Mobiele Apps 2



Michiel Vanbergen

ChitChat

[Inleiding 2](#_Toc504333666)

[Opdrachtomschrijving 2](#_Toc504333667)

[Werking van de App 3](#_Toc504333668)

[Pagina 1 – Het Login venster 3](#_Toc504333669)

[achterliggende code 4](#_Toc504333670)

[Pagina 2 – Het Registratie venster 5](#_Toc504333671)

[Pagina 3 – Het Gebruikersnaam venster 6](#_Toc504333672)

[Pagina 4 – Het Gesprekken venster 7](#_Toc504333673)

[achterliggende code 8](#_Toc504333674)

[Pagina 5 – Het Gesprek venster 9](#_Toc504333675)

[achterliggende code 10](#_Toc504333676)

[Pagina 6 – Het Scan venster 12](#_Toc504333677)

[achterliggende code 13](#_Toc504333678)

[Pagina 7 – Het Code venster 15](#_Toc504333679)

[Werking van de services 16](#_Toc504333680)

[Service 1 – De Notification service 16](#_Toc504333681)

[achterliggende code 17](#_Toc504333682)

[Service 2 – De Widget 19](#_Toc504333683)

[achterliggende code 20](#_Toc504333684)

# Inleiding

Voor het vak Mobiele Apps 2 in de derde fase van de opleiding professionele bachelor elektronica-ICT keuzerichting ICT moesten wij, de studenten, als eindproef zelf een app ontwikkelen. Hier kwam bij kijken dat de studenten in staat moesten zijn om van concept tot eindproduct zelf een mobiele applicatie te ontwikkelen voor het Android platform. Dit hield in kunnen werken met de Android Studio IDE, en bovenop de basis Android ontwikkeling die ons aangeleerd werd tijdens de contacturen zelf in staat zijn zich te verdiepen in nieuwe onbehandelde technieken om deze te leren gebruiken en integreren in de applicatie op zelfstandige basis. In dit dossier bespreek ik het eindresultaat van dit project, de chat-applicatie ChitChat.

## Opdrachtomschrijving

Ik heb ervoor gekozen om een chat-applicatie (ChitChat) te ontwikkelen.

De bedoeling van de applicatie is dat gebruikers na het installeren van de app zich kunnen registreren aan de hand van hun e-mailadres en een wachtwoord naar keuze. Vervolgens zouden ze een gebruikersnaam moeten opgeven om zich te identificeren. Als dit afgerond is, is de gebruiker vrij om de verschillende functionaliteiten te benutten. Zo kan de gebruiker:

* Zijn username wijzigen
* Een QR-code generen op basis van een unieke ID
* QR-codes scannen van andere gebruikers om met deze een gesprek te beginnen
* Een gesprek uitkiezen uit een lijst van bestaande gesprekken
* Berichten versturen en ontvangen binnen specifieke gesprekken
* Zich afmelden

Bovenop deze functionaliteiten die directe interactie met de gebruiker vereisen, krijgt de gebruiker ook nog meldingen wanneer er binnen een bestaand gesprek een nieuw bericht wordt ontvangen en kan de gebruiker zijn unieke QR-code ook weergeven op het thuis-scherm van zijn Android toestel met behulp van een widget.

Externe technologieën die ik hiervoor ga gebruiken zijn:

* Firebase

<https://firebase.google.com/>

* ZXing barcodescanner

<https://github.com/zxing/zxing>

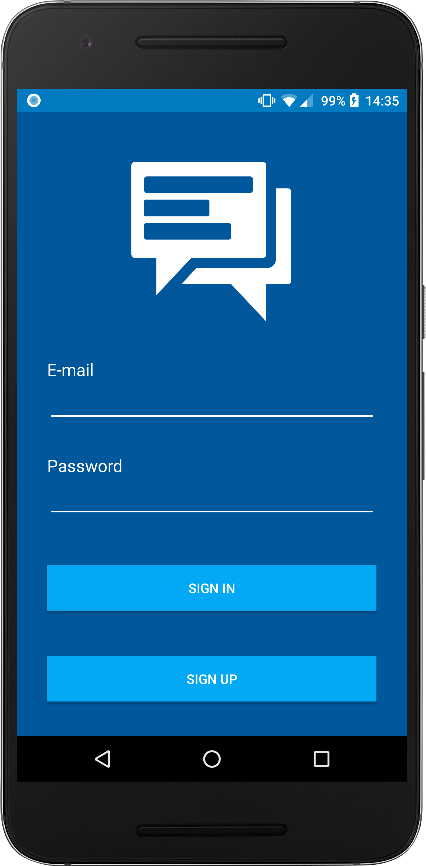
* QRGen

<https://github.com/kenglxn/QRGen>

# Werking van de App

Hier ga ik in het kort voor elke pagina van de applicatie de functionaliteit overlopen om deze te plaatsen binnen de algemene werking van de applicatie.

## Pagina 1 – Het Login venster



Wanneer de gebruiker de applicatie opent door middel van het icoon ( Figuur 1 ) aan te klikken is het Login venster ( Figuur 2 ) het eerste scherm dat de gebruiker zal zien.

Figuur 1

Figuur 2

Hier kan de gebruiker indien deze al geregistreerd is zich aanmelden met zijn e-mailadres en zijn wachtwoord. Voordat de gebruiker dan toegang krijgt tot de MainActivity kijkt de app na of de gebruiker al een geldige gebruikersnaam heeft.

Indien de applicatie een gebruikersnaam terugvindt verwijst deze de gebruiker door naar de MainActivity.

Indien dit niet het geval is verwijst hij de gebruiker door naar het UserName venster.

Indien de gebruiker nog geen ChitChat account heeft kan deze doorgaan naar het Registratie venster met de Sign Up knop.

### achterliggende code

Zoals vermeld in de pagina omschrijving van het Login venster moet nadat de gebruiker aangemeld is met firebase authenticatie de app nagaan of deze reeds over een geldige gebruikersnaam beschikt.

Als de gebruiker over een geldige gebruikersnaam beschikt zal de applicatie verdergaan door de MainActivity voor hem te openen.

Indien dit niet het geval is moet de app dit opvangen en de gebruiker doorsturen naar de UserName activity.

In de code doen we dit door zodra de gebruiker geauthenticeerd is uit de database zijn profiel, bestaande uit de gebruikers voor en achternaam, op te vragen op het adres “profile/” + de gebruikers UID ( Unique ID ).

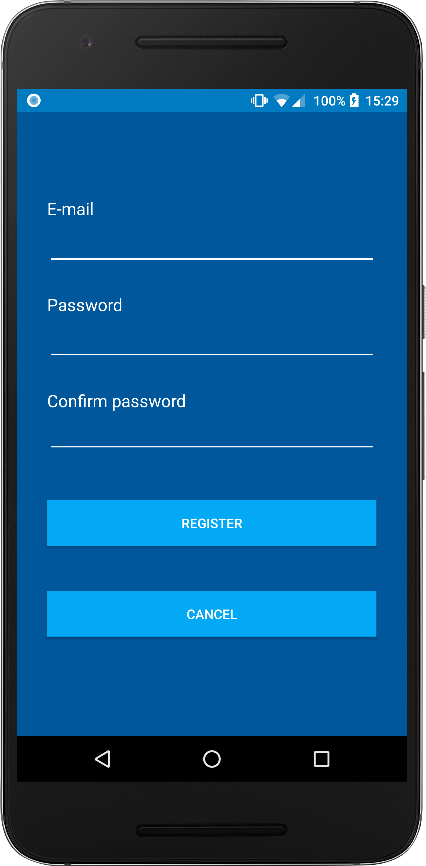
Indien de app geen profiel terugvindt of wel een profiel terugvindt maar merkt dat deze geen firstName en lastName property bevat zal hij de UserName Activity openen.

Indien hij wel een profiel terugvindt en dit beschikt over een firstName en een lastName property zal de app volgens de normale gang van zaken de MainActivity openen voor de gebruiker.

**database** = FirebaseDatabase.*getInstance*();  
  
**UID** = (String) **mAuth**.getCurrentUser().getUid();  
  
**myProfileRef** = **database**.getReference(**"profile/"** + **UID**);  
**myProfileRef**.addValueEventListener(**new** ValueEventListener() {  
  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()) {  
 **final** Map data = (Map) dataSnapshot.getValue();  
 **if**(data.containsKey(**"firstName"**) && data.containsKey(**"lastName"**)) {  
 Intent myIntent = **new** Intent(LoginActivity.**this**, MainActivity.**class**);  
 LoginActivity.**this**.startActivity(myIntent);  
 } **else** {  
 Intent myIntent = **new** Intent(LoginActivity.**this**, UserNameActivity.**class**);  
 LoginActivity.**this**.startActivity(myIntent);  
 }  
 } **else** {  
 Intent myIntent = **new** Intent(LoginActivity.**this**, UserNameActivity.**class**);  
 LoginActivity.**this**.startActivity(myIntent);  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
});

1 - LoginActivity Code

## Pagina 2 – Het Registratie venster



In het registratie venster ( Figuur 3 ) is de gebruiker in staat een account aan te maken door een e-mailadres en een wachtwoord op te geven.

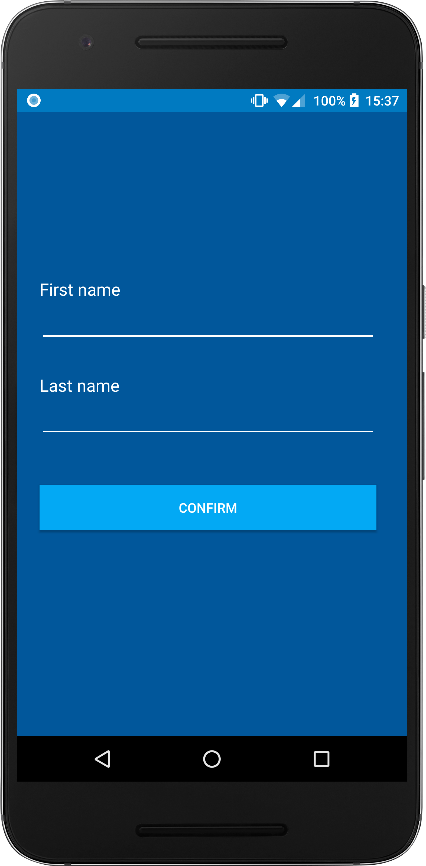
De gebruiker wordt gevraagd het wachtwoord twee maal op te geven. Deze twee wachtwoorden worden dan met elkaar vergeleken om er zeker van te zijn dat de gebruiker tijdens de registratie geen typfouten gemaakt heeft waardoor hij later niet in staat zou zijn om aan te melden op zijn account.

Moest er een fout zijn gemaakt in het invullen van het registratieformulier ( bv. het e-mailadres is al in gebruik of de wachtwoorden komen niet overeen ) Dan zal de app de gebruiker hiervan op de hoogte brengen via een toast melding.

Wanneer de registratie van de gebruiker via Firebase dan succesvol is afgerond zal de applicatie de UserName Activity openen.

Figuur 3

## Pagina 3 – Het Gebruikersnaam venster



Vooraleer de gebruiker volledig gebruik kan maken van de app is het noodzakelijk dat deze een gebruikersnaam opgeeft.

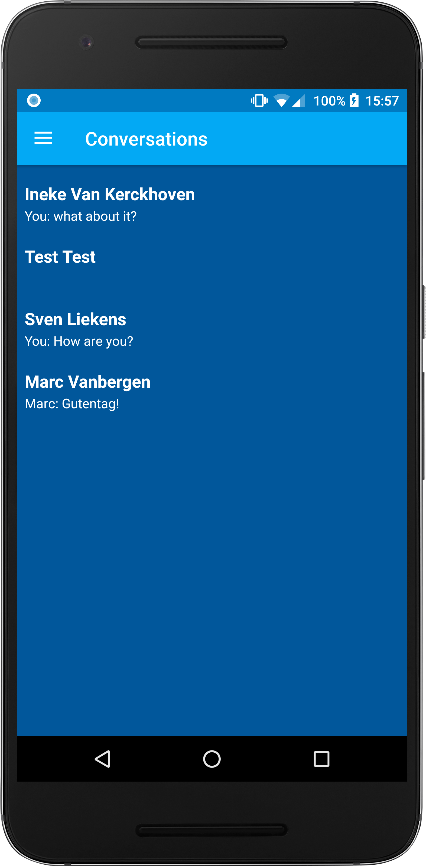
Om deze hiertoe te verplichten zal zowel bij het registreren als het aanmelden het UserName venster ( Figuur 4 ) aan de gebruiker getoond worden zolang deze geen geldige gebruikersnaam heeft ingegeven.

Eens de gebruiker een geldige naam heeft ingegeven ( niet leeg of enkel spaties ) kan deze op confirm klikken.

Dan wordt de naam opgeslagen onder de gebruiker zijn profiel op Firebase’s realtime database en wordt de gebruiker doorverwezen naar de MainActivity.

Figuur 4

## Pagina 4 – Het Gesprekken venster



Het Gesprekken venster ( Figuur 5 ), doorheen dit dossier ook naar verwezen als de MainActivity, is het middelpunt van de applicatie.

Het is vanaf hier dat de gebruiker in staat is naar alle andere kernfunctionaliteiten van de app te navigeren door middel van het menu ( Figuur 6 ).

In dit menu kan de gebruiker ervoor kiezen te navigeren naar:

* Edit username
* Scan code
* Show code
* Sign Out

Over deze pagina’s, met uitzondering van de username pagina die we eerder dit dossier behandeld hebben, later meer.

Het is ook hier dat de gebruiker een opsomming krijgt van alle bestaande gesprekken ( Figuur 5 ).

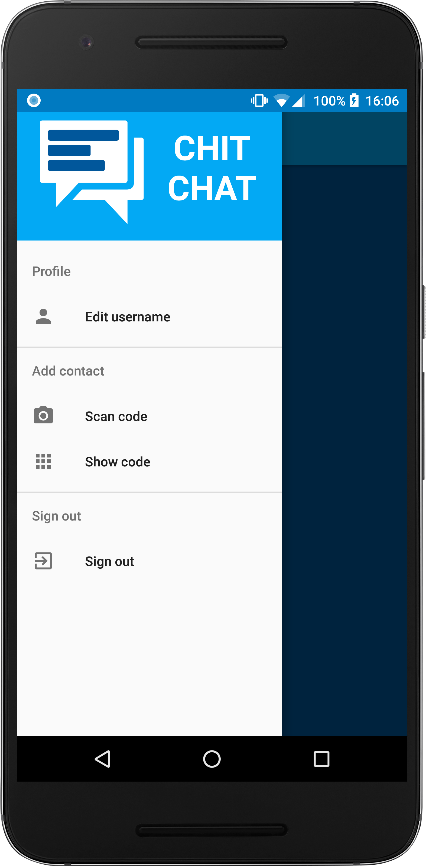
Voor elk gesprek worden er verschillende gegevens weergegeven:

Figuur 5

* De contactpersoon
* Wie het laatste bericht heeft verstuurd
* De inhoud van het laatste bericht

Op de achtergrond houdt deze lijst ook voor elk bestaand gesprek de unieke gesprekssleutel bij zodat wanneer een gesprek wordt uitgekozen om te openen de app weet voor welk gesprek hij voorgaande berichten uit de database moet ophalen.

Tot slot wordt op deze pagina ook een STICKY achtergrond service gestart die meldingen voor deze applicatie zal beheren zodat de gebruiker op de hoogte wordt gebracht wanneer deze een nieuw bericht ontvangt. Ongeacht of de app geopend of gesloten is op het toestel.



Figuur 6

### achterliggende code

Zoals vermeld in de pagina omschrijving van het Gesprekken venster toont het Gesprekken venster een lijst van alle bestaande gesprekken aan de gebruiker.

In deze weergave moet de gebruiker in staat zijn snel relevante informatie terug te vinden zodat hij niet lang moet zoeken naar het gewenste gesprek.

Om dit voor elkaar te krijgen heb ik een eigen klasse ‘Contact’ geschreven.

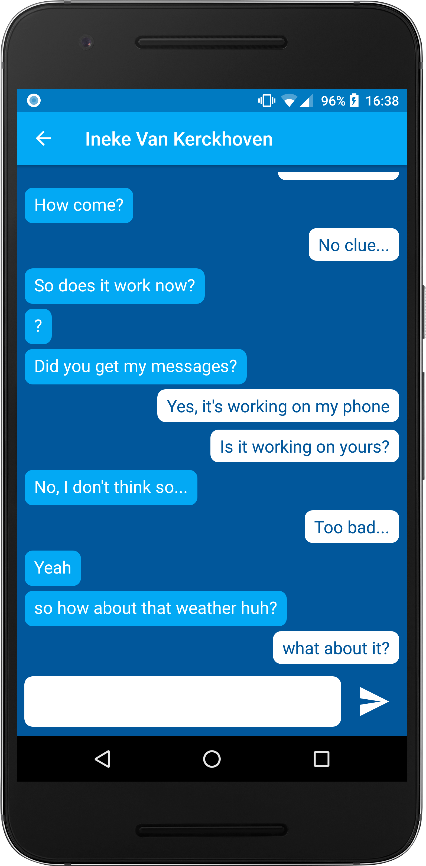
Deze klasse is een verzamelplaats voor relevante gegevens voor één gesprek die worden verzameld van op verschillende plaatsen in Firebase’s realtime database.

Het zijn objecten van deze klasse die weergegeven worden in de ListView op de MainActivity.

2 - Klasse Contact

**public class** Contact {  
 **private** String **contactKey** = **null**;  
 **private** String **name** = **null**;  
 **private** String **lastMessage** = **null**;  
 **private** String **conversationKey** = **null**;  
  
 **public void** setContactKey(String contactKey){  
 **this**.**contactKey** = contactKey;  
 }  
 **public** String getContactKey(){  
 **return contactKey**;  
 }  
  
 **public void** setName(String name){  
 **this**.**name** = name;  
 }  
 **public** String getName(){  
 **return name**;  
 }  
  
 **public void** setLastMessage(String lastMessage){  
 **this**.**lastMessage** = lastMessage;  
 }  
 **public** String getLastMessage(){  
 **return lastMessage**;  
 }  
  
 **public void** setConversationKey(String conversationKey) {  
 **this**.**conversationKey** = conversationKey;  
 }  
 **public** String getConversationKey(){  
 **return conversationKey**;  
 }  
  
}

## Pagina 5 – Het Gesprek venster



Het Gesprek venster ( Figuur 7 ), laat de gebruiker toe een gesprek te voeren met één specifiek contactpersoon.

Er bestaat hier een duidelijk onderscheid tussen ontvangen en verzonden berichten.

Ontvangen berichten hebben een witte tekst op een lichtblauwe achtergrond en worden links uitgelijnd.

Verzonden berichten hebben een donkerblauwe tekst op een witte achtergrond en worden rechts uitgelijnd.

Dit allemaal in een chronologische lijst van berichten waar de gebruiker doorheen kan navigeren door middel van scrollen.

Wanneer de gebruiker een nieuw bericht verstuurt of ontvangt zal dit onderaan de lijst van berichten worden toegevoegd.

Figuur 7

### achterliggende code

Desondanks dat de Gesprek pagina qua functionaliteit niet zeer moeilijk was, met name het uitlezen van bericht met de UID van hun zender uit Firebase’s realtime database, was het niet vanzelfsprekend om dit weer te geven aan de gebruiker.

Om het gesprek op een intuïtieve manier weer te geven aan de gebruiker moest ik een eigen ArrayAdapter ‘ConversationArrayAdapter’ uitwerken die elk object van de klasse ‘Message’ kon evalueren en onderscheid kon maken of dit bericht verstuurd of ontvangen is. Op basis hiervan kon deze adapter dan één van de twee speciaal hiervoor voorziene layouts ( ‘received\_message.xml’ en ‘sent\_message.xml’ ) toekennen aan het bericht alvorens het deze weergaf in de listview.

3 - Klasse Message

**public class** Message {  
 **public boolean sent**;  
 **public** String **message**;  
  
 **public** Message(**boolean** sent, String message) {  
 **super**();  
 **this**.**sent** = sent;  
 **this**.**message** = message;  
 }  
}

4 - ConversationArrayAdapter Code

**public** View getView(**int** position, View convertView, ViewGroup parent) {  
 Message messageObject = getItem(position);  
 View row = convertView;  
 LayoutInflater inflater = (LayoutInflater) **this**.getContext().getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***);  
 **if** (messageObject.**sent**) {  
 row = inflater.inflate(R.layout.***sent\_message***, parent, **false**);  
 }**else**{  
 row = inflater.inflate(R.layout.***received\_message***, parent, **false**);  
 }  
 **chatText** = (TextView) row.findViewById(R.id.***msgr***);  
 **chatText**.setText(messageObject.**message**);  
 **return** row;  
}

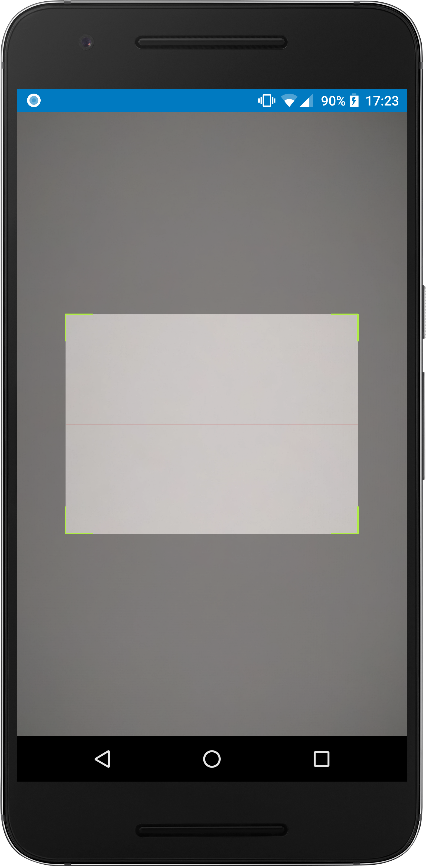
5 - Layout Received\_message

<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**TextView  
 android:id="@+id/msgr"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="left"  
 android:layout\_marginBottom="5dp"  
 android:layout\_marginRight="20dp"  
 android:background="@drawable/message\_received"  
 android:paddingBottom="8dp"  
 android:paddingLeft="10dp"  
 android:paddingRight="10dp"  
 android:paddingTop="5dp"  
 android:text="Sampleleft"  
 android:textColor="#ffffff"  
 android:textSize="18sp"** />  
</**RelativeLayout**>

6 - Layout Sent\_Message

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"**>  
  
 <**TextView  
 android:id="@+id/msgr"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="right"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginLeft="20dp"  
 android:background="@drawable/message\_sent"  
 android:paddingBottom="5dp"  
 android:paddingLeft="10dp"  
 android:paddingRight="10dp"  
 android:paddingTop="5dp"  
 android:text="Sample"  
 android:textColor="#01579b"  
 android:textSize="18sp"** />  
</**LinearLayout**>

## Pagina 6 – Het Scan venster



Wanneer vanuit het menu dat zich bevindt in de MainActivity de scan pagina wordt geopend, wordt deze geopend met het specifieke doel een UID uit te lezen uit een QR-code en deze terug te spelen naar de MainActivity.

Indien het de eerste keer is dat de gebruiker deze pagina opent vraagt de applicatie de gebruiker eerst om toestemming om de camera op het toestel te gebruiken.

Zodra de app dan toestemming krijgt van de gebruiker opent deze de camera feed en begint deze te zoeken naar een geldige code.

Zodra deze een geldige code vindt scant hij deze en haalt hij hier de waarde uit. Deze speelt hij door naar de MainActivity waar wordt gekeken of deze code overeenkomt met een reeds bestaand gesprek.

Vindt de applicatie deze code terug, dan opent hij dit reeds bestaande gesprek.

Figuur 8

Vindt de applicatie geen bestaand gesprek met deze UID als contactpersoon, dan kijkt hij na of deze UID ook echt tot een gebruiker toebehoort en indien dit klopt begint hij een nieuw gesprek dat hij koppelt aan zowel de gebruiker zijn account als de contactpersoon zijn account.

Het kan ook voor komen dat de gebruiker gewoon wenst deze pagina te verlaten. De gebruiker kan dan op het back pijltje klikken waarna er aan de MainActivity zal worden doorgegeven dat de scanner geen code heeft gevonden.

In dit geval gebeurt er verder niks.

### achterliggende code

De functionaliteit van het Scan venster, voor het scannen van QR-codes, is gebaseerd op het gebruik van de ZXing barcodescanner bibliotheek.

De belangrijkste code geassocieerd met deze activity bevindt zich dan ook niet in de ScannerActivity zelf maar in de MainActivity, waar de ScannerActivity de gegevens die hij uitleest uit een QR-code naar doorspeelt.

Dit doen we door ScannerActivity op te roepen met het oog op een result.

Indien de ScannerActivity in zijn opzet slaagt geeft hij de UID die hij heeft uitgelezen en een melding dat hij in zijn opzet geslaagd is terug door aan de MainActivity.

Ook indien de ScannerActivity er niet slaagt in het gewenste resultaat te verkrijgen geeft deze een melding terug aan de MainActivity, deze keer om te melden dat hij niet het gewenste resultaat heeft bekomen.

7 - MainActivity Code ( oproepen ScannerActivity )

**if** (id == R.id.***nav\_camera***) {  
 Intent myIntent = **new** Intent(MainActivity.**this**, ScannerActivity.**class**);  
 MainActivity.**this**.startActivityForResult(myIntent, 0);  
}

@Override  
**public void** onBackPressed() {  
 setResult(-1);  
 finish();  
 **scannerView**.stopCameraPreview();  
 **scannerView**.stopCamera();  
}

8 - ScannerActivity Code ( Mislukt )

@Override  
**public void** handleResult(Result result) {  
 String contactKey = result.getText();  
 setResult(0, **new** Intent().putExtra(**"contactKey"**, contactKey));  
 finish();  
 **scannerView**.stopCameraPreview();  
 **scannerView**.stopCamera();  
}

9 - Scanneractivity code ( gelukt )

10 - MainActivity code ( verwerken ScannerActivity resultaat )

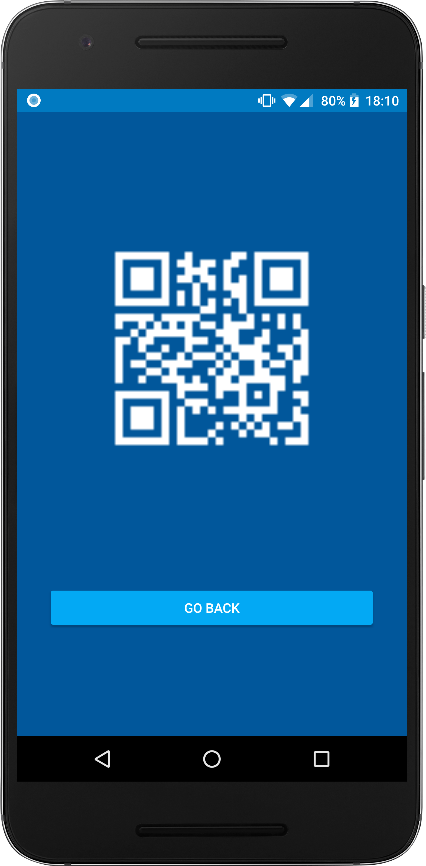
@Override  
**protected void** onActivityResult(**int** requestCode,

**int** resultCode,

Intent data) {  
 **if**(requestCode == resultCode){

…  
 }  
}

## Pagina 7 – Het Code venster



Het Code venster ( Figuur 9 ), genereert een QR-code op basis van de gebruikers UID.

De gebruiker kan deze dan laten scannen door een andere gebruiker die de scan functionaliteit van de app aan het gebruiken is om met deze een nieuw gesprek te beginnen.

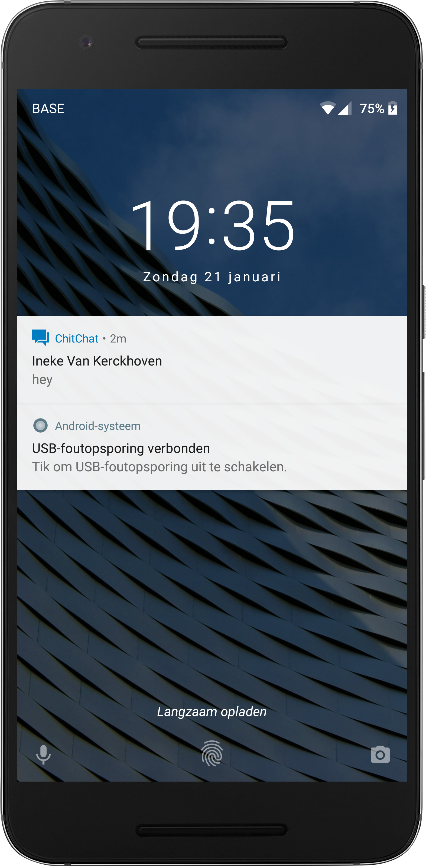
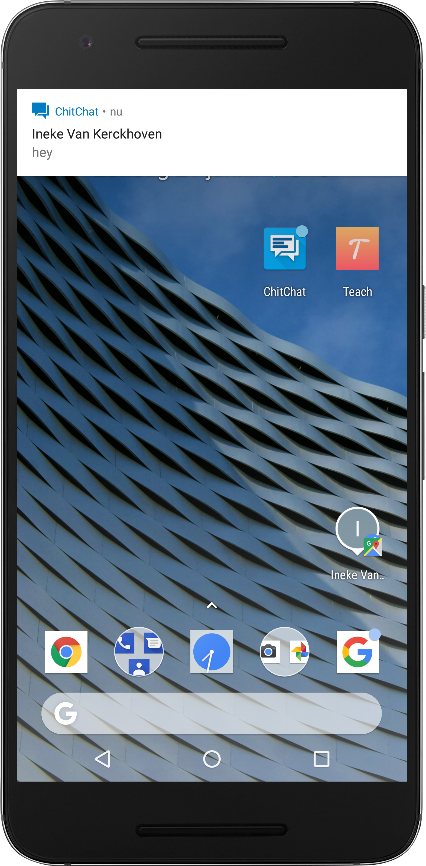
De functionaliteit voor dit venster is voornamelijk voorzien door de QRGen bibliotheek. Het is om deze reden dat ik niet dieper inga op de achterliggende code van deze pagina.

Figuur 9

# Werking van de services

Hier ga ik in het kort voor elke service die op de achtergrond functionaliteit aan de app levert hun werking overlopen om hun rol binnen de algehele functionaliteit van de applicatie duidelijk te maken.

## Service 1 – De Notification service



Figuur 12

Figuur 10

Figuur 11

De Notification service is een STICKY ( blijft uitvoeren na het sluiten van de app ) background service die wanneer de gebruiker een nieuw bericht ontvangt deze op de hoogte brengt door middel van een melding.

Er is dan een visuele melding terug te vinden op het vergrendelingsscherm ( Figuur 10 ), de melding zone van de status bar ( Figuur 11 ) en bij het app-icoontje ( Figuur 12 ).

Deze visuele melding gaat gepaard met een geluid om de aandacht te trekken, het flikkeren van een led lampje op de gsm indien aanwezig, en het trillen van de gsm als haptische feedback.

Wanneer de gebruiker van ergens op zijn toestel de melding aanklikt opent dit de MainActivity, de lijst van alle gesprekken waar de gebruiker aan deelneemt.

### achterliggende code

Zoals vermeld in de service omschrijving van de Notification service luistert de Notification service naar nieuwe berichten. Wanneer deze een nieuw bericht opmerkt zal deze een melding genereren om aan de gebruiker voor te leggen.

Om dit te kunnen doen moeten we in de NotificationService een notificationManager definiëren. Hierin stellen we standaard instellingen die overeenkomen voor alle meldingen in. Voorbeelden hiervan zijn:

* Het geluid van de melding
* De kleur van het knipperend LED licht
* De trilling van de melding
* De beschrijving van de melding

Vervolgens moeten we wanneer er een nieuw bericht is binnengekomen de werkelijke melding genereren. Dit doen we met de NotificationBuilder waaraan we melding specifieke waarden meegeven. Voorbeelden van deze specifieke gegevens zijn:

* Het icoon van de melding
* De titel van de melding
* De inhoud van de melding
* De kleur van de melding

Als we dan de notification die we net opgebouwd hebben doorgeven aan de NotificationManager ontvangt de gebruiker de melding dat er een nieuw bericht is binnengekomen.

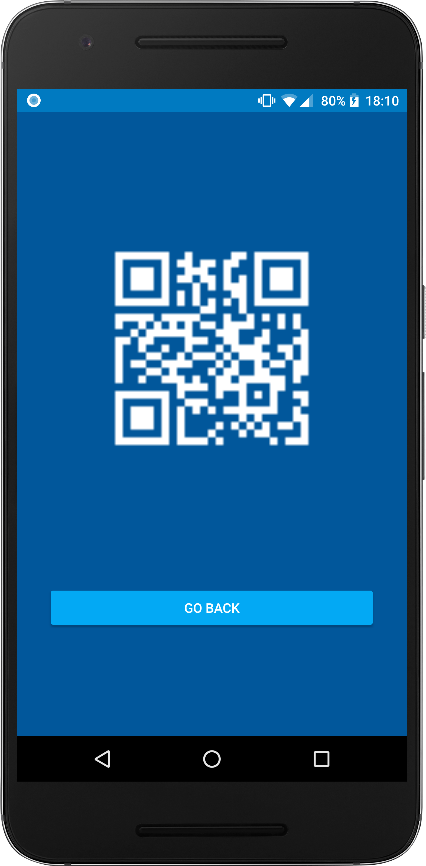
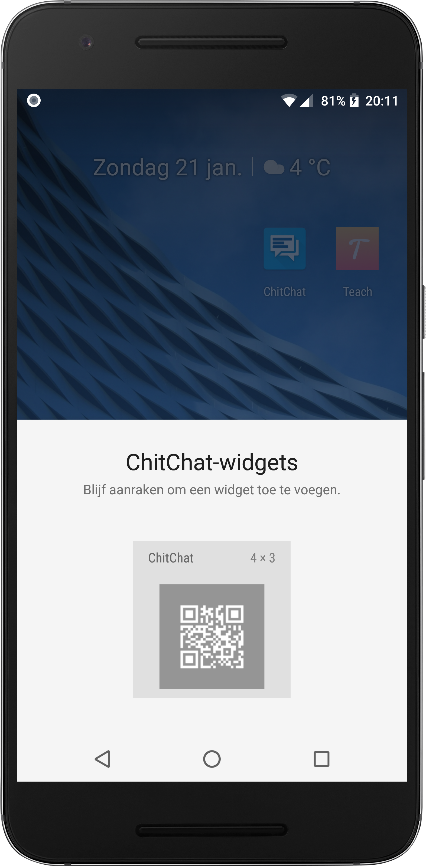
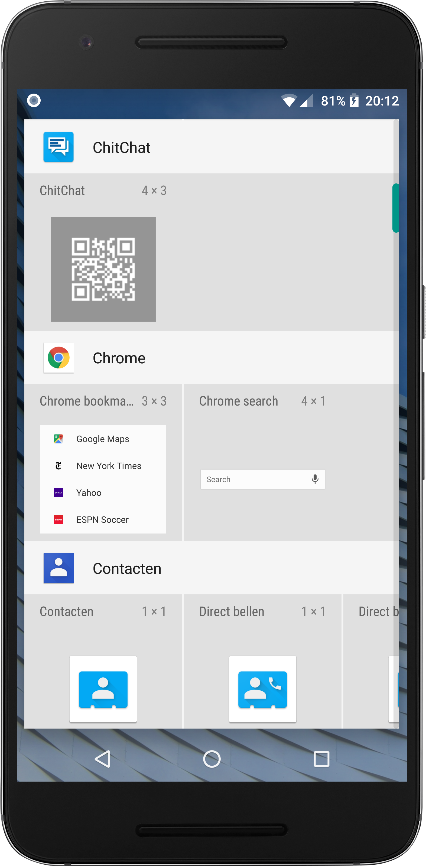
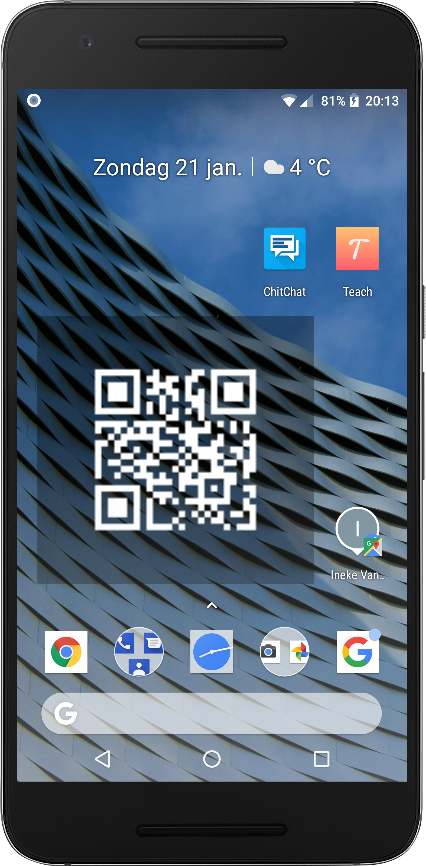
11 - Notificationservice code ( Definiëren notificationchannel )

**notificationManager** = (NotificationManager) getSystemService(Context.***NOTIFICATION\_SERVICE***);  
**notificationChannelId** = **"notification\_channel"**;  
  
**notificationChannel** = **new** NotificationChannel (**notificationChannelId**, **"notifications"**, NotificationManager.***IMPORTANCE\_HIGH***);  
  
**notificationChannel**.setDescription(**"message notifications"**);  
**notificationChannel**.enableLights(**true**);  
  
**notificationChannel**.setLightColor(Color.***BLUE***);  
**notificationChannel**.setVibrationPattern(**new long**[] {0, 250, 250, 250});  
  
**notificationManager**.createNotificationChannel(**notificationChannel**);

12 - NOTIFICATIONSERVICE CODE ( Opbouwen melding )

NotificationCompat.Builder notificationBuilder = **new** NotificationCompat.Builder(**this**, **notificationChannelId**)  
 .setSmallIcon(R.drawable.***ic\_logo***)  
 .setContentTitle(title)  
 .setContentText(content)  
 .setAutoCancel(**true**)  
 .setColor(0xFF03a9f4);  
Intent resultIntent = **new** Intent(**this**, LoginActivity.**class**);  
  
TaskStackBuilder stackBuilder = TaskStackBuilder.*create*(**this**);  
  
stackBuilder.addParentStack(LoginActivity.**class**);  
  
stackBuilder.addNextIntent(resultIntent);  
PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0, PendingIntent.***FLAG\_UPDATE\_CURRENT***);  
  
notificationBuilder.setContentIntent(resultPendingIntent);  
  
**notificationManager**.notify(notificationId, notificationBuilder.build());

## Service 2 – De Widget



Figuur 15

Figuur 13

Figuur 14

De widget die ik gekozen heb om te maken voor de applicatie is de weergave van de gebruikers QR-code buiten de app ( Figuur 15 ).

De gebruiker kan deze widget toevoegen aan het start scherm van zijn toestel door het icoon van de app ingedrukt te houden en widgets aan te klikken ( Figuur 13 ) of door het start scherm ingedrukt te houden, widgets te selecteren en naar ChitChat te navigeren.

### achterliggende code

Net als bij het code venster is een groot deel van de achterliggende functionaliteit voor de Widget voorzien door de QRGen bibliotheek. Om dit in een widget te verwerken moest ik een klasse ‘WidgetProvider’ maken die afgeleid is van de klasse ‘AppWidgetProvider’.

13 - Klasse widgetprovider

**public class** WidgetProvider **extends** AppWidgetProvider {  
  
 **private** SharedPreferences **savedValues**;  
  
 **private** String **UID**;  
  
 **private** ImageView **QrCodeImageView**;  
  
 @Override  
  
 **public void** onUpdate

(Context context, AppWidgetManager appWidgetManager, **int**[] appWidgetIds) {

**final int** N = appWidgetIds.**length**;  
  
 **for** (**int** i=0; i<N; i++) {  
 **int** appWidgetId = appWidgetIds[i];  
  
 Intent intent = **new** Intent(context, QrCodeActivity.**class**);

PendingIntent pendingIntent = PendingIntent.*getActivity*

(context, 0, intent, 0);  
  
 RemoteViews views = **new** RemoteViews

(context.getPackageName(), R.layout.***widget***);

**savedValues** = context.getSharedPreferences

(**"SavedValues"**, ***MODE\_PRIVATE***);

**UID** = **savedValues**.getString(**"UID"**, **""**);  
  
 Bitmap myQrCode = QRCode.*from*(**UID**).withColor

(0xFFFFFFFF, 0x00ff0000).bitmap();

views.setImageViewBitmap(R.id.***QrCodeImageView***, myQrCode);  
  
 appWidgetManager.updateAppWidget(appWidgetId, views);  
 }  
 }  
}