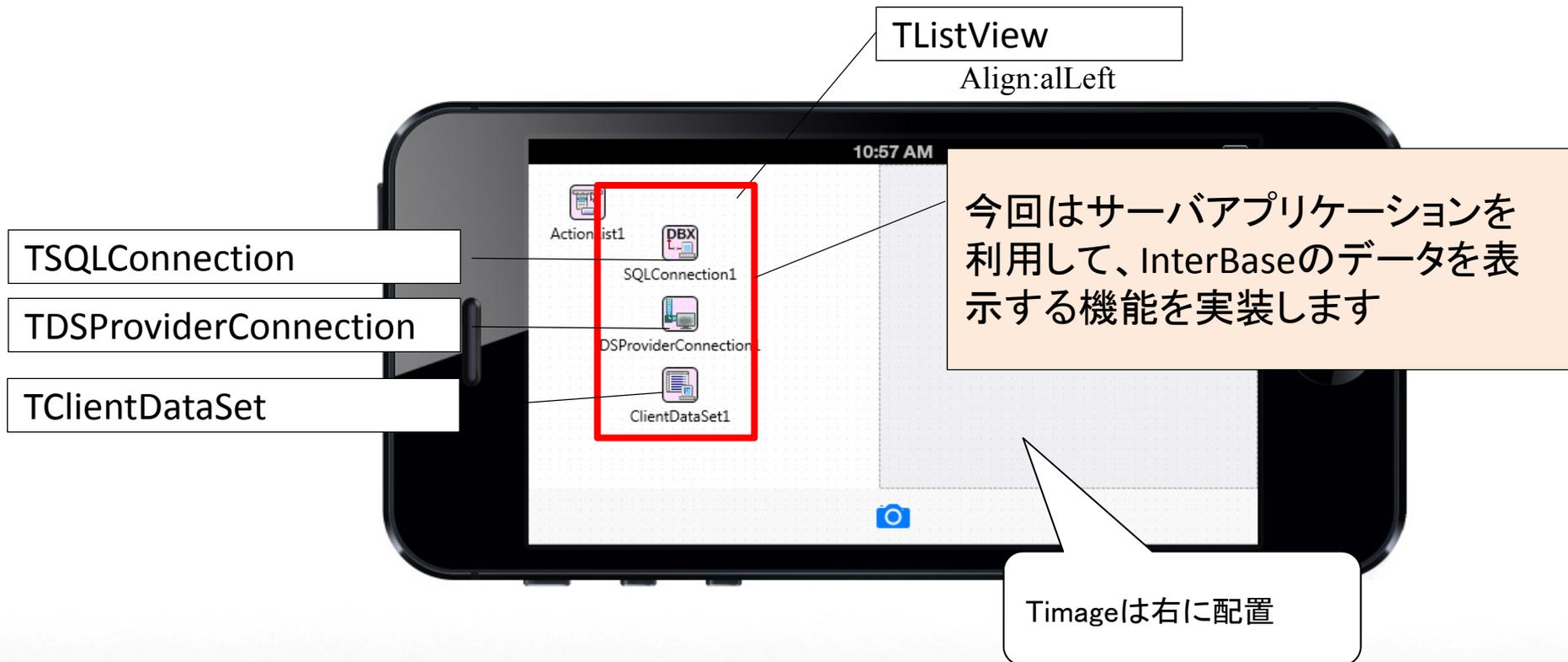
3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順1



先ほど3-2.で作成した写真撮影のアプリをカスタマイズします。

TListView、TSQLConnection、TDSProviderConnection、TClientDataSetを追加配置

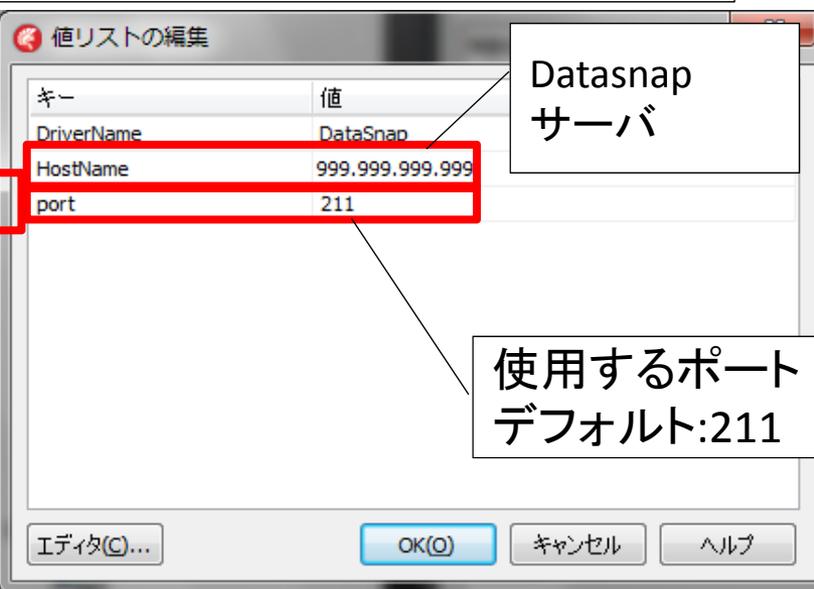


3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順2

SQLConnectionコンポーネントの設定

ConnectionStringプロパティを
“DATASNAPCONNECTION”に設定



Datasnap
サーバ

使用するポート
デフォルト:211

LoginPromptプロパティ
を“False”に設定

Paramsプロパティ
を設定

3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順3

DSPProviderConnectionコンポーネントの設定

ServerClassNameプロパティ
を” TServerMethods1”に入力設定

SQLConnectionプロパティ
を” SQLConnection1”に設定

すべての項目が表示されています

3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順4

ClientDataSetコンポーネントの設定

The screenshot shows the Delphi IDE with the ClientDataSet1 component selected in the Object Inspector. The Properties window is open, showing the CommandText property set to "SELECT * FROM PRODUCT". The RemoteServer property is set to "DSPProviderConnection1". The ProviderName property is set to "DataSetProvider1".

②RemoteServerを設定していると
ProviderNameプロパティが選択できるの
で"DataSetProvider1"を設定
サーバアプリケーションが起動している必要があります。

CommandTextプロパティ
に実行するSQLを設定
例)SELECT * FROM PRODUCT

①RemoteServerプロパティに
"DSPProviderConnection1"を設定

設定ができれば「すべてのフィールドの追加」で取り込み

3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順5

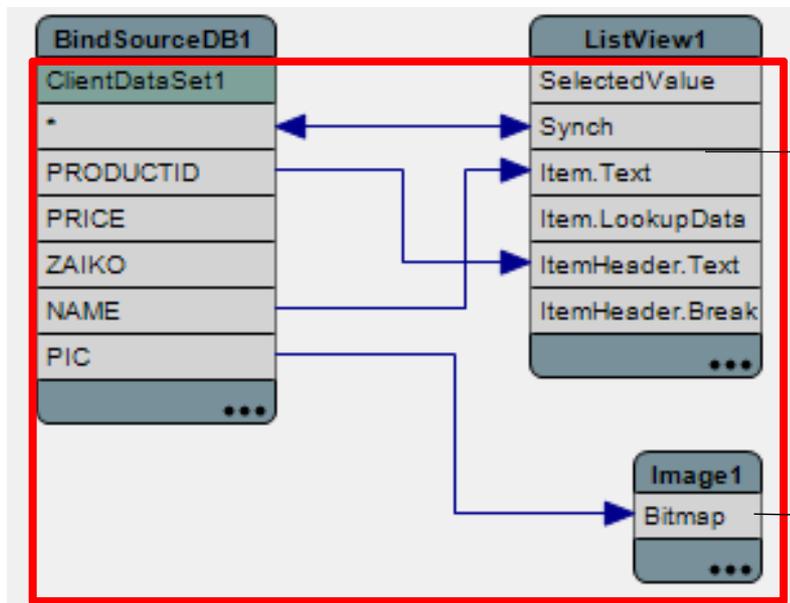
データ表示をライブバインディング機能で実装

The screenshot shows the Delphi IDE interface for a project named 'Project1 - Delphi XE7 - Unit1 [ビルド完了]'. The main window displays a visual design for a form with components like 'SQLConnection1', 'DSProviderConnection1', and 'ClientDataSet1'. A context menu is open over the 'ClientDataSet1' component, with the '位置(Z)' (Position) option highlighted in red. A callout box points to this menu with the text: '右クリックから「ビジュアルにバインド」を起動' (Start 'Visual Bind' from right-click). Another callout box points to the design view with the text: '「ライブバインディング」の設計画面' (Design view of 'Live Binding'). The design view shows a 'LiveBinding デザイン' (LiveBinding Design) diagram with a red border, illustrating the data flow from 'BindSourceDB1' through 'ClientDataSet1' to 'ListView1', 'Image1', and 'ToolBar1'. The 'ClientDataSet1' component is expanded to show fields like 'PRODUCTID', 'PRICE', 'ZAIKO', 'NAME', and 'PIC'. The 'Object Inspector' on the right shows the properties of the selected 'Button1' component, with the 'Action' property set to 'TakePhotoFromC...'. The 'Object Inspector' also shows the 'LiveBindings' section, which is currently empty.

3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順6

TClientDataSetの項目をTListViewとTImageに
ドラッグ&ドロップでリンク



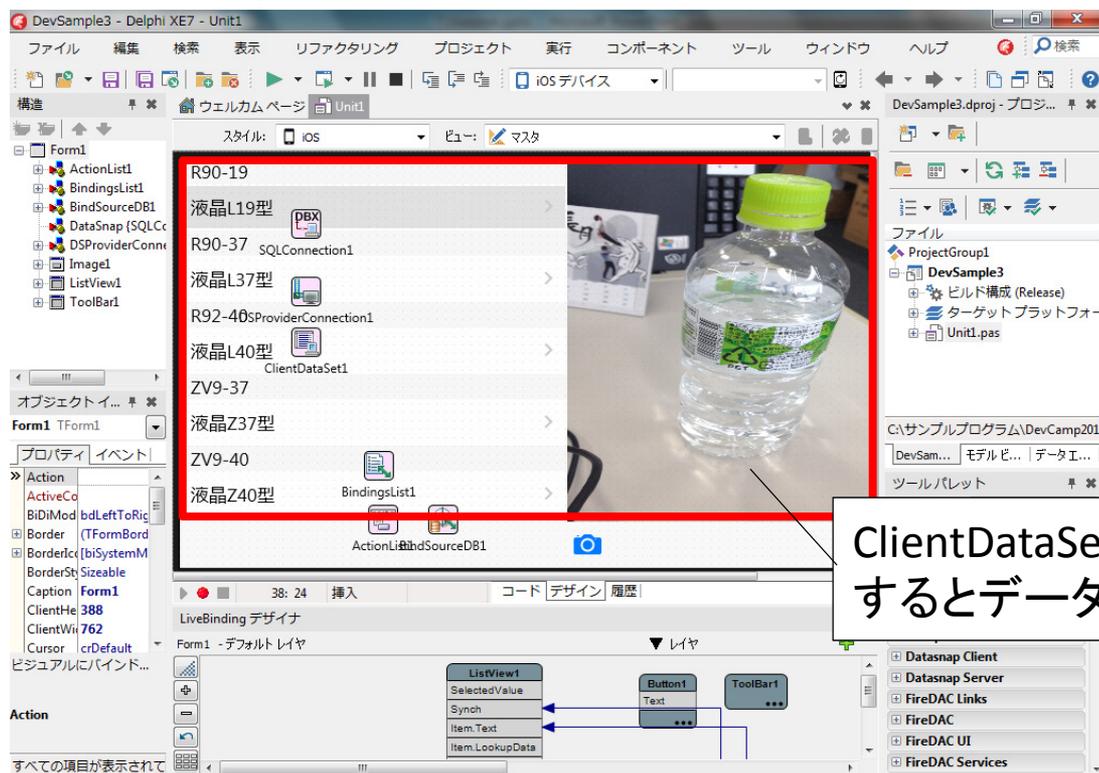
<ListView1>...ヘッダーにID、テキストに名称
*をSynchにリンクして同期
PRODUCTIDをItemHeaderTextにリンク
NAMEをItemTextにリンク

<Image1>...画像フィールド
PICをBitmapにリンク

3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順7

データ表示の確認



3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順8

OnDidFinishTakingイベントをDB更新に若干変更

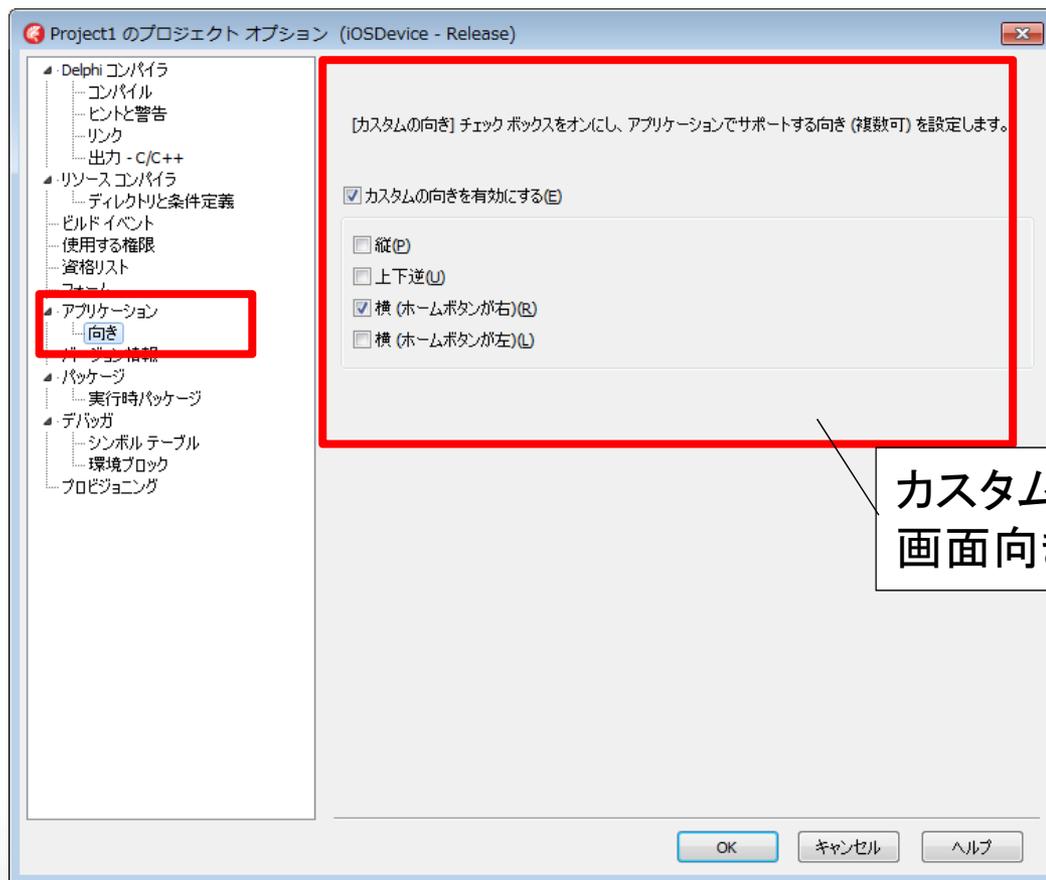
OnDidFinishTaking処理(カメラ撮影終了処理)

```
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
begin
  ClientDataSet1.Edit;
  ClientDataSet1.FieldByName('PIC').Assign(Image);
  ClientDataSet1.Post;
  ClientDataSet1.ApplyUpdates(-1); //元DBへ反映
  // Image1.Bitmap.Assign(Image);
end;
```

3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順9

[プロジェクト|オプション]のアプリケーションで向きを固定

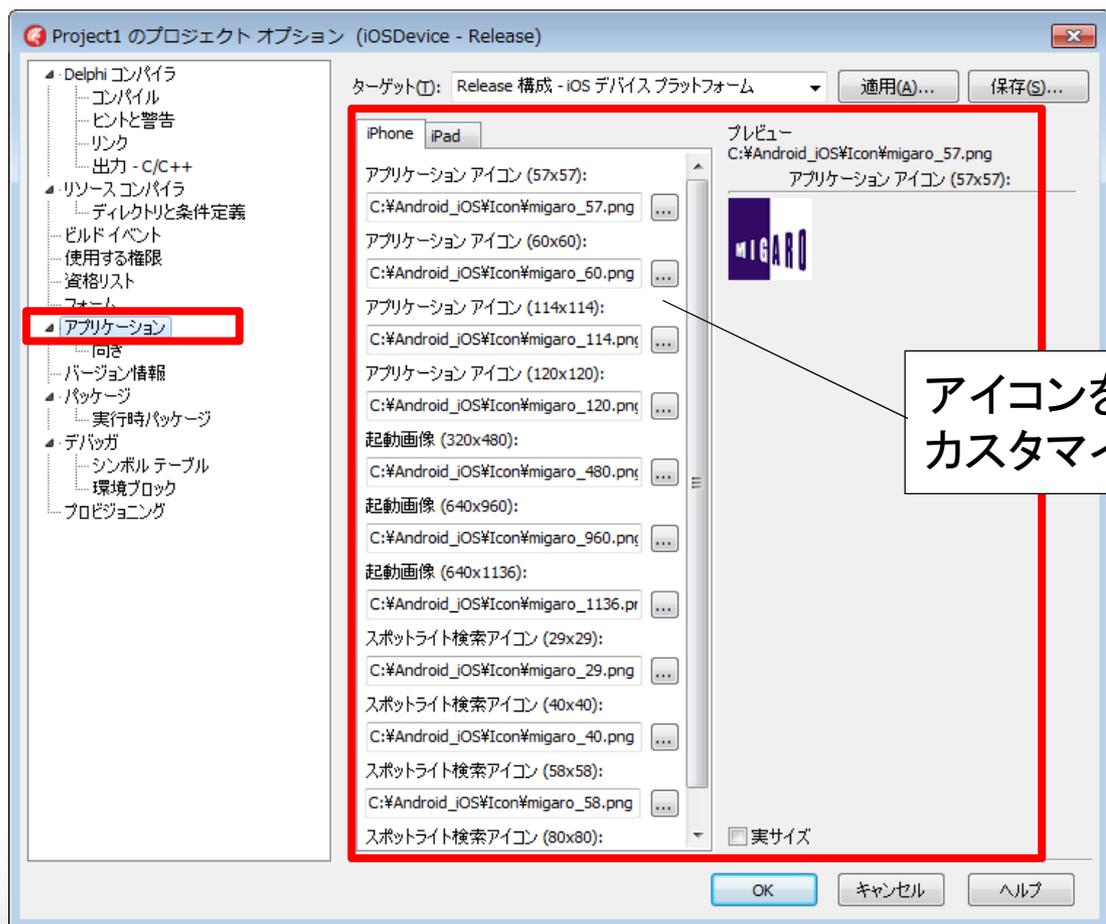


カスタム設定で
画面向きを固定できる

3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順10

[プロジェクト|オプション]のアプリケーションでアイコンを設定

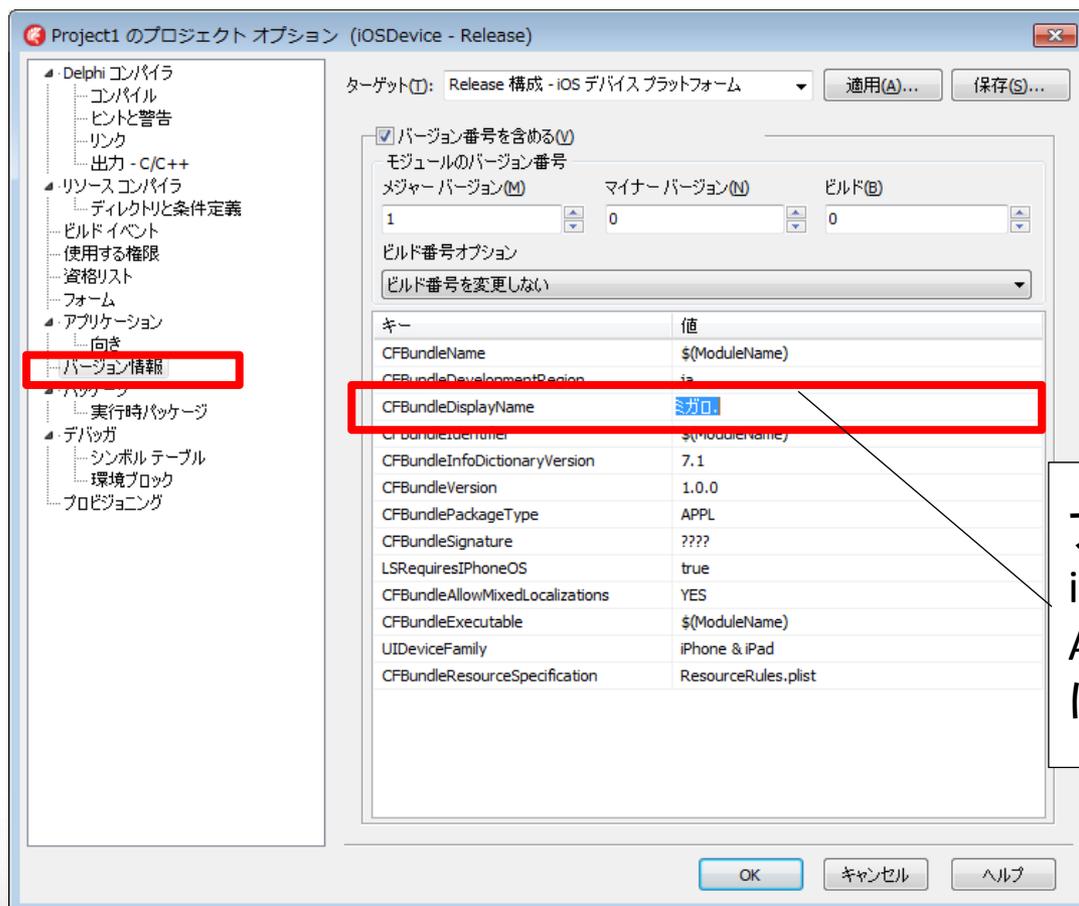


アイコンを用意しておけば
カスタマイズ可能

3-3. DBに接続するネイティブアプリの開発

• 業務DBに接続するネイティブアプリ開発手順11

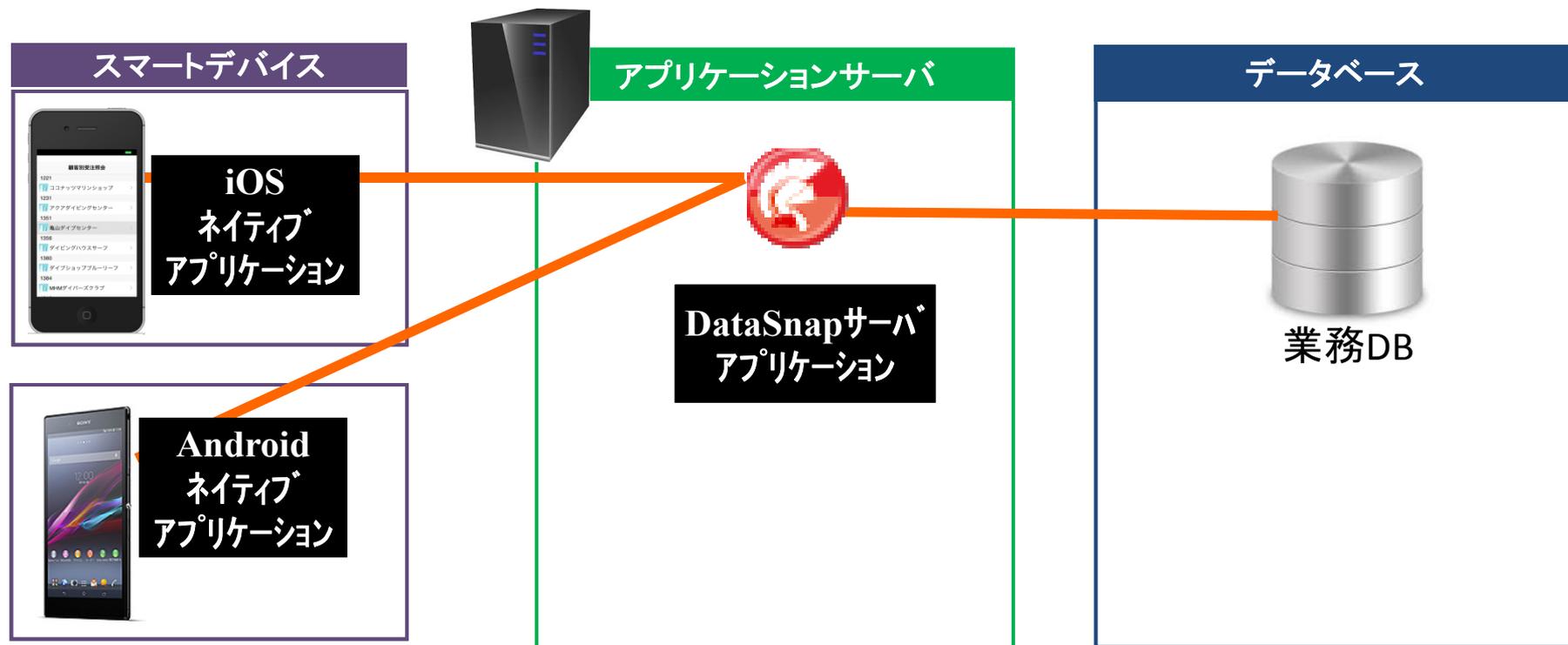
[プロジェクト|オプション]のバージョン情報設定



アプリのアイコン表示名は
iOS:CFBundleDisplayName
Android:Label
に設定

3-3. DBに接続するネイティブアプリの開発

- 3階層によってスマートデバイスからDBへ接続完成



本資料では設定内容を全て記載して页数多いですが、プロパティ設定をしていくだけなので簡単！
常時接続が安定しない環境であれば、都度接続処理が◎

3-4. ネイティブアプリケーションの配布

- ネイティブアプリはスマートデバイスに配布・インストールする方法が2つあります。

社内公開での配布

開発者はWebサーバ上にネイティブアプリのファイルを公開して配布します。

一般公開での配布

開発者はiOSであればAppStore、AndroidであればPlayStoreにネイティブアプリを公開して配布します。

3-4. ネイティブアプリケーションの配布

- 社内公開と一般公開の配布方法の違い

社内公開での配布

開発

Webサーバ公開

iOSの場合、
iOS Developer Program
の種類によって制限あり。
EnterpriseのIn-House利用が◎



iOSアプリ開発
(社内公開)



Webサーバ



Androidアプリ開発
(社内公開)

ユーザー利用



iOSユーザー



Androidユーザー

一般公開での配布

ストア公開

開発



AppStore



iOSアプリ開発
(一般公開)



PlayStore



Androidアプリ開発
(一般公開)

各ストア公開には
審査があります

3-4. ネイティブアプリケーションの配布

- 社内公開と一般公開のメリット/デメリット

	社内公開	一般公開
メリット	<ul style="list-style-type: none">社内だけで配布・利用できる。審査がないため、社内専用のアプリケーションが開発できる。	<ul style="list-style-type: none">ストアで公開するため、どこからでもすぐにインストールして利用できる。
デメリット	<ul style="list-style-type: none">Webサーバ等を用意して、配布環境の構築・運用が必要。	<ul style="list-style-type: none">誰でも利用できてしまう。公開には審査が必要。 (自社用アプリの公開は難しい)

3-4. ネイティブアプリケーションの配布

- 配布

社内公開するアプリは、配布用にコンパイルしたファイルをWebサーバ等に配置して、リンクでダウンロードできるように準備します。

- 配布Webサーバ

関連する拡張子はMIMEタイプを事前登録しておきます。

<iOS>

ipaファイル.....application/octet-stream

plistファイル.....text/xml

<android>

apkファイル.....application/vnd.android.package-archive

iOS7.1からplistファイルの配布に[https\(SSL\)](#)が必須です。
Dropbox等で代用もできます。

3-4. ネイティブアプリケーションの配布

- 配布ファイル

社内公開するアプリは、配布用にコンパイルしたファイルをWebサーバ上に配置して、リンクでダウンロードできるように準備。

【Android配布用のファイル】

apkファイル



ストアアプリではないので、
[提供元不明のアプリ]の許可が必要

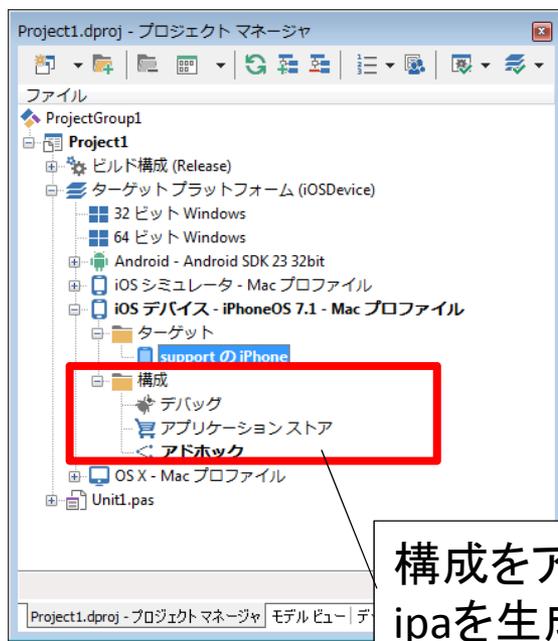
ダウンロード用HTML例(Android)

```
<h1>Androidダウンロードサイトサンプル</h1>
<form>
  <a href="./ Project1.apk" type="application/vnd.android.package-archive">
    アプリケーションダウンロード</a><br>
</form>
```

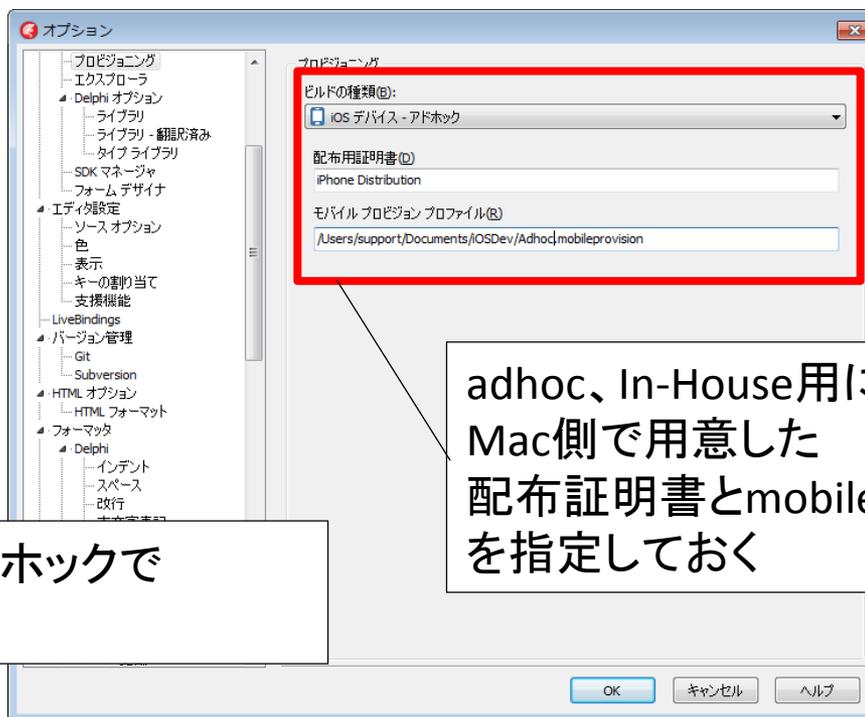
3-4. ネイティブアプリケーションの配布

【iOS配布用のファイル】

iOS: ipa、plistファイル



構成をアドホックで
ipaを生成



adhoc、In-House用に
Mac側で用意した
配布証明書とmobileprovision
を指定しておく

配布証明書やmobileprovisionファイルは適切に設定できていないとエラーになるので、Mac環境でキーチェーンやXcode上の認証をしっかりと確認が必要。

3-4. ネイティブアプリケーションの配布

ダウンロード用plist例

```
<key>assets</key>
<array>
  <dict>
    <key>kind</key>
    <string>software-package</string>
    <key>url</key>
    <string>https://Webサーバ /Project1.ipa</string>
  </dict>
</array>
```

ipaを配置したurlを指定

ダウンロード用HTML例(iOS)

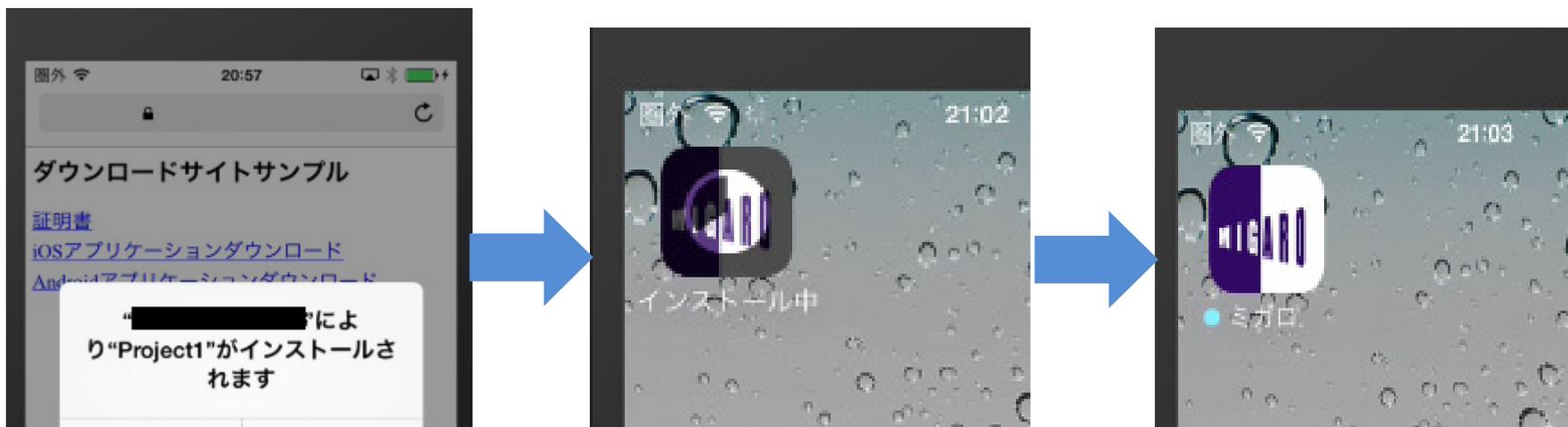
```
<h1>iOSダウンロードサイトサンプル</h1>
<form>
  <a href="itms-services://?action=download-manifest&url=https://Webサーバ /Project1.plist">
    アプリケーションダウンロード</a><br>
</form>
```

3-4. ネイティブアプリケーションの配布

- インストール

配布Webサーバにアクセスしてリンクからダウンロード&インストール。

例) iOS7.1



Androidも同じ (SSL不要)





まとめ

4.まとめ

- 企業でのスマートデバイス導入も増えてきてスマートデバイス用の業務アプリケーション需要も高い
- Delphiなら主流のiOSとAndroidの開発が簡単
- 業務アプリケーションの社内DB接続はDataSnapサーバを使った3階層方式
- 配布は社内公開と一般公開で方法が異なるのでしっかり環境想定・準備が必要

ご静聴ありがとうございました