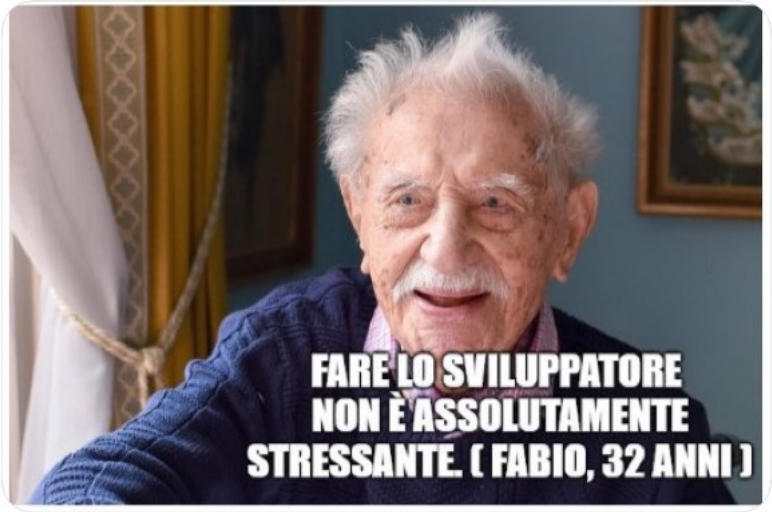
Push Notifications

Delphi-Firemonkey OneSignal e Firebase

wintech italia

embarcadero
reseller partner

😁 "Being a developer is no stressing at all" Fabio, 32 😁



DelphiDay
italian conference
PIACENZA EDITION

2019 - XVIII Edizione

wintech italia

Marco Mottadelli

Dna Software Sas

Linguaggi: Delphi, Html5, Javascript ExtJS

Ambiti: Retail, Healthcare, MobileApp

mail: dna@dnasoftware.it

website: www.dnasoftware.it

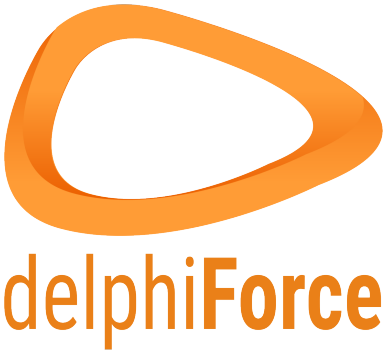
twitter: @mottadelli75

DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

wintech italia



mail: info@delphiforce.it

website: www.delphiforce.it

DelphiForce was founded in July 2018 by Antonio Polito, Carlo Narcisi, Fabio Codebue, Marco Mottadelli, Maurizio del Magno, Omar Bossoni, Thomas Ranzetti

AGENDA

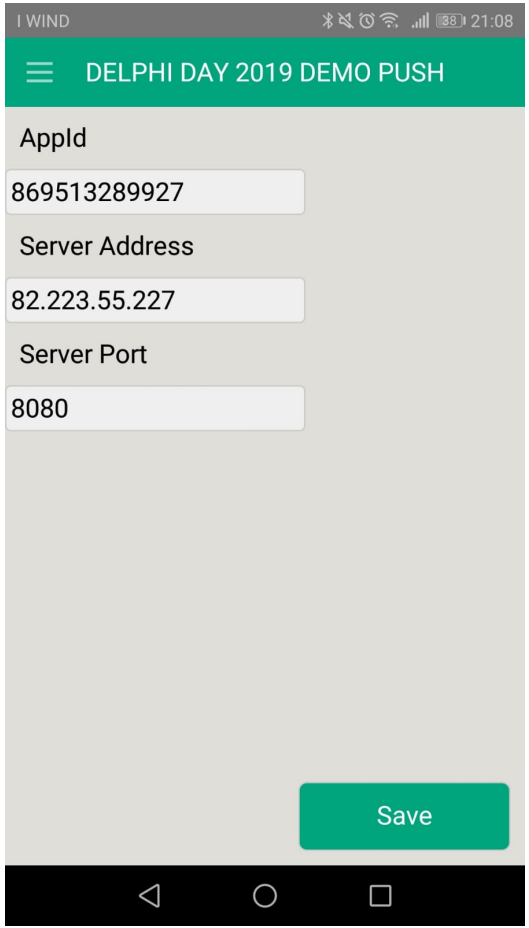
- ➔ COSA SONO LE PUSH NOTIFICATIONS
- ➔ VANTAGGI DELLE NOTIFICHE PUSH
- ➔ ARCHITETTURA DI UN SISTEMA DI GESTIONE DELLE PUSH
- ➔ SERVIZI APPLE PUSH NOTIFICATION E GOOGLE FIREBASE
- ➔ IL NOSTRO INTERMEDIARIO – ONESIGNAL
- ➔ DEMO TIME 1 – LA NOSTRA APP MOBILE CHE RICEVE LE PUSH
- ➔ DEMO TIME 2 - INTEGRAZIONE DIRETTA APP MOBILE CON FIREBASE
- ➔ DEMO TIME 3 - APPLICAZIONE VCL CHE INVIA LE PUSH CON FIREBASE

ROBOTTINO A CHI ?

SCARICHIAMO ED INSTALLIAMO APK DELLA DEMO

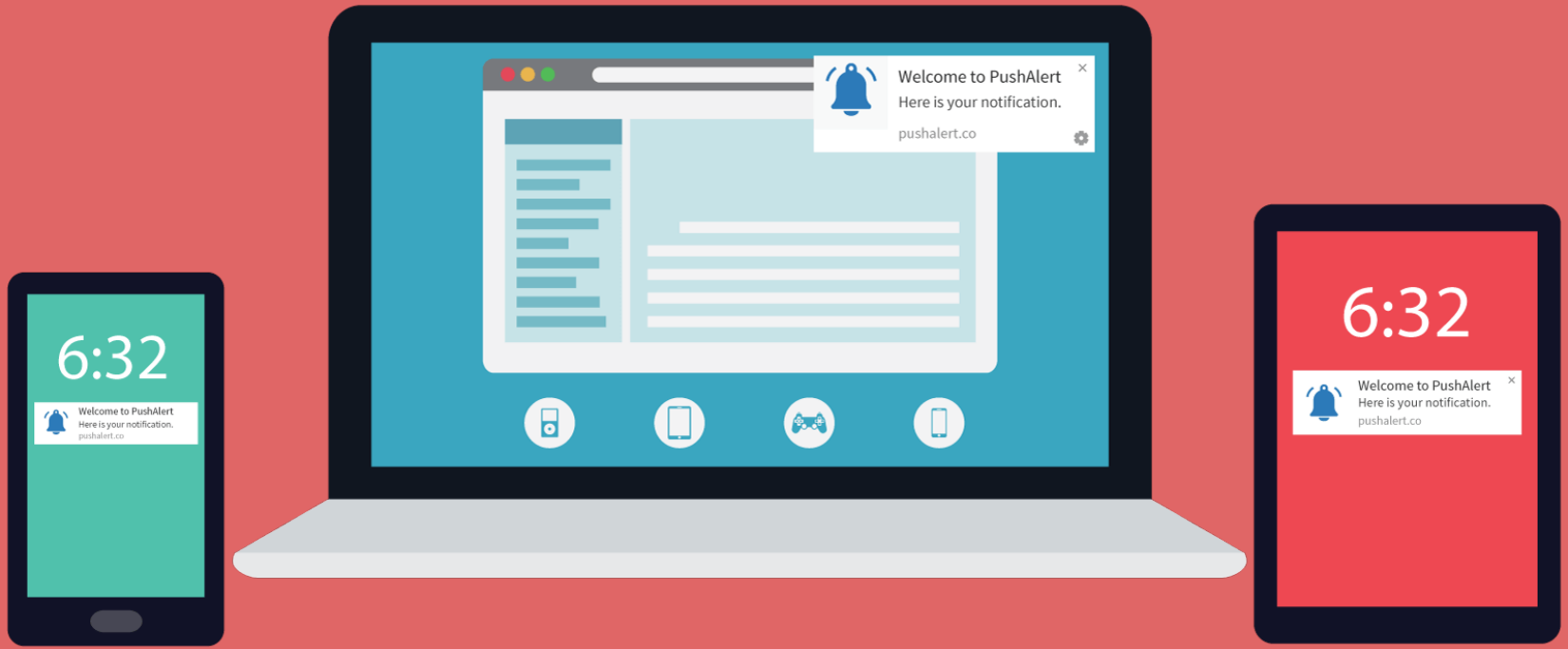
www.dnasoftware.it

PARAMETRI DA INSERIRE NELLA CONFIGURAZIONE



COSA SONO LE PUSH NOTIFICATIONS

ESEMPI DI NOTIFICHE



COSA SONO LE PUSH NOTIFICATIONS

Le notifiche push sono disponibili su tutti i sistemi operativi: iOS, Android, Windows Phone, BlackBerry

Le notifiche push sono una parte fondamentale di ogni strategia di **Mobile Marketing**.

Push Notification informativa o di aggiornamento
Una app può informare l'utente se ci sono state variazioni sul proprio volo, sulla prenotazione di un albergo

Informazioni che guidano l'utente a completare azioni virtuose
Una app può guidare a fare un acquisto, condividere un'azione sui social, completare l'iscrizione ad un evento

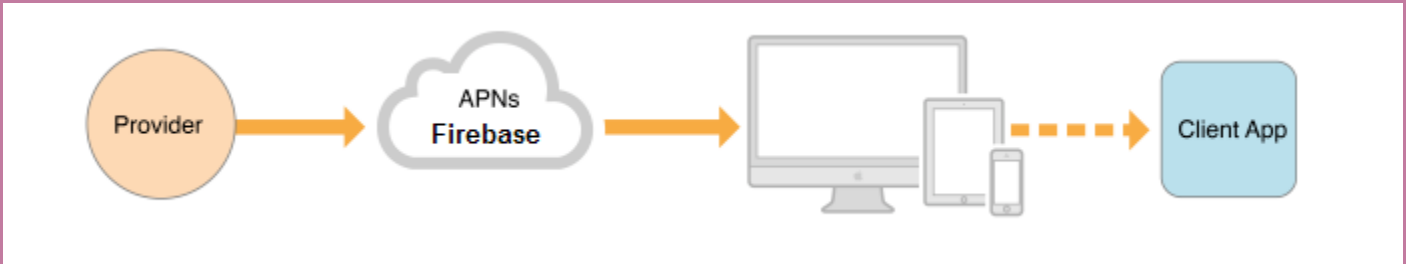
VANTAGGI DELLE NOTIFICHE PUSH

VANTAGGI DELLE NOTIFICHE

- ➔ Sono messaggi che arrivano ad un’app installata in uno specifico dispositivo
- ➔ Sono paragonabili a degli SMS ma utilizzano la connessione a Internet
- ➔ Sono quasi in tempo reale
- ➔ Non drenano la batteria
- ➔ Sono efficienti perché sono erogate direttamente dagli stessi sviluppatori del sistema operativo
- ➔ Sono attuali

ARCHITETTURA DI UN SISTEMA DI GESTIONE DELLE PUSH

ARCHITETTURA SISTEMA DI GESTIONE DELLE PUSH



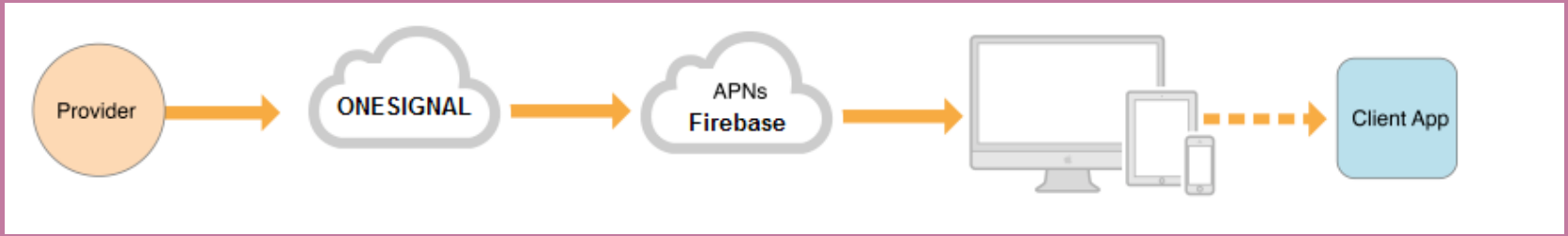
ARCHITETTURA DIRETTA

PROVIDER: è la nostra applicazione server che parla con FIREBASE e APN e che riceve i DEVICE TOKEN dai dispositivi.

APN e FIREBASE: sono i servizi messi a disposizione da GOOGLE e APPLE per l'invio delle notifiche

CLIENT APP: è la nostra applicazione mobile che riceve le notifiche

ARCHITETTURA SISTEMA DI GESTIONE DELLE PUSH



ARCHITETTURA TRAMITE INTERMEDIARIO

PROVIDER: è la nostra applicazione server che parla con ONESIGNAL e che riceve i DEVICE TOKEN dai dispositivi

ONESIGNAL: è l'intermediario che si occupa di parlare con APN e FIREBASE

APN e FIREBASE: sono i servizi messi a disposizione da GOOGLE e APPLE per l'invio delle notifiche

CLIENT APP: è la nostra applicazione mobile che riceve le notifiche

RESPONSABILITA' DEI VARI ELEMENTI

- PROVIDER – E' il nostro server REST
  - Riceve i device TOKEN dall'applicazione MOBILE
  - Comunica i device TOKEN a ONESIGNAL
  - Informa ONESIGNAL quando deve essere inviata una notifica

- ONESIGNAL – E' il nostro intermediario
  - Riceve i device TOKEN dal nostro server trasformandoli in «Players» (Users
  - Riceve dal nostro server l'informazione di inviare la push
  - Effettua la chiamata verso APN e FIREBASE per inviare la notifica al dispositivo MOBILE

- CLIENTAPP – E' la nostra applicazione mobile
  - Opportunamente configurata riceve il device TOKEN
  - Comunica i device TOKEN al nostro server
  - Si mette in ascolto per la ricezione della notifica

SERVIZIO DI APPLE PUSH NOTIFICATIONS (APN)

CONFIGURAZIONE PER APPLE

☑ Operazioni preliminari

- Iscrizione al developer program di APPLE
- Creazione di una nuova app
- Generazione dei provisioning profile
- Generazione dei certificati SSL per le push notifications

A questo link sono riportate tutte le procedure per poter generare i certificati per il mondo APPLE

https://medium.com/@ankushaggarwal/generate-apns-certificate-for-ios-push-notifications-85e4a917d522

ESEMPI DI MESSAGGI JSON APN

Di seguito sono riportati due esempi di notifiche inviate dai server APPLE. Sono dei semplici messaggi JSON

Esempio Payload Semplice

```
{"aps" :
        {
         "alert":"Message received from Bob”},
         "acme2" : [ "bang",  "whiz" ]
        }
```

Esempio Payload Complesso N.B. Attualmente non gestito da Embarcadero

```
{"aps":
        {"alert":
            {“title":"Game",
             "body" : "Bob wants to play”,
             "badge" : 5
            },
            “acme1":"bar",
            "acme2" : [ "bang",  "whiz" ]
        }
```

BUGFIX - STANDARD DELPHI

Per gestire il Payload esteso è stato necessario modificare alcune unit standard Delphi

REST.Backend.PushDevice

```
procedure TPushData.TAPS.Save(const AJSONObject: TJSONObject; const ARoot: string);
var
  LBadge: Integer;
begin
  TPushData.SetString(AJSONObject, ARoot, TNames.Alert, Alert);
  if TryStrToInt(Badge, LBadge) then
    TPushData.SetValue(AJSONObject, ARoot, TNames.Badge, TJSONNumber.Create(LBadge))
  else
    // Badge may not be a number.  For, example "Increment" may be used with Parse
    TPushData.SetString(AJSONObject, ARoot, TNames.Badge, Badge);
  TPushData.SetString(AJSONObject, ARoot, TNames.Sound, Sound);

  // M.M.
  if Alert='' then
  begin
    TPushData.SetString(AJSONObject, TNames.Alert, TNames.Title, Title);
    TPushData.SetString(AJSONObject, TNames.Alert, TNames.Body, Body);
  end;
end;
```

BUGFIX - STANDARD DELPHI

REST.Backend.PushDevice

```
procedure TPushData.TAPS.Load(const AJSONObject: TJSONObject; const ARoot:
string);
begin
  Alert :=  TPushData.GetString(AJSONObject, ARoot, TNames.Alert);
  Badge :=  TPushData.GetString(AJSONObject, ARoot, TNames.Badge);
  Sound :=  TPushData.GetString(AJSONObject, ARoot, TNames.Sound);

  if Alert='' then
  begin
    Title :=  TPushData.GetString(AJSONObject, TNames.Alert, TNames.Title);
    Body :=  TPushData.GetString(AJSONObject, TNames.Alert, TNames.Body);
  end;
end;
```

BUGFIX - STANDARD DELPHI

FMX.Platform.iOS

```
procedure TNotificationCenterDelegate.userNotificationCenter(center: UNUserNotificationCenter;
response: UNNotificationResponse; completionHandler: Pointer); cdecl;
var
  CompletionHandlerImpl: procedure; cdecl;
begin
  // We don't use TMessage<UNNotificationResponse>, because we declare 2 types of
UNNotificationResponse>
  // in the different units with the same guid for avoiding interface breaking changes.
  // So TMessage<UNNotificationResponse> are not equaled types. So we use Pointer instead.
  // It's only for Update 1!

  //TMessageManager.DefaultManager.SendMessage(Self,
TMessage<Pointer>.Create(NSObjectToID(response)));

  TMessageManager.DefaultManager.SendMessage(Self,
TMessage<UNNotificationResponse>.Create(response));

  @CompletionHandlerImpl := imp_implementationWithBlock(completionHandler);
  CompletionHandlerImpl;
  imp_removeBlock(@CompletionHandlerImpl);
end;
```

SERVIZIO DI GOOGLE FIREBASE

CONFIGURAZIONE GOOGLE

☑Operazioni preliminari

- Iscrizione alla piattaforma GOOGLE tramite creazione di un account
- Effettuare la configurazione di FIREBASE

A questo link sono riportate tutte le procedure di funzionamento del cloud messaging di GOOGLE

https://firebase.google.com/docs/cloud-messaging/

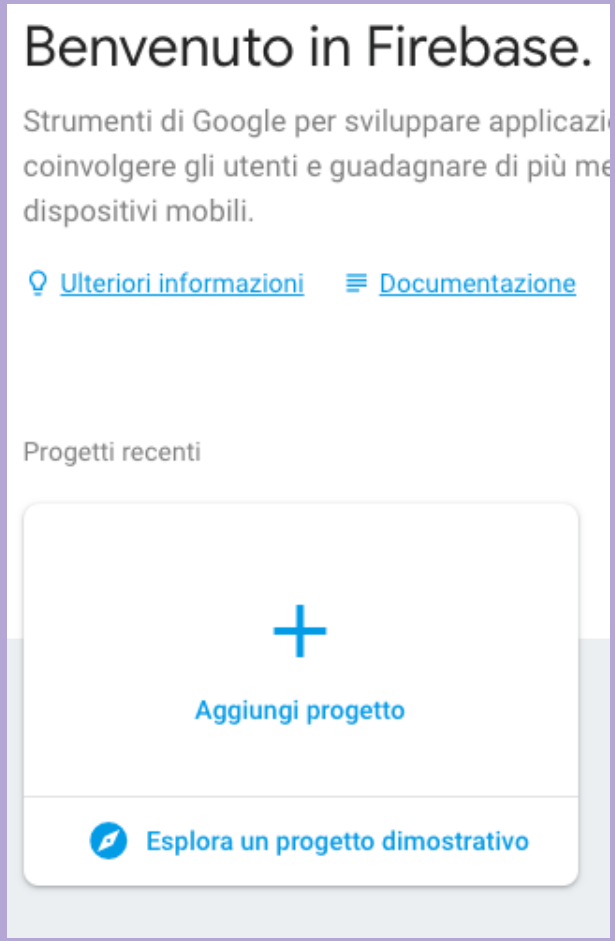
ESEMPI DI MESSAGGI JSON FIREBASE - ONE SIGNAL

Di seguito è riportato un esempio di notifica inviate dai server GOOGLE.
Sono dei semplici messaggi JSON

```
DA FIREBASE
{     …..
     “gcm.notification.title”:”PROVA TITOLO”,
     ”gcm.notification.body”:”TEST BODY”
     }

DA ONE SIGNAL
{     …..
     “title”:”PROVA TITOLO”,
     ”alert”:”TEST BODY”
     }
```

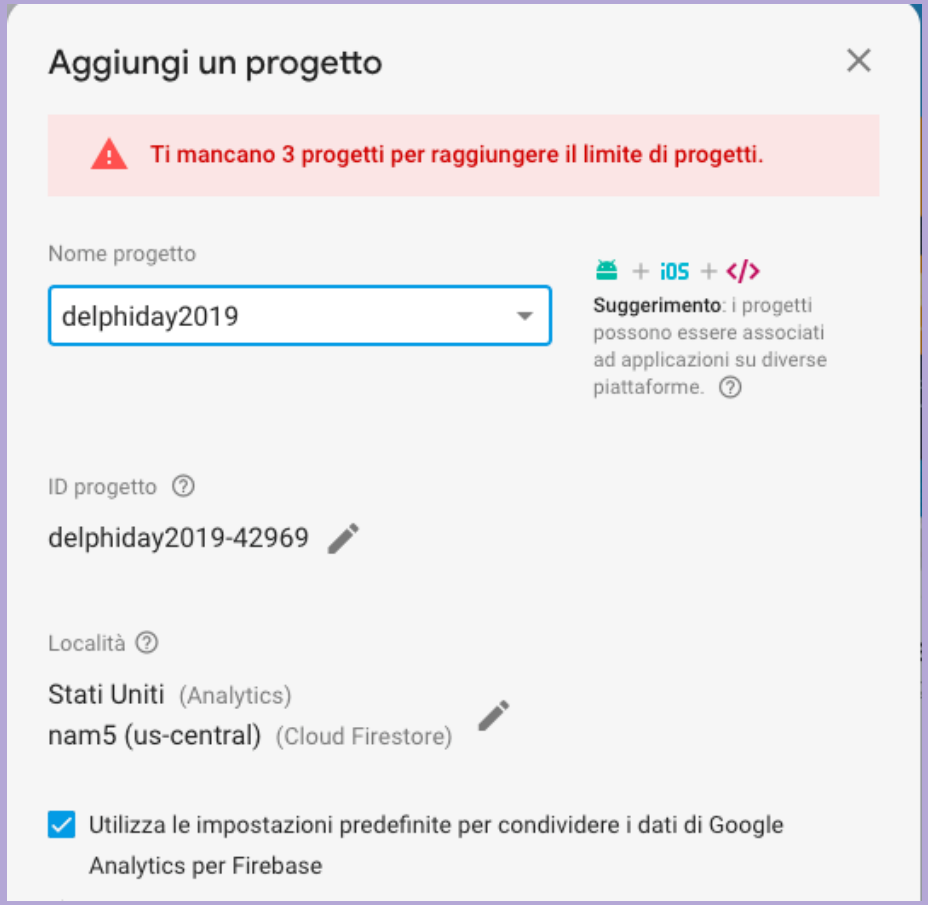
CONFIGURAZIONE GOOGLE FIREBASE

Come prima fase è necessario avere un account google.
Dopo la registrazione accedere al sito https://console.firebase.google.com
e creare il nuovo
progetto premendo su «Aggiungi Progetto»



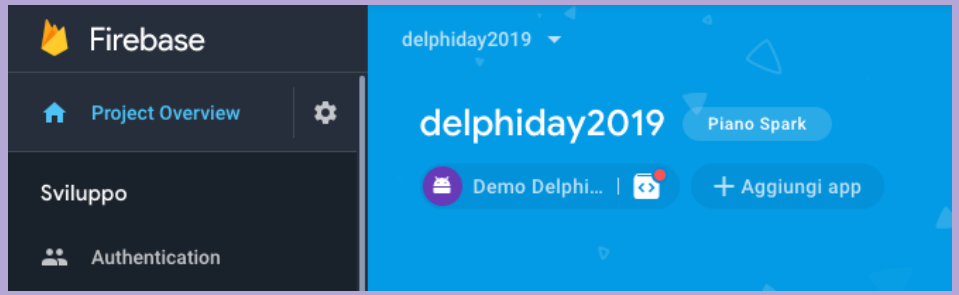
CONFIGURAZIONE GOOGLE FIREBASE

E’ necessario inserire i dati relativi al progetto.
Nominiamo il progetto “delphiday2019” premiamo “Crea Progetto”

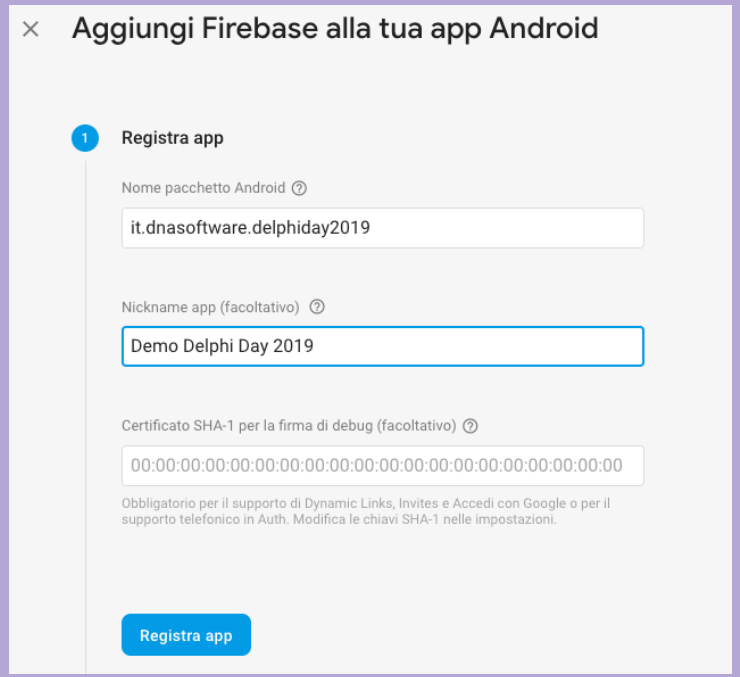


CONFIGURAZIONE GOOGLE FIREBASE

Terminata questa fase il progetto è creato e verrà presentata la seguente maschera

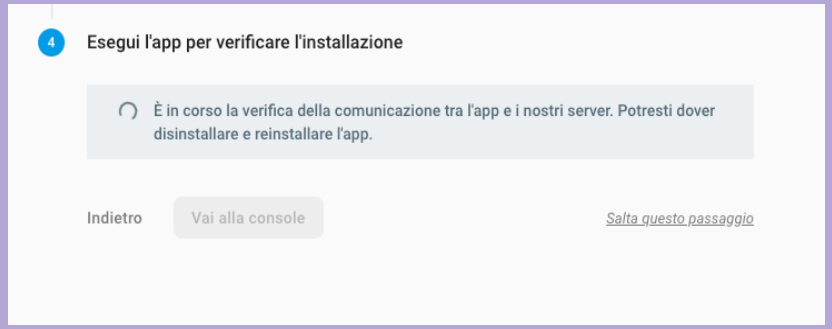


Premendo sull'icona di Android possiamo andare a inserire la nostra App per abilitarla alla ricezione delle push

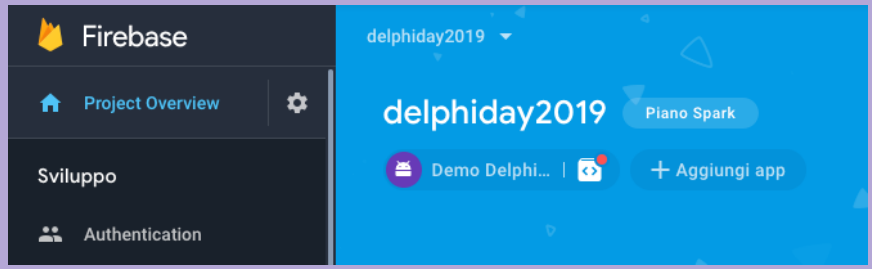


CONFIGURAZIONE GOOGLE FIREBASE

N.B. Prestate attenzione al punto 4 perché la verifica dell'installazione è una cosa che deve essere saltata in questa fase. Quando arrivate alla fase 4 premete sul pulsante "Salta questo passaggio" come nell'immagine sotto.

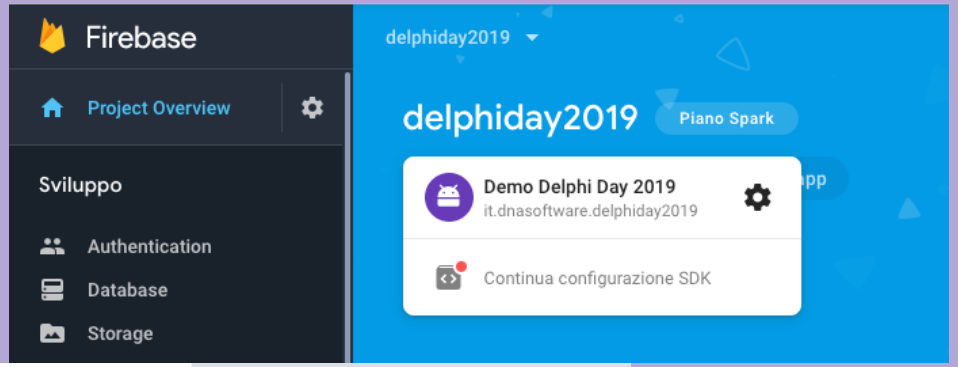


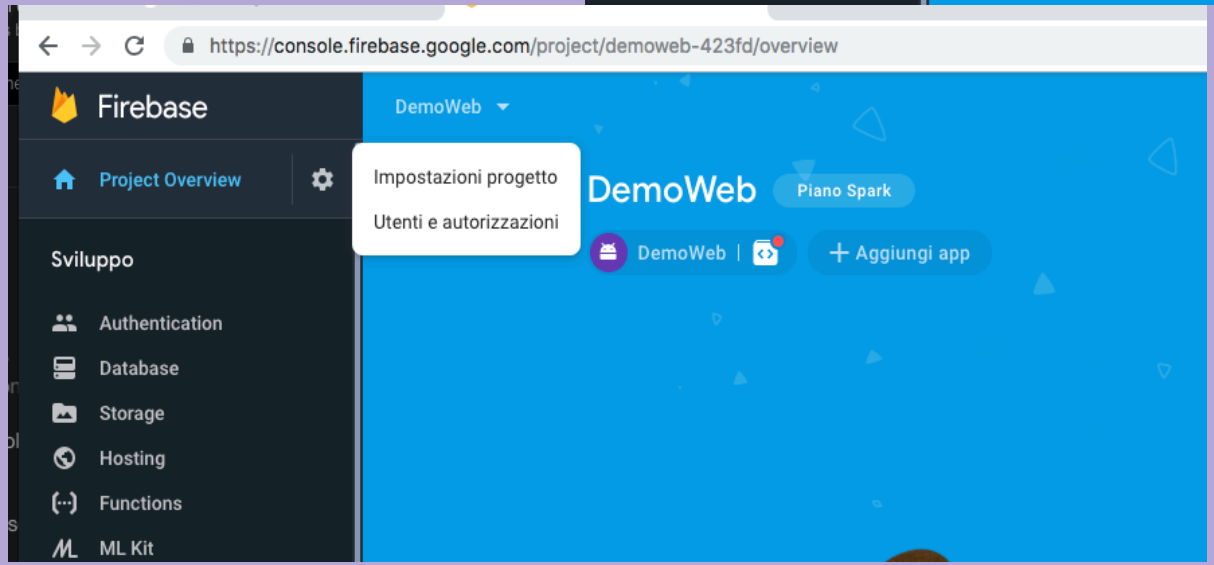
Terminati tutti i passaggi il nostro progetto sarà configurato con una nuova app android come si può notare dall'icona presente sotto al nome del nostro progetto nell'immagine seguente.



CONFIGURAZIONE GOOGLE FIREBASE

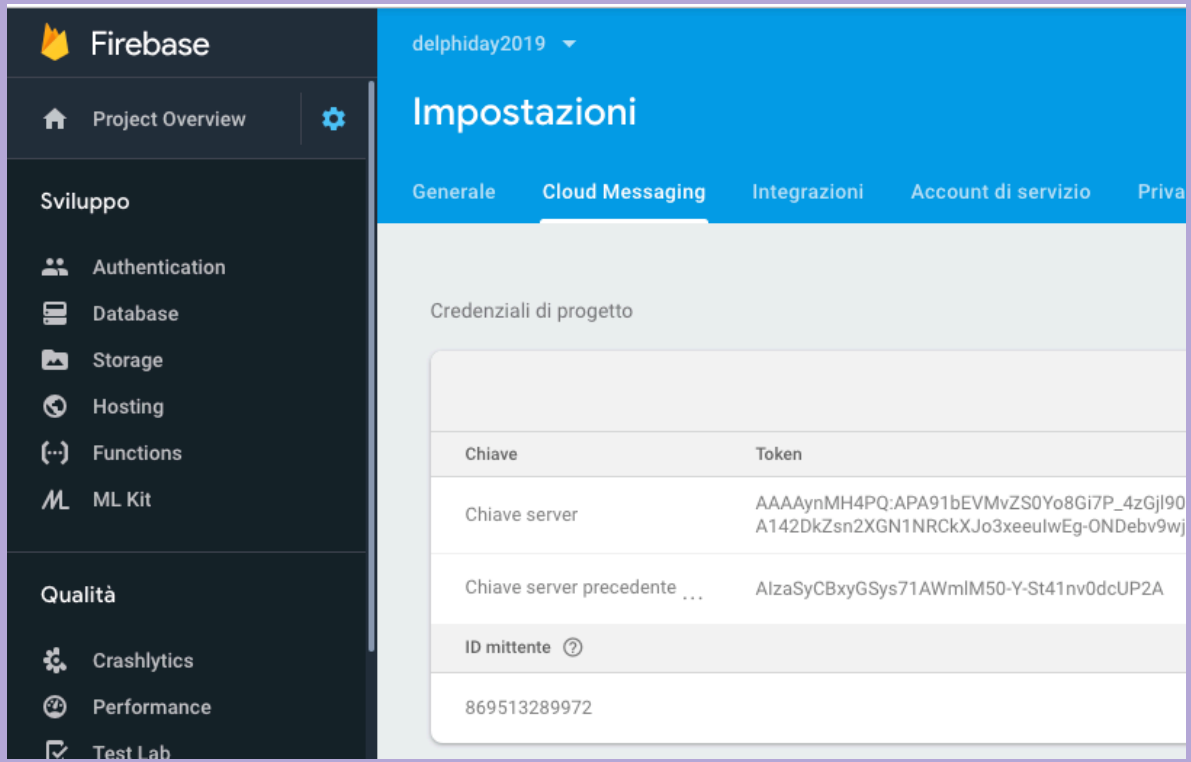
A questo punto è possibile entrare nella configurazione del progetto, vedi immagine seguente, per recuperare i valori che serviranno successivamente per poter abilitare l'invio e la ricezione delle push.





CONFIGURAZIONE GOOGLE FIREBASE

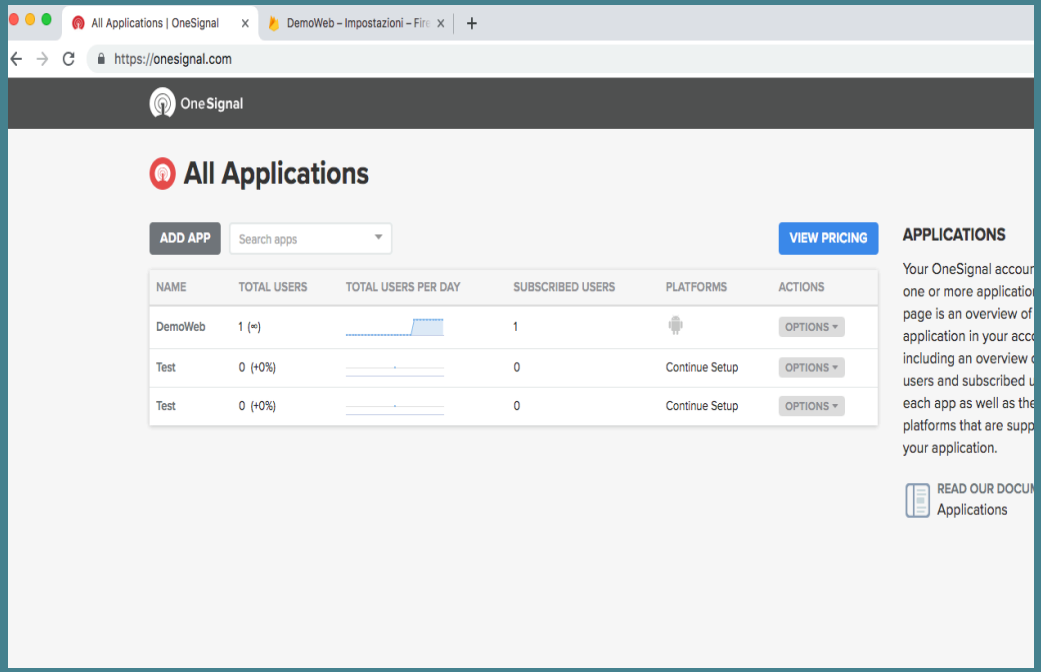
Recuperare i valori chiamati **“ID MITTENTE”** o “SENDER ID” che serviranno nella nostra APP



CONFIGURAZIONE INTERMEDIARIO ONESIGNAL

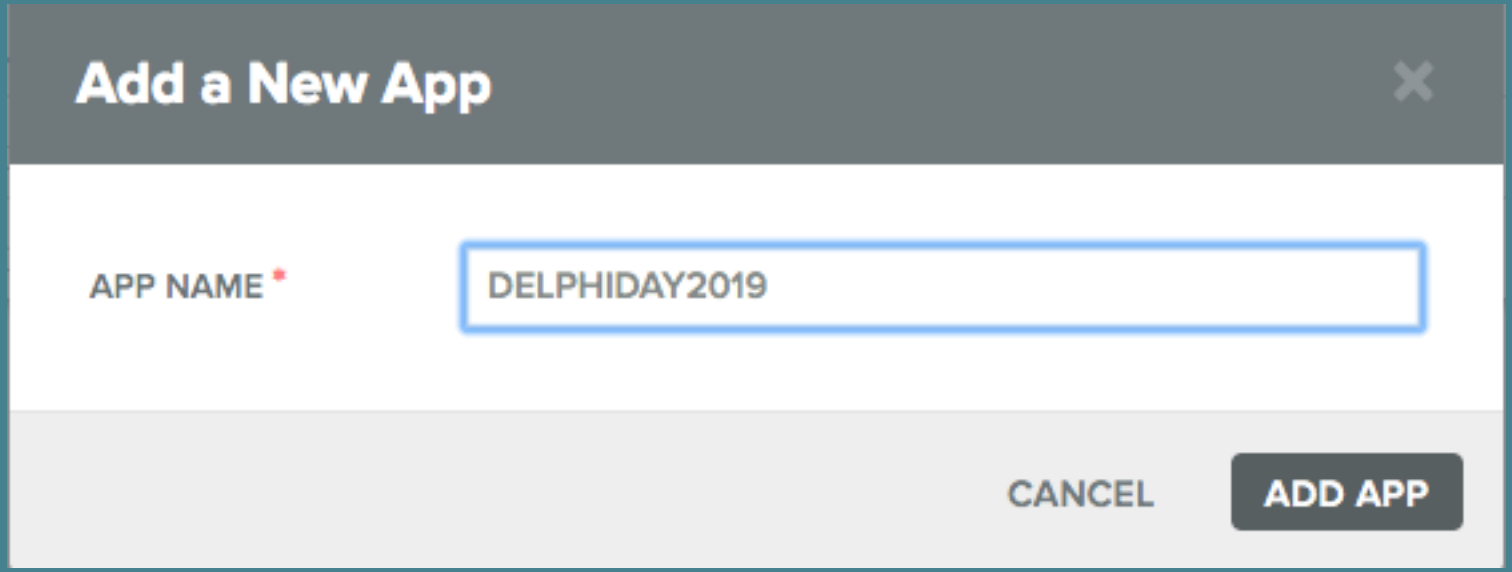
CONFIGURAZIONE INTERMEDIARIO - ONESIGNAL

Entrare nel sito https://onesignal.com ed effettuare la registrazione.
Dopo la registrazione per aggiungere una nuova app è necessario premere sul pulsante “Add App” come nell’immagine qui sotto



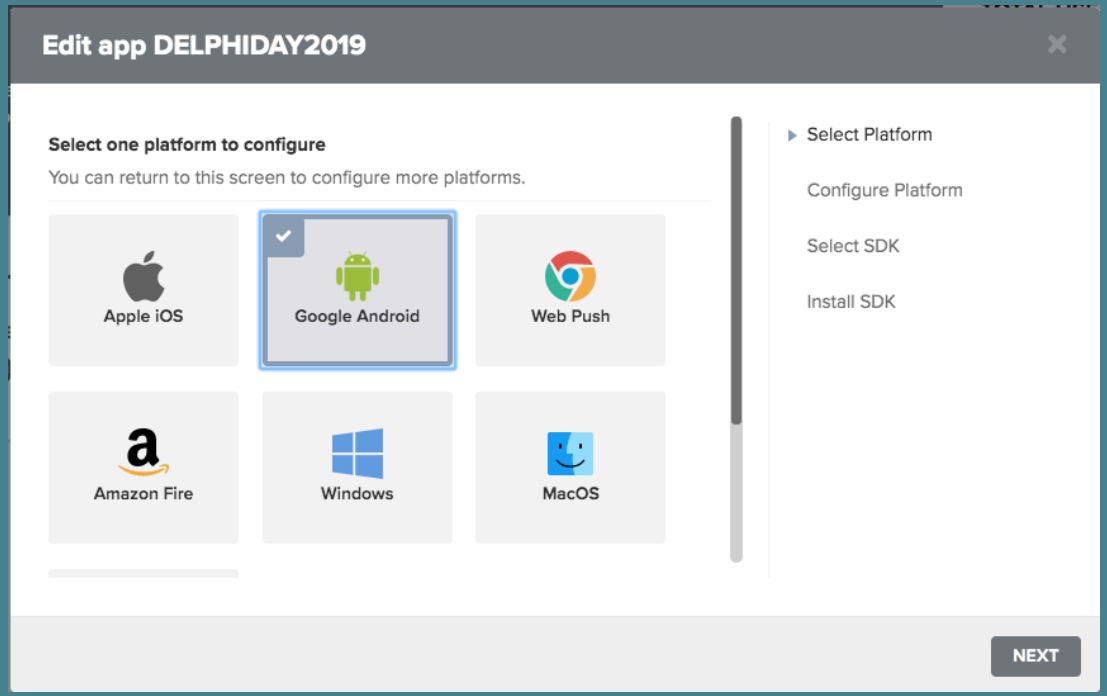
CONFIGURAZIONE INTERMEDIARIO - ONESIGNAL

Inserire il nome da dare all'applicazione. (DELPHIDAY2019)



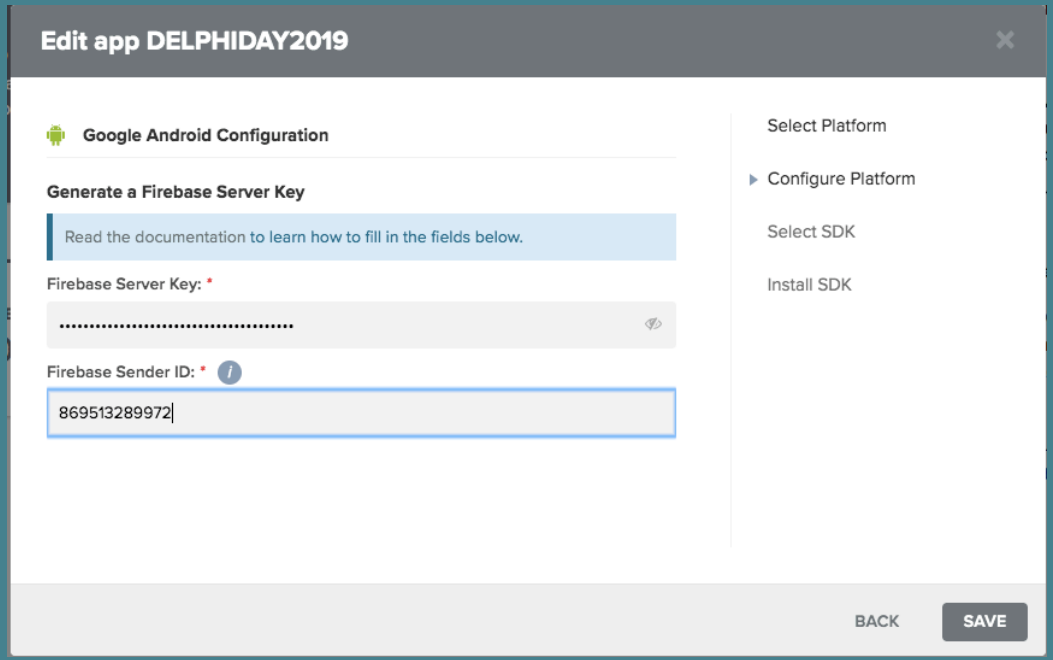
CONFIGURAZIONE INTERMEDIARIO - ONESIGNAL

A questo punto viene richiesto la piattaforma da abilitare per le push.
Nel nostro caso selezioniamo Google Android



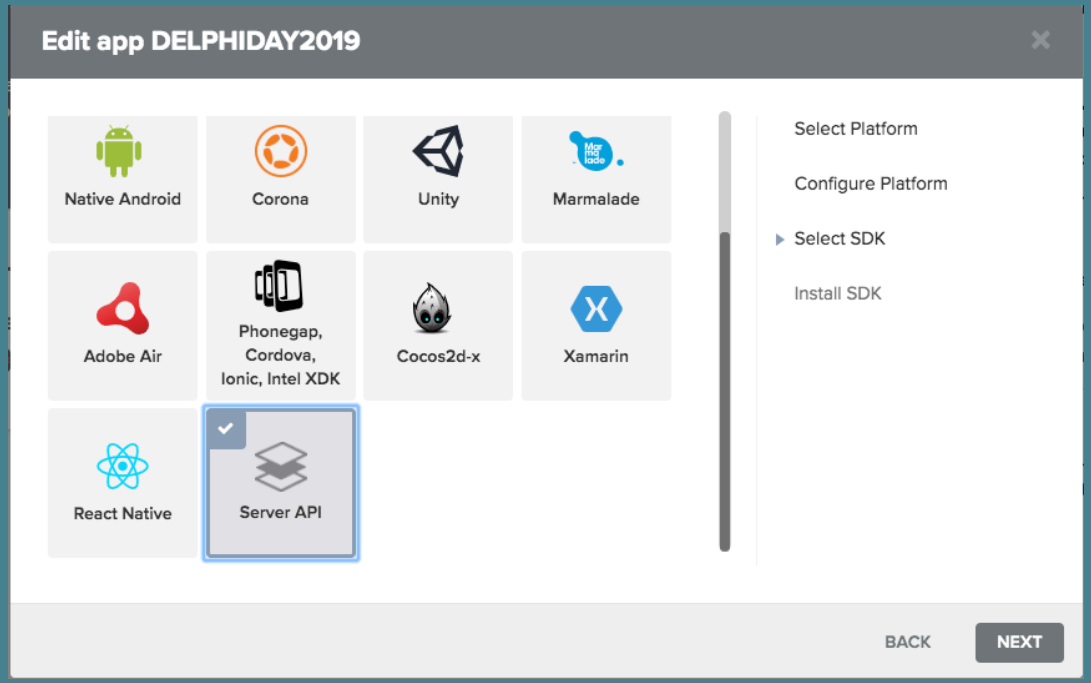
CONFIGURAZIONE INTERMEDIARIO - ONESIGNAL

A questo punto ci viene richiesto di configurare i parametri di FIREBASE. Questo perché ONESIGNAL fa da ponte tra il nostro server e GOOGLE ma poi le push vengono mandate direttamente tramite FIREBASE.



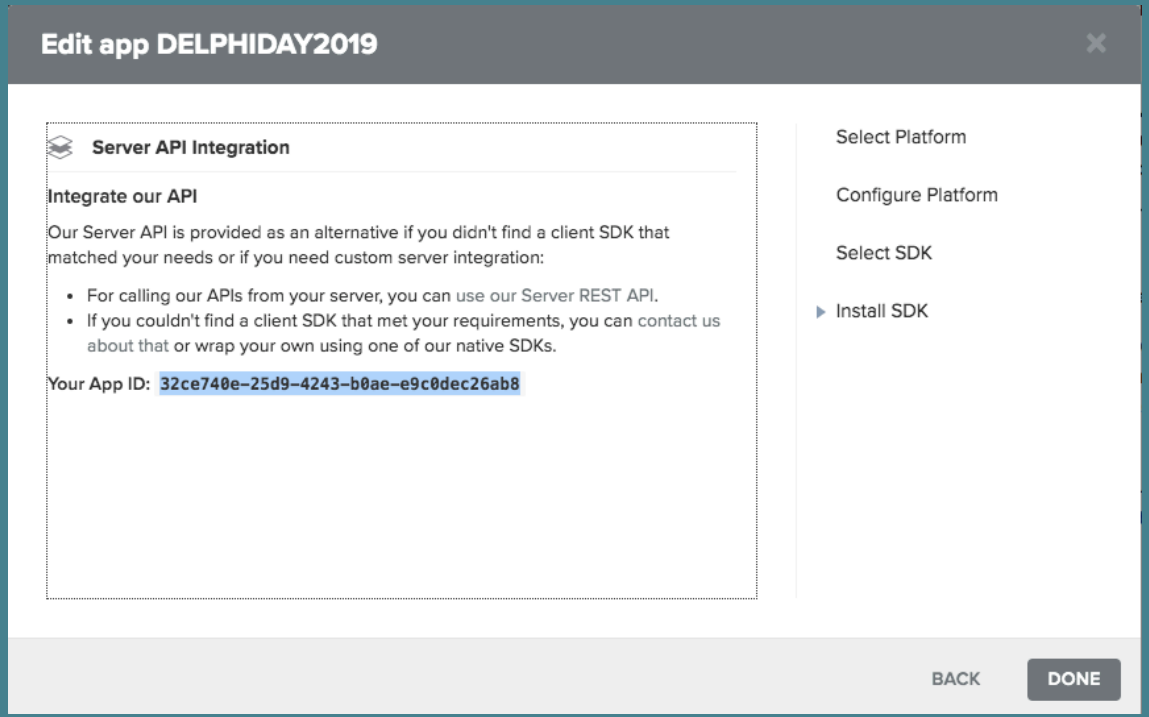
CONFIGURAZIONE INTERMEDIARIO - ONESIGNAL

A questo punto ci viene richiesto quale SDK vogliamo utilizzare per la comunicazione con ONESIGNAL. Chiaramente **non esiste un plugin per Delphi** e quindi selezioniamo il **plugin per REST**



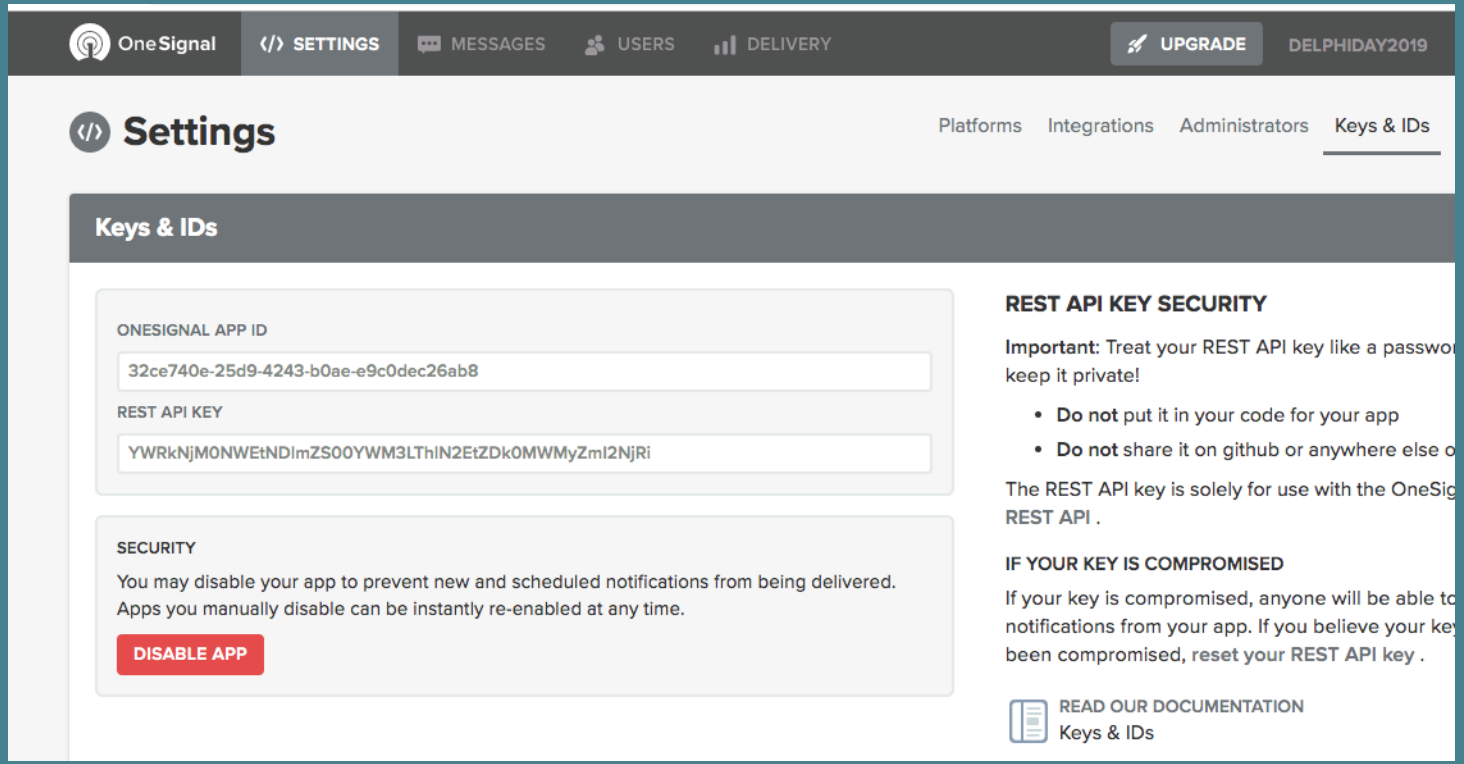
CONFIGURAZIONE INTERMEDIARIO - ONESIGNAL

A questo punto ci viene presentata la chiave della nostra applicazione che dovremo utilizzare nelle chiamate verso ONESIGNAL dal nostro server REST



CONFIGURAZIONE INTERMEDIARIO - ONESIGNAL

Terminata correttamente la configurazione si accede alla DASHBOARD



DEMO TIME 1

DEMO TIME - RIFERIMENTI

- SERVER REST
  - DelphiDay2019\Talk1\Demo\server\source\DEMOSERVER.DPR
- CLIENT MOBILE
  - DelphiDay2019\Talk1\Demo\client\source\DEMOPUSH.DPR
- PROJECT GROUP
  - DelphiDay2019\Talk1\Demo\DELPHIDAY2019.groupproj

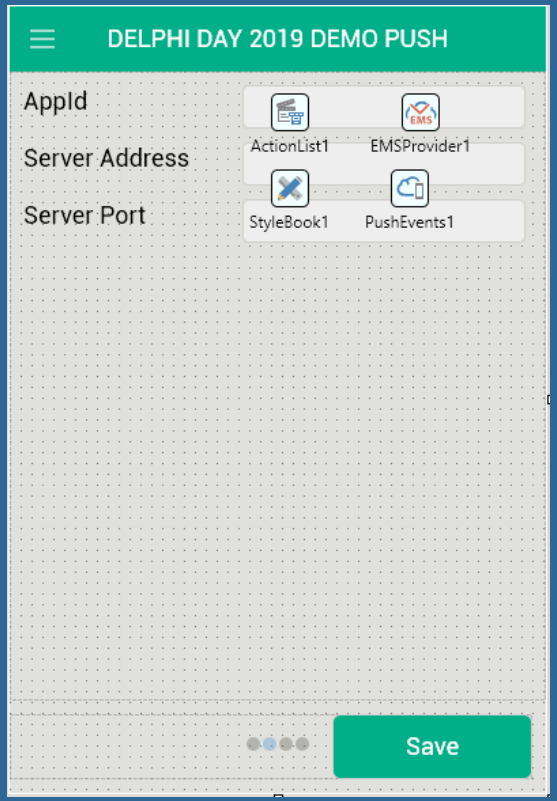
LA NOSTRA APP CHE RICEVE LE PUSH

APPLICAZIONE MOBILE
Recupera il device token e lo invia al nostro server
Recupera il device token e lo invia a GOOGLE e APPLE
Si mette in attesa per la ricezione delle notifiche

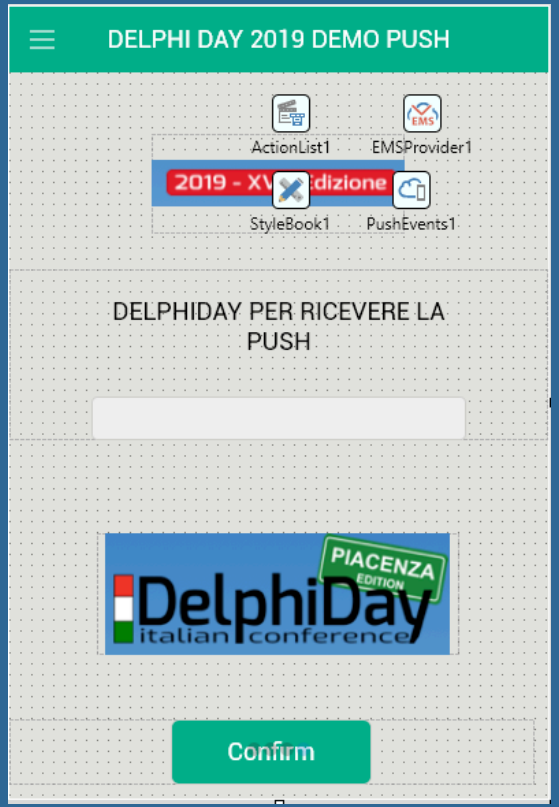
SERVER REST
Registra i device nel nostro server
Registra i device su ONESIGNAL
Scatena l'invio di una push da ONESIGNAL

APPLICAZIONE MOBILE

VIEW DI CONFIGURAZIONE



VIEW MASTER



APPLICAZIONE MOBILE COMPONENTI ED EVENTI

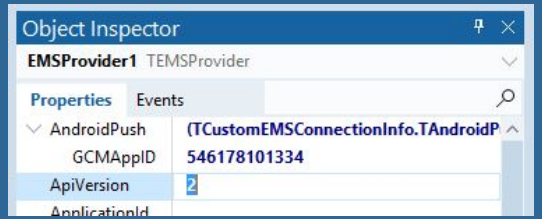
Quali componenti standard utilizzare

- TEMSProvider
  (nel caso di Android attenzione alla proprietà GCMAppID = SENDER_ID)
- TPushEvents
  (attenzione al flag AutoregisterDevice = False)

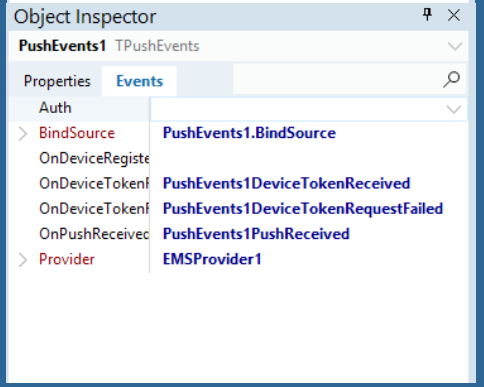
Quali eventi dobbiamo gestire

- OnDeviceTokenReceived
- OnDeviceTokenRequestFailed
- OnPushReceived

TEMSProvider

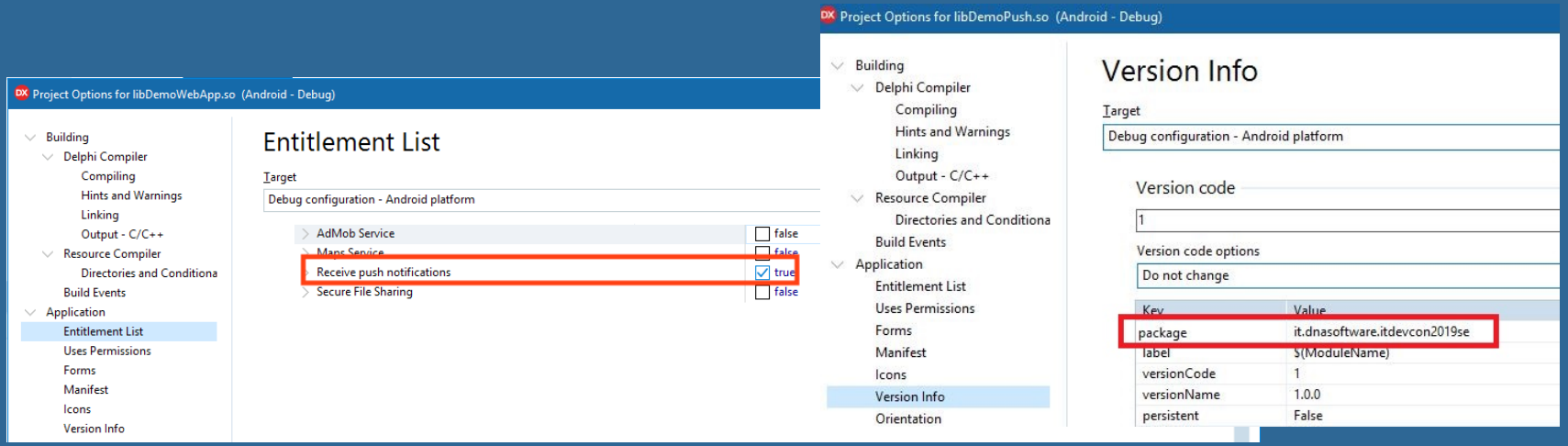


TPushEvents



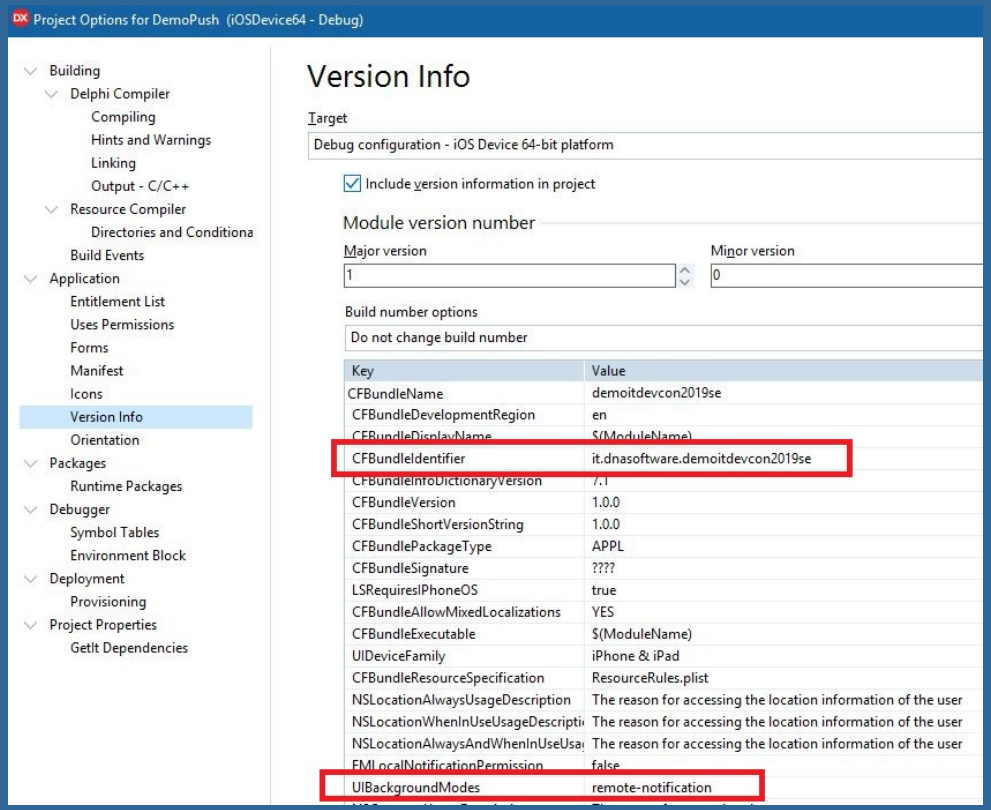
APPLICAZIONE MOBILE ANDROID INFORMAZIONI NEL PROGETTO

- Per attivare l'applicazione alla gestione delle push nel progetto
  - Abilitare il flag per la ricezione delle push (RECEIVE PUSH NOTIFICATIONS = TRUE)
  - Il nome del package deve corrispondere con il nome dato al progetto in FIREBASE al momento della configurazione



APPLICAZIONE MOBILE iOS INFORMAZIONI NEL PROGETTO

- Per attivare l'applicazione alla gestione delle push nel progetto sono necessari i seguenti accorgimenti
  - Abilitare il flag per la ricezione delle push (UIBACKGROUNDMODES =REMOTE-NOTIFICATION)
  - Il nome del CFBUNDLEIDENTIFIER deve corrispondere al nome dato alla APP nel portale APPLE



CODICE DEGLI EVENTI DA GESTIRE

RICEZIONE DEL DEVICE TOKEN CORRETTA

```
procedure TSGMainForm.PushEvents1DeviceTokenReceived(Sender: TObject);
begin
  FDeviceId := PushEvents1.DeviceID;     ← RECUPERA IL DEVICEID
  FDeviceToken := PushEvents1.DeviceToken; ←  RECUPERA IL DEVICETOKEN
  Memo1.Lines.Add('DeviceID: ' + FDeviceId);
  Memo1.Lines.Add('Device Token: ' + FDeviceToken);
  Memo1.Lines.Add('Ready to receive!');
  RegisterTokenOnServerAction.Execute;    ←  INVIA IL TOKEN AL SERVER REST
end;
```

RICEZIONE DEL DEVICE TOKEN FALLITA

```
procedure TSGMainForm.PushEvents1DeviceTokenRequestFailed(Sender: TObject; const
AErrorMessage: string);
begin
  ShowMessage('Device Token Request Failed '+AErrorMessage);
end;
```

CODICE DEGLI EVENTI DA GESTIRE

INVIO DEVICE TOKEN AL SERVER

```
procedure TSGMainForm.RegisterTokenOnServerActionExecute(Sender: TObject);
begin
  ITask(TTask.Create(
      procedure
      begin
       LResponse := FClientRest.UpdateDevice(FDeviceId, FDeviceToken);
       LJSONObject := TJSONObject.ParseJSONValue(LResponse) as TJSONObject;
       LJSONObject.TryGetValue<integer>('statuscode',LStatusCode);
       LJSONObject.TryGetValue<string>('message',LMessage);
       if LStatusCode<>200 then
                raise Exception.Create('error in update device');
      end)).Start;
end;
```

CODICE DEGLI EVENTI DA GESTIRE

RICEZIONE DEL MESSAGGIO DI NOTIFICA

```
procedure TSGMainForm.PushEvents1PushReceived(Sender: TObject; const AData: TPushData);
begin
      {$IFDEF ANDROID}
      Memo1.Lines.Add('Message: ' + AData.Message);
      Memo1.Lines.Add('Title: ' + AData.GCM.Title);
      // Se la notifica arriva quando l'app è aperta la mostra tra le notifiche
      if Assigned(Sender) then
ShowNotification('MyNot',AData.Message,AData.GCM.Title,True,0);
      {$ENDIF}
      {$IFDEF IOS}
      Memo1.Lines.Add('Alert: ' + AData.APS.Alert);
      Memo1.Lines.Add('Title: ' + AData.APS.Title);
      Memo1.Lines.Add('Body: ' + AData.APS.Body);
      {$ENDIF}
);
end;
```

APPLICAZIONE SERVER REST – BASE DATI E MODELLO

Per persistere i dati relativi ai device in arrivo dai diversi dispositivi MOBILE nel nostro modello di business abbiamo definito

- CLASSE DEVICE con la seguente interfaccia

```
TDevice = class(TBaseBO)
public
        property DeviceId: string read FDeviceId write SetDeviceId;
        property DeviceToken: string read FDeviceToken write SetDeviceToken;
        property Description: string read FDescription write SetDescription;
        property ExternalId: string read FExternalId write SetExternalId;
        property DeviceType: string read FDeviceType write SetDeviceType;
        property DeviceOs: string read FDeviceOs write SetDeviceOs;
        property DeviceOsVersion: string read FDeviceOsVersion write SetDeviceOsVersion;
        property DeviceModel: string read FDeviceModel write SetDeviceModel;
        property MailAddress: string read FMailAddress write SetMailAddress;
 end;
```

- TABELLA DEVICES con la seguente struttura

```
CREATE TABLE `devices` (
      `ID` int(11) NOT NULL AUTO_INCREMENT,
      `DEVICEID` varchar(255) NOT NULL,
      `DEVICETOKEN` varchar(255) DEFAULT NULL,
      `DESCRIPTION` varchar(255) DEFAULT NULL,
      `EXTERNALID` varchar(255) DEFAULT NULL,
      `DEVICETYPE` varchar(10) DEFAULT NULL,
      `DEVICEOS` varchar(255) DEFAULT NULL,
      `DEVICEOSVERSION` varchar(255) DEFAULT NULL,
      `DEVICEMODEL` varchar(255) DEFAULT NULL,
      `MAILADDRESS` varchar(100) DEFAULT NULL,
 PRIMARY KEY (`ID`),  UNIQUE KEY `DEVICEID_UNIQUE` (`DEVICEID`))
```

APPLICAZIONE SERVER REST – RISORSE

- RISORSA DEVICE amministrata dal seguente controller

```
[MVCPath('/device')]
TDeviceController = class(TBaseController)
public
  [MVCPath]
  [MVCHTTPMethod([httpGET])]
  procedure GetDevices;
  [MVCPath('/($deviceid)')]
  [MVCHTTPMethod([httpGET])]
  function GetDeviceByID(deviceid: string; ARender: boolean = True): TDevice;
  [MVCPath('/($deviceid)')]
  [MVCHTTPMethod([httpDelete])]
  procedure DeleteDeviceByID(deviceid: string);
  [MVCPath('/($deviceid)')]
  [MVCHTTPMethod([httpPUT])]
  procedure UpdateDeviceByID(deviceid: string);
  [MVCPath]
  [MVCHTTPMethod([httpPOST])]
  procedure CreateDevice(Context: TWebContext);
  [MVCPath('/authorize/($code)/($deviceid)')]
  [MVCHTTPMethod([httpPOST])]
  procedure AuthorizeDevice(code: string; deviceid: string);

end;
```

DIALOGO CLIENT – SERVER

- In fase di apertura della APP il client chiama un metodo UpdateDEVICE che si occupa di effettuare la chiamata al server REST passando il DEVICETOKEN, DEVICEID.

```
LResponse := FClientRest.UpdateDevice(FDeviceId, FDeviceToken, FModelName, FOSPlatform, FOSMajMin, LDeviceType, FModelName);

function TClientRest.UpdateDevice(const aDeviceId: string; const aDeviceToken: string; const aDeviceDescription: string; aDeviceOs: string;
aDeviceOsVersion: string; aDeviceType: string; aDeviceModel: string): string;
begin
        Result := DoPut('/device/'+aDeviceId, Format('{"devicetoken":"%s", "description" : "%s", "devicetype": "%s", "deviceos": "%s",
        "deviceosversion": "%s", "devicemodel":"%s"}',
        [aDeviceToken,aDeviceDescription,aDeviceType,aDeviceModel,aDeviceOsVersion,aDeviceModel]));
end;
```

DIALOGO CLIENT – SERVER

- Il server ricevuta la chiamata verifica se il DEVICEID è già presente nel DB tabella DEVICES. Se presente aggiorna i dati altrimenti inserisce un nuovo record nella tabella. Dopo questa operazione viene chiamato ONESIGNAL passando il DEVICETOKEN e aggiornando sulla tabella DEVICES il campo EXTERNALID (PLAYERID o USERID di ONESIGNAL) che corrisponde all'identificativo univoco assegnato da ONESIGNAL al nostro DEVICE.

```
procedure TDeviceController.UpdateDeviceByID(deviceid: string);
begin
  LDeviceFromBody := Context.Request.BodyAs<TDevice>;
  LDeviceFromDb := GetDeviceByID(deviceid, False);
  if Assigned(LDeviceFromDb) then
  begin
    LDeviceFromDb.DeviceToken := LDeviceFromBody.DeviceToken;
    LDeviceFromDb.Description := LDeviceFromBody.Description;
GetDevicesService.Update(LDeviceFromDb);
    LDeviceFromDb.ExternalId := TOneSignal.UpdatePlayer(GetDataModule.Configuration.AppIdPushServer,
GetDataModule.Configuration.APIKeyPushServer, LDeviceFromDb.DeviceToken, LDeviceFromDb.ExternalId,
LDeviceFromDb.DeviceType, LDeviceFromDb.DeviceModel, LDeviceFromDb.DeviceOs+' '+LDeviceFromDb.DeviceOsVersion);
    GetDevicesService.UpdateExternalId(LDeviceFromDb);
    LDeviceFromDb.Free;
    Render(200, 'Device Updated');
  end
  else
  begin
    LDeviceFromBody.DeviceId := deviceid;
    GetDevicesService.Add(LDeviceFromBody);
    LDeviceFromBody.ExternalId := TOneSignal.UpdatePlayer(GetDataModule.Configuration.AppIdPushServer,
GetDataModule.Configuration.APIKeyPushServer, LDeviceFromBody.DeviceToken, '', LDeviceFromBody.DeviceType,
LDeviceFromBody.DeviceModel, LDeviceFromBody.DeviceOs+' '+LDeviceFromBody.DeviceOsVersion);
    GetDevicesService.UpdateExternalId(LDeviceFromBody);
    LDeviceFromBody.Free;
    Render(200, 'Device Created');
  end;
end;
```

DIALOGO CLIENT – SERVER

- Quando il client vuole inviare una notifica chiama Il server con il seguente metodo passando il deviceid e il testo che abbiamo inserito nell'edit

```
function TClientRest.AuthorizeDevice(const aCode: string; const aDeviceId: string): string;
begin
  Result := DoPost('/device/authorize/'+aCode+'/'+aDeviceId,'');
end;
```

- Il server riceve la chiamata e se il codice corrisponde a DELPHIDAY viene scatenato l'invio della push chiamando direttamente ONESIGNAL

```
procedure TDeviceController.AuthorizeDevice(code: string; deviceid: string);
begin
  if GetDevicesService.Authorize(code, LUrl) then  ← VERIFICA SE IL CODICE E' DELPHIDAY
  begin
    LDeviceFromDb := GetDeviceByID(deviceid, False);
    // Invia una notifica Push al dispositivo chiamando ONESIGNAL in un thread separato se nella tabella
    // il dispositivo ha già associato un ExternalId che corrisponde al playerid di ONESIGNAL
    if Assigned(LDeviceFromDb) and (not LDeviceFromDb.ExternalId.IsEmpty) then
    begin
   TOneSignal.SendPushToPlayer(GetDataModule.Configuration.AppIdPushServer,
GetDataModule.Configuration.APIKeyPushServer,
LDeviceFromDb.ExternalId); ← INFORMA ONESIGNAL DI INVIARE LA NOTIFICA
    end;
    if Assigned(LDeviceFromDb) then LDeviceFromDb.Free;
    Render(200, LUrl)
  end
  else
    Render(401, 'Device Unauthorized');
end;
```

DIALOGO CLIENT – SERVER

- Questo è il metodo che controlla il codice in arrivo dal client

```
function TDevicesService.Authorize(ACode: string; out oUrl: string):
boolean;
begin
  Result := False;
  oUrl := '';

  if LowerCase(ACode)='delphiday' then
  begin
    Result := True;
    oUrl := 'www.dnasoftware.it';
  end;
end;
```

DIALOGO CLIENT – SERVER

- Questo è il metodo che informa ONESIGNAL di INVIARE la NOTIFICA

```
class function TOneSignal.SendPushToPlayer(const aAppId, aAPIKey,
  aExternalId: string): boolean;
begin
  LIdHTTP := TIdHTTP.Create(nil);
  try
    LIdIOHandler := TIdSSLIOHandlerSocketOpenSSL.Create(LIdHTTP);
    LIdIOHandler.SSLOptions.Method := sslvSSLv23;
    LIdHTTP.IOHandler := LIdIOHandler;
    LIdHTTP.Request.ContentType := 'application/json';
    LIdHTTP.Request.CustomHeaders.AddValue('Authorization','Basic '+aAPIKey);
    LIdHTTP.Request.Charset := 'UTF-8';

    jsonString := '{"app_id" : "'+aAppId+'",'+'"include_player_ids":["'+aExternalId+'"],'+
                  '"contents": {"en": "Push notifications con DELPHI e ONESIGNAL"},'+
                  '"headings": {"en": "DELPHIDAY2019 - MARCO MOTTADELLI"}}';

    JsonToSend := TStringStream.Create(jsonString, TEncoding.UTF8);
    try
      LSuccess := False;
      response := LIdHTTP.Post('https://onesignal.com/api/v1/notifications', JsonToSend);
      LJSONObject := TJSONObject.ParseJSONValue(response) as TJSONObject;
      try
        LJSONObject.TryGetValue<string>('id', LExternalId);

        if not LExternalId.IsEmpty then Result := True else Result := False;

      finally
        LJSONObject.Free;
      end;
    except
      on E: EIdHTTPProtocolException do response := e.ErrorMessage;
    end;
  finally
    LIdHTTP.Free;
  end;
end;
```

GREAT NEWS!

PATCH RILASCIATA DA EMBARCADERO CON SUPPORTO A FIREBASE

30 MAGGIO 2019

DAL
BLOG DI MARCO CANTU’

Firebase Android Push Notification Support with RAD Studio 10.3.1

https://community.idera.com/developer-tools/b/blog/posts/firebase-android-push-notification-support-with-rad-studio-10-3-1

DEMO TIME 2
INTEGRAZIONE DIRETTA CON FIREBASE

DEMO TIME - RIFERIMENTI

- SERVER REST
  - DelphiDay2019\Talk1\Demo\server\source\DEMOSERVER.DPR
- CLIENT MOBILE
  - DelphiDay2019\Talk1\Demo\client\source\DEMOPUSHFIREBASE.DPR
- PROJECT GROUP
  - DelphiDay2019\Talk1\Demo\DELPHIDAY2019.groupproj

LA NOSTRA APP CHE RICEVE LE PUSH TRAMITE FIREBASE

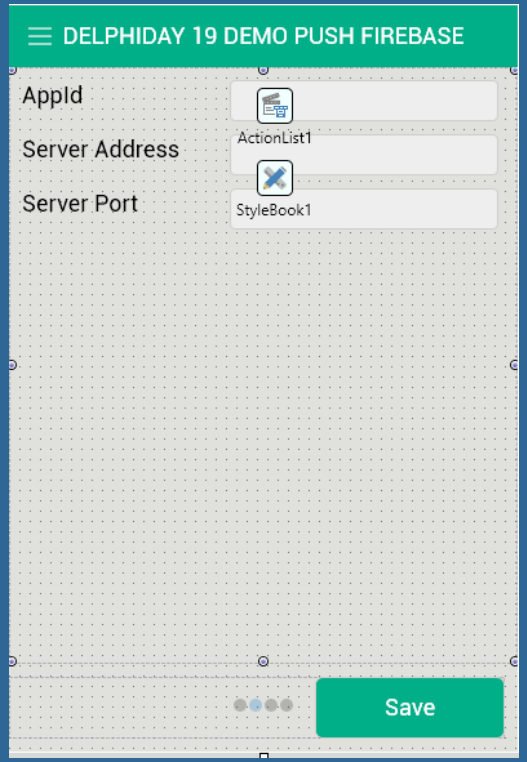
- ➔ Creare il progetto FIREBASE e registrare la nostra applicazione nella Google Firebase console
- ➔ Creare o aggiornare il nostro progetto FireMonkey usando il sistema di notifiche push
- ➔ Apportare le opportune modifiche al progetto FireMonkey per supportare Firebase al posto di Google Cloud Messaging

LA NOSTRA APP CHE RICEVE LE PUSH TRAMITE FIREBASE - STEP BY STEP

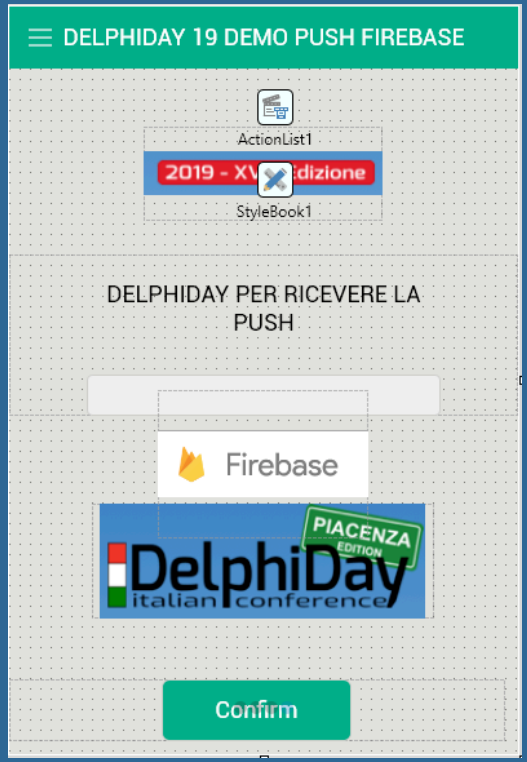
- ➔ da GETIT scaricare ed installare la PATCH
- ➔ sotto la cartella vengono scaricati tutti i file
  C:\Users\Public\Documents\Embarcadero\Studio\20.0\CatalogRepository\AndroidPushNotificationsPatch-1.0\Firebase
- ➔ integrare il file string.xml con i parametri presi dal JSON scaricato da FIREBASE
- ➔ nel progetto delphi andiamo a mettere tutto il codice necessario (vedi DemoPushFirebase)
- ➔ andiamo a disabilitare tutti i JAR e ad aggiungere quelli della patch
- ➔ andiamo ad inserire nel deploy string.xml
- ➔ andiamo a modificare il nostro android.manifest.template.xml
- ➔ compilare la app

APPLICAZIONE MOBILE

VIEW DI CONFIGURAZIONE



VIEW MASTER



CODICE DI GESTIONE DELLE PUSH

- Questo è il metodo che imposta gli oggetti necessari alla gestione delle notifiche

```
procedure TSGMainForm.FormShow(Sender: TObject);
var
  PushService: TPushService;
  ServiceConnection: TPushServiceConnection;
begin
  ApplySettingsOnEdit;

  if AppConfigurationOk then
  begin
    SetClientRestParameters;

    PushService := TPushServiceManager.Instance.GetServiceByName(TPushService.TServiceNames.GCM);
    ServiceConnection := TPushServiceConnection.Create(PushService);
    ServiceConnection.Active := True;
    ServiceConnection.OnChange := OnServiceConnectionChange;
    ServiceConnection.OnReceiveNotification := OnReceiveNotificationEvent;

    FDeviceId := PushService.DeviceIDValue[TPushService.TDeviceIDNames.DeviceId];
    Memo1.Lines.Add('DeviceID: ' + FDeviceId);
    Memo1.Lines.Add('Ready to receive!');

    // Si posiziona sul Tab della richiesta codice di controllo
    MainTabControl.TabIndex := 3;
  end
  else
    ShowConfigurationAction.Execute;
end;
```

CODICE DI GESTIONE DELLE PUSH

- Questo è il metodo che verifica e recupera il DEVICETOKEN

```
procedure TSGMainForm.OnServiceConnectionChange(Sender: TObject; PushChanges:
TPushService.TChanges);
var
  PushService: TPushService;
begin
  if TPushService.TChange.DeviceToken in PushChanges then
  begin
    {$IFDEF ANDROID}
    PushService :=
TPushServiceManager.Instance.GetServiceByName(TPushService.TServiceNames.GCM);
    FDeviceToken :=
PushService.DeviceTokenValue[TPushService.TDeviceTokenNames.DeviceToken];
    Memo1.Lines.Add('FireBase Token: ' + FDeviceToken);
    Log.d('Firebase device token: token=' + FDeviceToken);
    {$ENDIF}

    // Registra il token nel server
    RegisterTokenOnServerAction.Execute;
  end;
end;
```

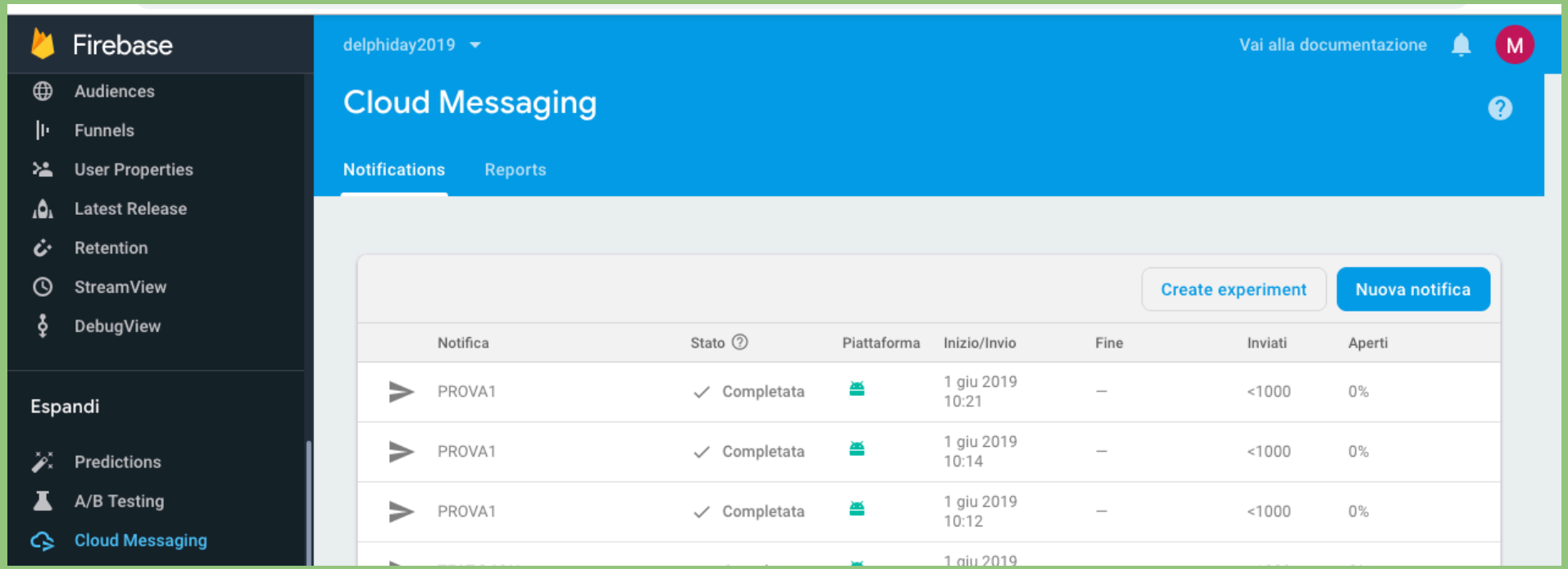
CODICE DI GESTIONE DELLE PUSH

- Questo è il metodo che si scatena quando arriva un push

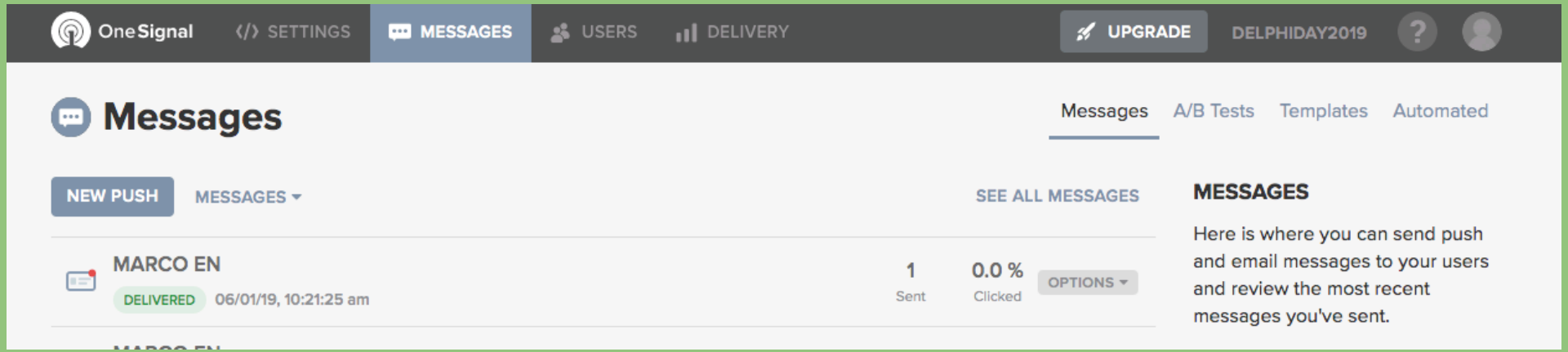
```
procedure TSGMainForm.OnReceiveNotificationEvent(Sender: TObject; const ServiceNotification: TPushServiceNotification);
begin
 // SE ARRIVA gcm.notification.body SIAMO NEL CASO DI NOTIFICA IN ARRIVO DA FIREBASE CONSOLE
  try
    lGcmBody := ServiceNotification.DataObject.GetValue('gcm.notification.body').Value;
    lGcmTitle := ServiceNotification.DataObject.GetValue('gcm.notification.title').Value;
  except
  end;

  // SE ARRIVA alert e title SIAMO NEL CASO DI NOTIFICA IN ARRIVO DA ONE SIGNAL
  ServiceNotification.DataObject.TryGetValue<string>('alert', lAlert);
  ServiceNotification.DataObject.TryGetValue<string>('title', lTitle);

  // SE ARRIVA MESSAGE TEXT SIAMO NEL CASO FIREBASE
  if lGcmBody<>'' then
  begin
    lFirebase := True;
    Memo1.Lines.Add('FIREBASE');
    Memo1.Lines.Add(DateTimeToStr(Now) + ' Body = ' + lGcmBody);
    Memo1.Lines.Add(DateTimeToStr(Now) + ' Title = ' + lGcmTitle);
  end;

  // SE ARRIVANO ALERT O TITLE SIAMO NEL CASO ONESIGNAL
  if (lAlert<>'') or (lTitle<>'') then
  begin
    lOneSignal := True;
    Memo1.Lines.Add('ONE SIGNAL');
    Memo1.Lines.Add(DateTimeToStr(Now) + ' Alert = ' + lAlert);
    Memo1.Lines.Add(DateTimeToStr(Now) + ' Title = ' + lTitle);
  end;

  if lOneSignal then
    ShowNotification('MyNot',lAlert, lTitle, True, 0);
  if lFirebase then
    ShowNotification('MyNot',lGcmBody, lGcmTitle, True, 0);
end;
```

INVIAMO LE NOSTRE PUSH

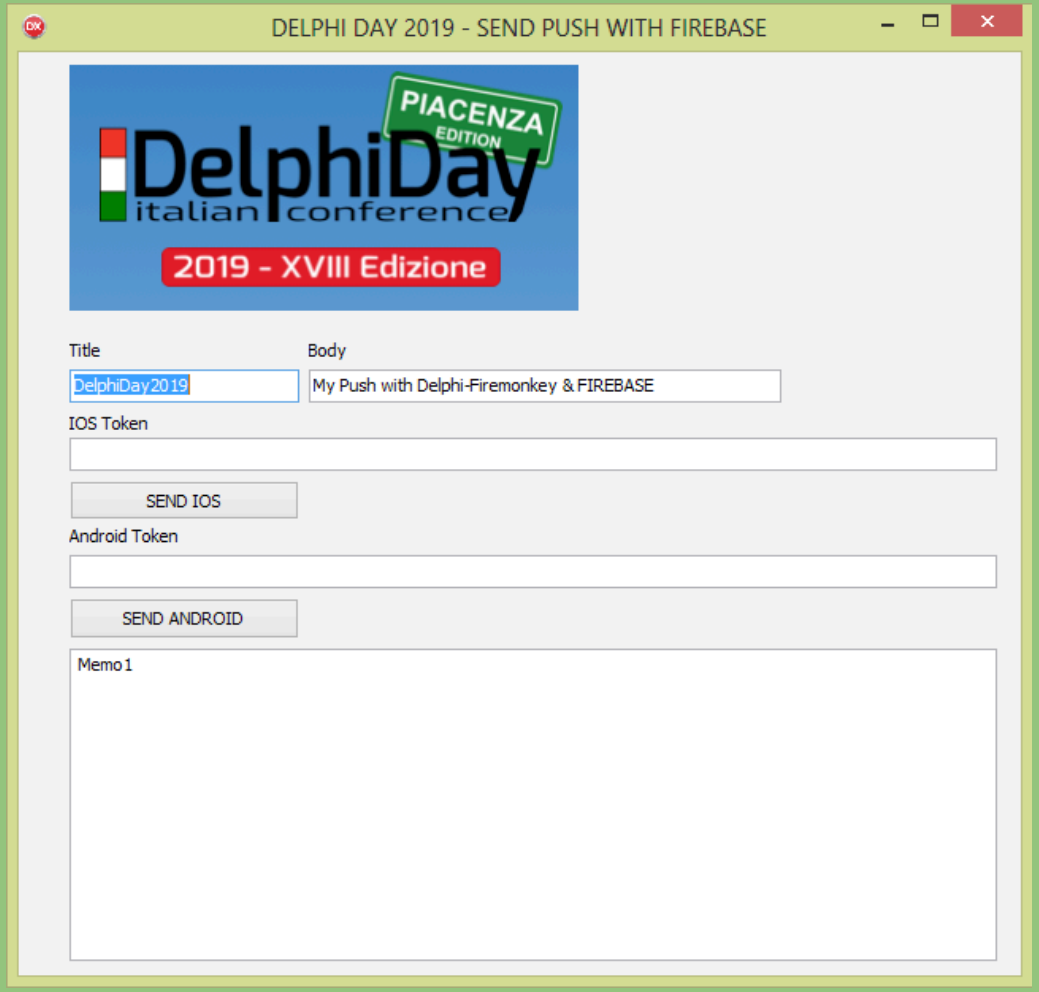
INVIO PUSH DA FIREBASE CONSOLE



INVIO PUSH DA ONE SIGNAL CONSOLE



INVIO PUSH DA APPLICAZIONE VCL



DEMO TIME 3
APPLICAZIONE VCL PER INVIO DELLE PUSH

DEMO TIME - RIFERIMENTI

- APPLICAZIONE VCL
  - DelphiDay2019\Talk1\Demo\clientpush_with_firebase\source\CLIENTPUSHWITHFIREBASE.DPR
- PROJECT GROUP
  - DelphiDay2019\Talk1\Demo\DELPHIDAY2019.groupproj

CODICE PER INVIO DELLE PUSH

- Questo è il metodo serve per inviare le push direttamente parlando con FIREBASE

```
procedure TForm23.SendPushAndroid(aTitle: string; aBody: string);
begin
  LIdHTTP := TIdHTTP.Create(nil);
  try
    LIdIOHandler := TIdSSLIOHandlerSocketOpenSSL.Create(LIdHTTP);
    LIdIOHandler.SSLOptions.Method := sslvSSLv23;
    LIdHTTP.IOHandler := LIdIOHandler;
    LIdHTTP.Request.ContentType := 'application/json';
    LIdHTTP.Request.Charset := 'UTF-8';
    // IMPOSTARE LA CHIAVE DI AUTORIZZAZIONE PER PARLARE CON FIREBASE
    // LA CHIAVE SI RECUPERA DAL PROGETTO CREATO IN FIREBASE
    LIdHTTP.Request.CustomHeaders.Values['Authorization'] := 'key='+FIREBASE_AUTH_KEY;

    jsonString := '{"to" : "'+AndroidTokenEdit.Text+'",' +
                  '"data" : {' +
                   '"gcm.notification.body" : "'+aBody+'",' +
                   '"colore" : "'+'rosso'+'",' +
                   '"gcm.notification.title" : "'+aTitle+'"}}';

    JsonToSend := TStringStream.Create(jsonString, TEncoding.UTF8);
    try
      response := LIdHTTP.Post('https://fcm.googleapis.com/fcm/send', JsonToSend);
      response := response.Replace(#10, '');
      Memo1.Lines.Insert(0, response);
    except
      on E: EIdHTTPProtocolException do response := e.ErrorMessage; end
  finally
    LIdHTTP.Free;
  end;
end;
```

DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

GRAZIE !

wintech italia

embarcadero reseller partner


That's all Folks!