# BroodCast voor iedereen

Eindverslag R&D

Bart van de Put s4400496
Tom Nikken s4229649
Thomas de Bel s3032817
Ronnie Swanink s4382838

Juni 2014

## Inhoudsopgave

# 1 Voorwoord

In dit document wordt de uiteindelijke opbouw en het ontwerp van de app 'BroodCast' behandeld, en wordt een reflectie gegeven over hoe het ontwikkelproces is verlopen. Het doel is om de app in zowel functionele als technische vorm te beschrijven, en een verantwoording te geven voor de keuzes die zijn gemaakt tijdens het project. Verder wordt er door de groepsleden een reflectie gedaan op het project.

Dit document is opgebouwd uit drie verschillende stukken: Beschrijving, Ontwerp en Reflectie. In de beschrijving wordt gekeken naar wat de app nou eigenlijk doet, en wordt de functionele beschrijving gegeven. Bij het ontwerp wordt er ingegaan op de technische kant van de app. Hier wordt de opbouw van de app uitgezet en wordt er verantwoord waarom alles op deze manier is opgezet. Als laatste is er de reflectie, hierin worden de ervaringen van de groepsleden beschreven.

# 2 Beschrijving

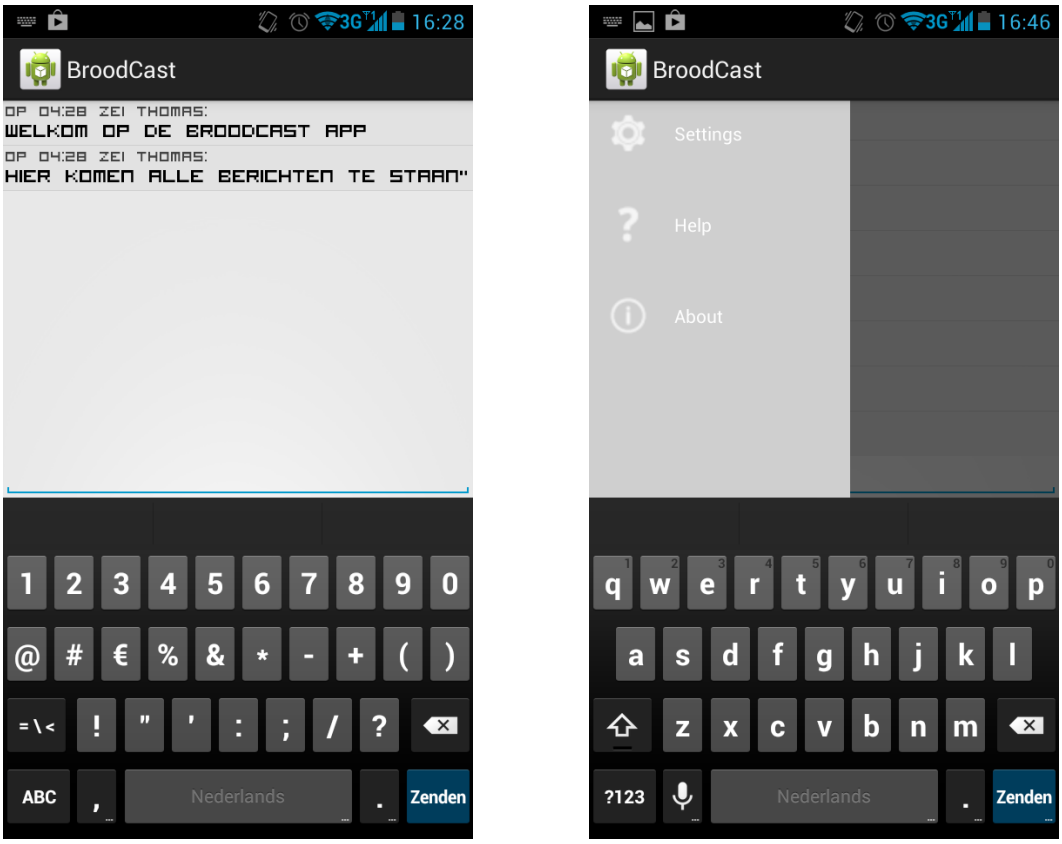
In dit onderdeel wordt ingegaan op de beschrijving van de app zelf. Onder 'Inleiding' is een algemenen beschrijving van de app gegeven. Verder is er onder 'Productverantwoording' een verantwoording gegeven van de app, en worden alternatieven gegeven. Als laatste is er onder 'Specificaties' een beschrijving van eigenschappen te vinden.

## 2.1 Inleiding


Figuur 1: *Dit scherm wordt getoond wanneer je de app voor het eerst opent.*

De app die wij hebben gemaakt, genaamd de Broodcast app, is een chatapp waarmee je kunt chatten over een netwerk. Bij het eerste gebruik verschijnt er eerst een welkomstscherm zoals in de afbeelding hierboven te zien is. De gebruiker moet hier een gebruikersnaam kiezen. Deze gebruikersnaam zal de identificatie zijn naar andere gebruikers toe. Deze gebruiker kan deze gebruikersnaam vervolgens wijzigen in de instellingen van de app. Ook kan de gebruiker hier de port en het ip-adres waar berichten over worden gestuurd wijzigen. Het is echter aangeraden om deze op de standaardinstellingen te laten staan.

Na het instellen van een gebruikersnaam opent de app het chatscherm, wat in de afbeelding hieronder te zien is. Dit is een chatscherm zoals je dat zou verwachten: onderaan staat een tekstveld, waar de gebruiker een boodschap in kan tikken. Daarboven staat een lijst met de geschiedenis van verzonden en ontvangen berichten. Door van links naar rechts te schuiven, schuift een menu links in het beeld, een zogenaamde drawer. Hier kan de gebruiker verschillende opties kiezen, zoals aangegeven in onderstaande afbeelding. Kiest de gebruiker voor de optie 'Settings', dan opent een nieuwe scherm scherm waarin de gebruiker zijn gebruikersnaam kan veranderen waarin de gebruiker het ip-adres en de poort die in de app worden gebruikt, kan veranderen. Kiest men in de drawer de optie 'Help', dan wordt een scherm geopend waarin de gebruiker kan lezen wat hij kan doen als de app geen berichten




(a) (b)

Figuur 2: *Het scherm waarop alle chatberichten worden getoond en waar je zelf een bericht kan typen (a). Vanuit het chatscherm kan een menu worden geopend (b).*

verstuurt en ontvangt. Als de gebruiker in de drawer kiest voor de optie 'About', dan opent een scherm waarin de gebruiker de namen van de makers van de app kan bewonderen.

## 2.2 Productverantwoording

De app 'BroodCast' hebben we gemaakt met als doel een app te maken die iets nieuws doet. Nu is een chat app natuurlijk niet nieuw meer. Maar 'BroodCast' is een app waarbij je direct kunt chatten met iedereen in hetzelfde WiFi netwerk. Er zijn geen contactpersonen in de app, iedereen in het netwerk kan berichten ontvangen en versturen.

Er is een aantal apps die dit ook doet, maar deze doen allemaal toch niet precies wat wij in gedachten hadden voor de app. Zo heb je bijvoorbeeld de app: 'Wi-Fi Talkie'. Het chatten bij deze app werkt niet optimaal, als je een bericht verstuurd ontvang je ook weer datzelfde bericht. Zo zie je je eigen berichten dubbel. Verder richt deze app zich niet specifiek op chatten, iets wat we met 'BroodCast' wel wilden. Je kunt met 'Wi-Fi Talkie' ook bestanden versturen en bellen.

Dan is er de app "WifiChat", hierbij selecteer je een WiFi netwerk, waarna je doorgelinkt wordt naar een website waar het chatten dan plaatsvind. Bij 'BroodCast' vind het chatten in de app zelf plaats.

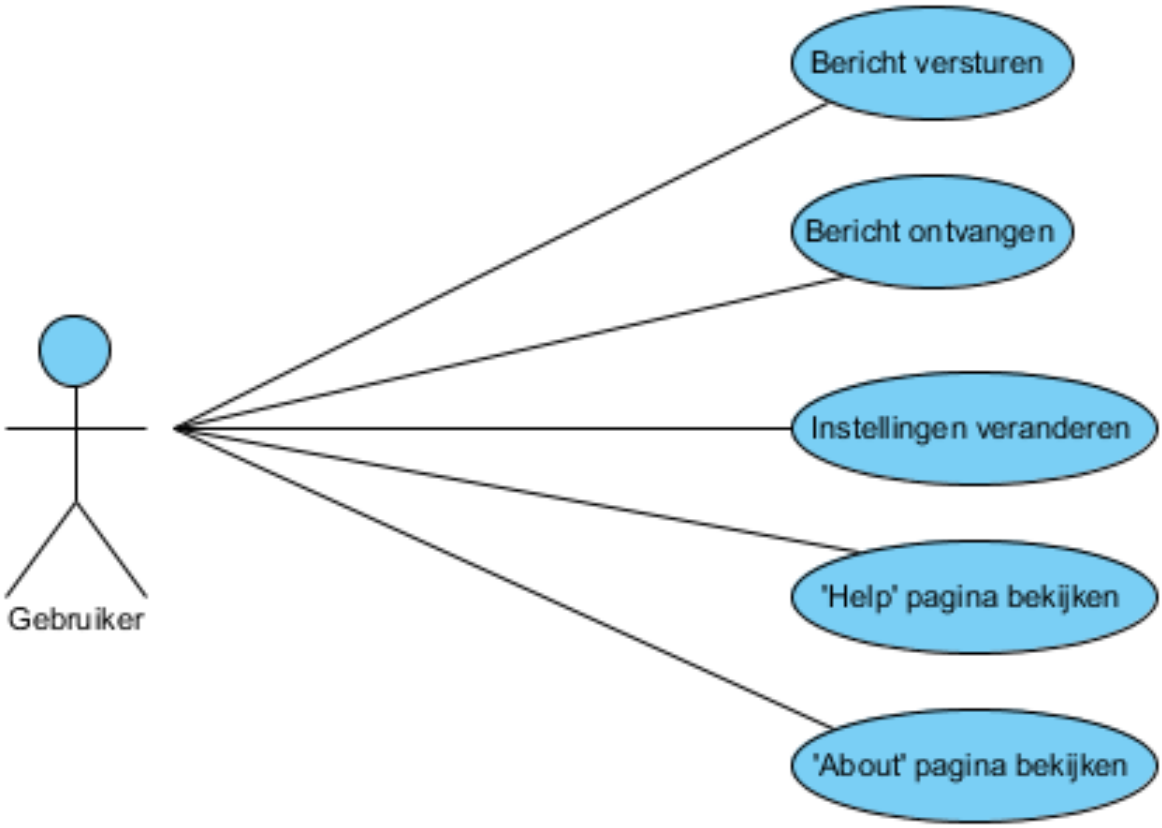
Als laatste vergelijkbare app is er nog "WiTalky". Maar deze app is ook weer meer gericht op het uitwisselen van bestanden dan echt chatten. Hierbij wordt dan ook echt een connectie gemaakt tussen de verschillende apparaten.

Met 'BroodCast' hebben we een chatapp waarmee je meteen kunt chatten, het is makkelijk te gebruiken omdat er niet allemaal extra (onnodige) functies zijn.

## 2.3 Specificaties

### 2.3.1 Functionele eigenschappen

Use case diagram:




Hieronder staat de uitgewerkte use case 'Bericht versturen'.

| Use Case1: | Bericht Verzenden |
|---|---|
| Use case diagram |  |
| Description | Het versturen van een bericht aan andere gebruikers. |
| Actor | Gebruiker |
| Trigger | De gebruiker wil een bericht versturen. |
| Basic course of events | 1. De gebruiker start de app op.<br>2. De app laat de chatbox zien.<br>3. De gebruiker typt zijn bericht in in het tekstveld.<br>4. De gebruiker drukt op de verzendknop.<br>5. De app verzend het bericht. |
| Alternate paths | |
| Exception paths | |
| Assumptions | • Er is verbinding met een wifi-netwerk.<br>• Het netwerk ondersteunt UDP multicasting. |
| Preconditions | |
| Postcondition | Het bericht dat door de gebruiker is getypt, is verstuurd |

### 2.3.2 Niet-functionele eigenschappen

- **Gebruikersgemak:** De app is erg makkelijk in gebruik.
- **Leerbaarheid:** Het gebruik van de app is makkelijk te leren.
- **Uiterlijk:** De app ziet er mooi uit en heeft een leuk lettertype.
- **Foutbestendigheid:** De app geeft bijna geen fouten.

# 3 Ontwerp

In dit onderdeel wordt ingegaan op de technische opbouw van de app. Onder 'Globaal ontwerp' wordt ingegaan op de functionele specificaties van de verschillende componenten die de app vormen, en onder 'Detailontwerp' wordt inzage gegeven in hoe deze functionele eisen in klassen en methoden zijn geïmplementeerd.

## 3.1 Globaal ontwerp

De app is opgebouwd uit meerdere delen. De belangrijkste componenten die samen de hele werking van de app verzorgen zijn de verschillende activities en de UDPMessenger. Helemaal aan de top staat de ChatActivity zelf, vanuit deze activity worden de andere activities gestart en deze stuurt ook de hele UDPMessenger aan. Verder is er de Messagebox voor de frontend en het verwerken van gebruikersinvoer en is er een FileHandler om alle berichten op te slaan op de telefoon.

### 3.1.1 ChatActivity

De ChatActivity is de activity die ook als eerste wordt gestart. De ChatActivity is verantwoordelijk voor het starten van de WelkomActivity als er nog geen gebruikersnaam bekend is. Als deze gegevens wel bekend zijn, dan maakt de ChatActivity een instantie van de UDPMessenger aan, of hij gebruikt een bestaande als deze al eerder was aangemaakt. Verder maakt hij een MessageBox aan zodat de gebruiker invoer kan geven en berichten kan lezen. Verder is de ChatActivity een doorvoerpunt voor events die door de UDPMessenger gegeven worden. De ChatActivity geeft alle events door aan zijn instantie van de MessageBox. De ChatActivity beschikt ook over een drawer, waarmee de Help-, About- en SettingsActivity kunnen worden gestart.

### 3.1.2 WelkomActivity

De WelkomActivity wordt gestart als er nog geen gegevens beschikbaar zijn, en kan niet op andere manieren worden geopend. De WelkomActivity geeft de gebruiker de mogelijkheid om een gebruikersnaam te kiezen en om eventueel een ander ip-adres of poort in te geven. Als de gebruiker akkoord gaat met de invoer slaat de WelkomActivity de voorkeuren op in de SharedPreferences van de app en wordt de ChatActivity nogmaals gestart.

### 3.1.3 AboutActivity

Deze activity bevat niet bijzonder veel extra functionaliteit. Een gebruiker kan hier zien wie de app gemaakt hebben.

### 3.1.4 HelpActivity

De HelpActivity geeft de gebruiker informatie in het geval dat de app niet doet wat het zou moeten doen. Ook hier is verder niks bijzonders.

### 3.1.5 SettingsActivity

De SettingsActivity is een activity die met behulp van de Preferences API van Android een settingsvenster toont. Hier kan de gebruiker zijn gebruikersnaam, het ip-adres en de poort die worden gebruikt wijzigen. De instellingen hier zijn dezelfde als die in de WelkomActivity worden gebruikt. Deze zijn globaal voor de gehele app.

### 3.1.6 MessageBox

De MessageBox verzorgt de front-end van de ChatActivity. De Messagebox is een fragment dat alle ruimte in het scherm inneemt. Onderaan de messagebox kan een gebruiker tekst invoeren om deze vervolgens via de zendtoets op het toetsenbord te versturen. Dit tekstveld staat relatief ten opzichte van de onderkant van het scherm, zodat het toetsenbord niet over andere elementen zit wanneer het tevoorschijn wordt gehaald. De MessageBox beschikt verder over een ListView waarin alle binnengekomen berichten worden weergegeven. Deze ListView wordt aangestuurd door op maat gemaakte ListAdapters. De MessageBox ontvangt alle events van de UDPMessenger via de ChatActivity. Als berichten die zijn gestuurd door deze MessageBox bevestiging krijgen dat deze wel al dan niet zijn aangekomen, wordt de achtergrondkleur van de view van deze berichten aangepast. Als de gebruiker een bericht verstuurt, stuurt de MessageBox deze direct via de UDPMessenger naar de andere gebruikers in het netwerk. Alle berichten worden door de MessageBox opgeslagen met de FileHandler. Bij het opstarten haalt de MessageBox alle oude berichten op via de FileHandler.

### 3.1.7 FileHandler

De FileHandler is een klasse die in staat is om willekeurige objecten die het Serializable interface implenteren op te slaan en weer terug te lezen. Intern wordt deze klasse vooral gebruikt om alle berichten tussentijds op te slaan in opdracht van de MessageBox.

### 3.1.8 UDPMessenger

De UDPMessenger is de klasse waar alle netwerkcode onder valt. In de app zelf wordt de UDPMessengerAndroid-klasse gebruikt om de UDPMessenger aan te maken, omdat er voor het gebruik van multicast op een Android-telefoon een lock nodig is. Deze lock wordt aangemaakt in de start- en stopmethoden van de Android-variant van de UDPMessenger.

Intern maakt de UDPMessenger twee timers en een thread aan om al zijn werk te kunnen doen. Verder slaat hij de instantie van zichzelf op in een statische variabele in de klasse zodat het opnieuw starten van de activity niet direct het einde van de threads oplevert. In de thread luistert de UDPMessenger constant naar nieuwe berichten op het netwerk, om deze bij binnenkomst om te zetten naar het bijpassende Packet-object. Deze pakketten worden daarna verwerkt. Als er berichten binnenkomen worden deze doorgegeven aan de ChatActivity.

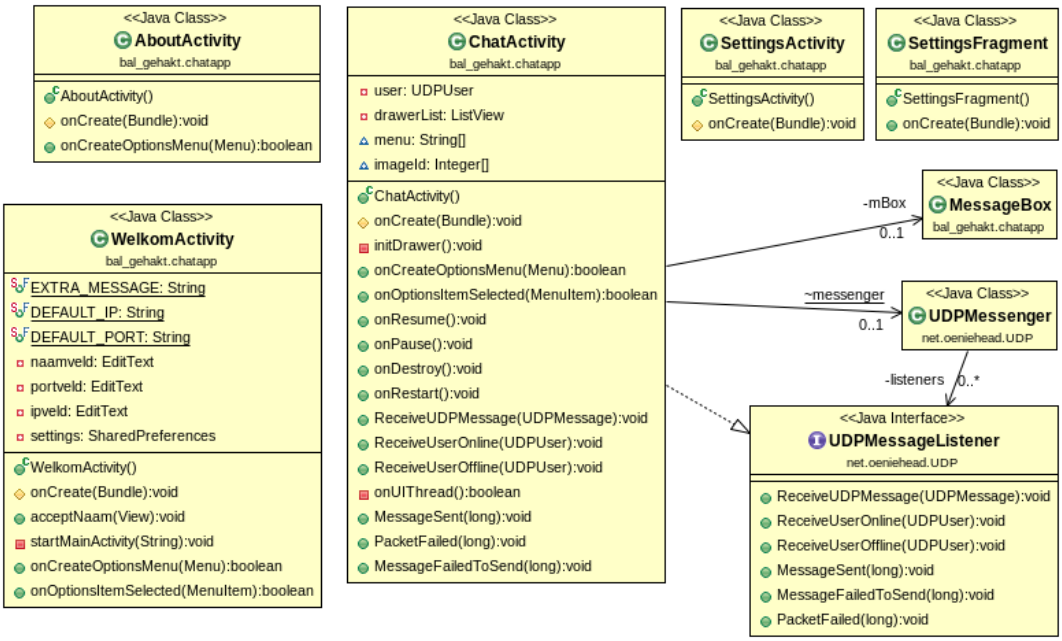
De UDPMessenger ondersteunt het bijhouden van online gebruikers. Een van de timers stuurt eens per gedefinieerd tijdsinterval een pakket waarin wordt aangegeven dat de gebruiker van deze telefoon online is. Als een van deze pakketten wordt ontvangen wordt de gebruiker die in dit pakket staat als online gezien en wordt deze toegevoegd aan de lijst van online gebruikers. Als een gebruiker twee keer de interval geen bericht meer heeft gestuurd wordt deze als offline aanschouwd en wordt de gebruiker van de online-lijst verwijderd.

Als de MessageBox de UDPMessenger de opdracht geeft om een bepaald bericht te versturen komt dit bericht in een lijst terecht en wordt er direct een poging gedaan het bericht te versturen. De tweede timer zorgt ervoor dat er eens per tijdsinterval wordt gecontroleerd welke berichten door andere partijen zijn ontvangen en schrapt deze van de lijst. Andere berichten die nog niet zijn ontvangen door andere partijen worden opnieuw verzonden.

Als een pakket wordt ontvangen dat reeds een keer ontvangen is, wordt dit pakket genegeerd door het systeem. Elk pakket dat binnenkomt en geen ontvangstbevestiging is, wordt 'geacknowledged' door het systeem. Dat wil zeggen dat er een nieuw pakket het netwerk in gestuurd wordt, waarin wordt aangegeven dat het pakket is aangekomen.

## 3.2 Detailontwerp

### 3.2.1 Activities



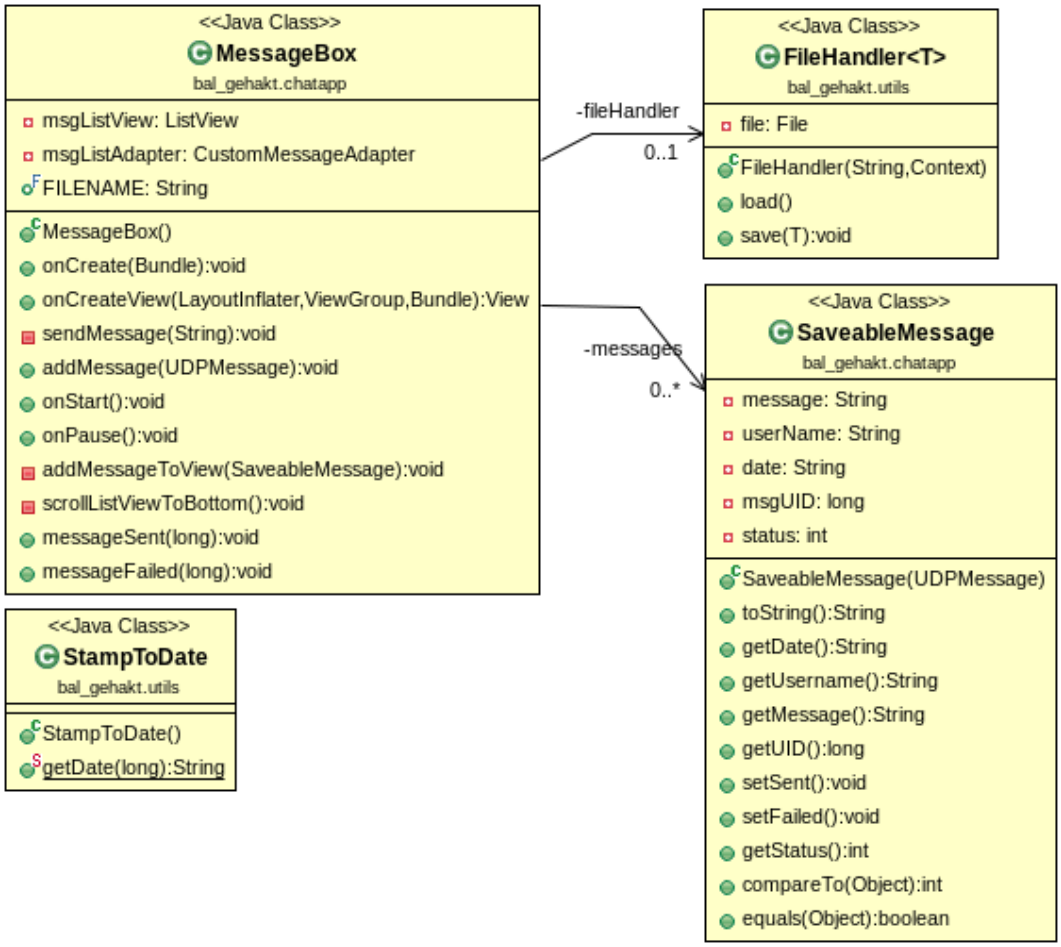

Figuur 3: *Het UML diagram van de activities en verwante klassen.*

De activities zijn opgebouwd uit verschillende onderdelen. Allereerst zijn er de standaard methoden die moeten worden geïmplementeerd om te kunnen werken op Android. De WelkomActivity heeft daarnaast nog een aantal variabelen om de elementen in op te slaan waar later de gegevens uit moeten worden gelezen. Ook heeft deze een variabele om de SharedPreferences in op te slaan en een methode acceptNaam() om een input-validatie te kunnen doen. De SettingsActivity maakt gebruik van een SettingsFragment dat ervoor zorgt dat de settings eenvoudig kunnen worden weergegeven.

De ChatActivity is de meest uitgebreide klasse. Deze klasse implementeert ook de UDPMessageListener, om events te kunnen ontvangen van de UDPMessenger. Er is nog een utility-methode onUIThread() gemaakt om te controleren of een methode wel vanaf de UI-thread is aangeroepen, dit om te voorkomen dat er wijzigingen aan UI-elementen worden gedaan vanuit threads die dit niet mogen. De ChatActivity houdt ook de elementen en plaatjes van de drawer bij. Als laatste is er een methode om de drawer ook daadwerkelijk toe te voegen aan de activity.

De ChatActivity heeft als variabelen een instantie van de MessageBox en een referentie naar de instantie van de UDPMessenger.

### 3.2.2 MessageBox




Figuur 4: *Het UML diagram van de MessageBox en verwante klassen.*

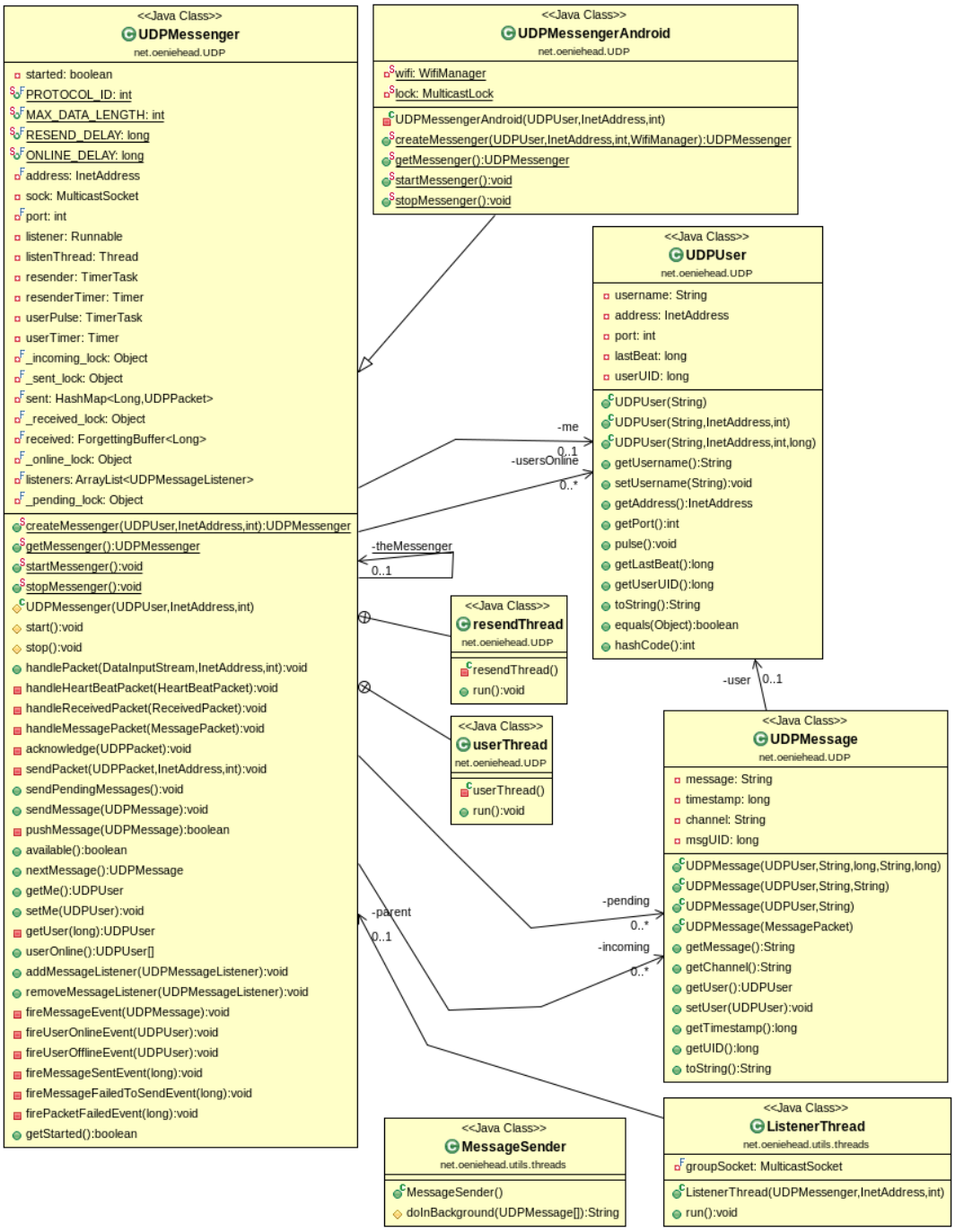
De MessageBox implementeert een aantal methoden die bij een Fragment horen. Er is een bestandsnaam statisch vastgelegd waar de lijst met reeds ontvangen berichten wordt opgeslagen en uitgelezen. De variabelen msgListView en msgListAdapter verzorgen het tonen van de ontvangen in de berichten in de ListView. MessageBox heeft een aantal methoden om berichten te versturen met zijn tekstveld, en ook om een binnen gekomen bericht toe te voegen aan de view.

Er zijn de methoden messageSent en messageFailed om aan te geven of berichten wel of niet zijn aangekomen, deze passen het bericht in de msgListAdapter aan, en updaten de view. Er wordt gebruik gemaakt van de StampToDate klasse die in staat is een timestamp in de vorm van een long om te zetten naar een leesbare tijd als String.

De MessageBox werkt met berichten in de vorm van SaveableMessage. Deze klasse implementeert de Serializable klasse, zodat deze door de FileHandler makkelijk kan worden weggeschreven naar het opslagbestand. De SaveableMessage heeft meerdere methoden om data in zichzelf uit te lezen en te manipuleren.

De SaveableMessage wordt binnen het systeem geïdentificeerd door zijn msgUID. De msgUID is een long die of bij het aanmaken van de klasse gegenereerd wordt, of die wordt gekopieerd uit een UDPMessage.

### 3.2.3 Messenger




Figuur 5: *Het UML diagram van de UDPMessenger en verwante klassen.*

De UDPMessenger is een klasse die alle netwerkcode huist. De ingegeven connectiegegevens worden bijgehouden in variabelen, evenals alle binnengekomen berichten, gebruikers die online zijn en een aantal vaste variabelen voor de timing.

Er worden twee timers en een thread bijgehouden. De resendThread is een Timer die eens per RESEND_DELAY opnieuw alle berichten die nog niet geacknowledged zijn opnieuw verzonden en eventueel uit de lijst verwijderd als het te lang niet is gelukt om het bericht te verzenden. Vanuit deze timer wordt ook het MessageFailedToSend event gestuurd. De userThread timer is een timer die elke ONLINE_DELAY controleert of er gebruikers zijn die al een tijd geen heartbeat gestuurd hebben uit de online lijst verwijderd.

De ListenerThread is een thread met als doel om te luisteren naar binnenkomende berichten op de MulticastSocket. Binnengekomen berichten worden teruggestuurd naar de handlePacket methode van de UDPMessenger. In de handlePacket methode wordt dit pakket doorgegeven aan de goede verwerkingsmethode handleHeartBeatPacket, handleReceivedPacket of handleMessagePacket.

Als een bericht moet worden verstuurd gaat dit via de sendMessage methode van de UDPMessenger. Deze zet het te verzenden bericht in een wachtrij en start een MessageSender. De MessageSender is een AsyncTask die alle berichten in de wachtrij probeert te verzenden door de methode sendPendingMessages aan te roepen op de UDPMessenger.

De UDPMessenger heeft een aantal statische methoden om de messenger te starten en te stoppen. De UDPMessengerAndroid variant voegt hier methoden aan toe om ook direct een MulticastLock aan te maken. Om dit mogelijk te maken houdt deze intern een referentie naar de WifiManager en de eigenlijke MulticastLock bij.

De UDPMessenger accepteert het toevoegen en verwijderen van listeners voor de events met de addMessageListener en removeMessageListener methoden. Op het moment dat er listeners zijn krijgen deze een een event doorgestuurd als een van de fire...Event methoden wordt aangeroepen.

### 3.2.4 Packets


Figuur 6: *Het UML diagram van de packets.*

De packets die in het systeem worden gebruikt zijn opgebouwd uit één hoofdklasse, namelijk UDP-Packet. De pakketten beschikken allemaal over een uniek ID dat in de variabele uid staat. Elk pakket heeft ook een source- en destinationaddress, maar deze zijn in de huidige implementatie nog niet ectief gebruikt.

Als een pakket moet worden opgestuurd wordt er een byte array opgebouwd met de writePacket methode. Deze methode krijgt een DataOutputStream mee waar alle onderdelen van het pakket in worden geschreven. Allereerst schrijft het basispakket zijn gegevens in de stream, en daarna het pakket zelf.

Het aanmaken van een pakket gebeurt ook m.b.v. een byte array. Hiervoor wordt een DataInputStream aan de statische methode createPacket gegeven. Deze methode stelt het packet-id vast en maakt het goede pakket aan. Daarna wordt de readPacket methode aangeroepen en wordt uit de stream alle informatie weer uitgelezen.

Mocht een pakket niet worden gevonden omdat het packet-id niet bestaat, dan kan de createPacket methode een InvalidPacketIDException gebruiken om dit duidelijk te maken.

## 3.3 Ontwerpverantwoording

### 3.3.1 De main view

In de main view van de app hebben we er voor gekozen om alleen te focussen op het chatten. Dit betekent dat je als gebruiker alleen alle berichten ziet en als je zelf een bericht typt natuurlijk je toetsenbord. Door verder geen andere zaken zichtbaar te hebben wordt je als gebruiker niet afgeleid.

Het menu kan vanuit de main view meteen bereikt worden, door een swipe-beweging vanaf de linkerkant van het scherm naar rechts te maken. Op deze manier kun je toch meteen in het menu komen, maar is deze standaard verborgen. In Android heet dit een 'drawer'. De drawer zorgt ervoor dat het menu snel bereikbaar is, zonder dat het in de weg zit. Vanuit deze drawer is het vervolgens mogelijk om te navigeren naar andere schermen, zoals de instellingen. Behalve dat de drawer efficient werkt, ziet het er ook goed uit. Het esthetische aspect is hier dus ook belangrijk voor deze keuze.

We hadden er bijvoorbeeld ook voor kunnen kiezen om de actionbar te gebruiken voor de menu onderdelen, maar we wilden de app een eenvoudig uiterlijk geven. Daarom hebben we ervoor gekozen om niet allerlei knoppen in de actionbar te stoppen. Een andere mogelijkheid zou zijn geweest om de menuknop te gebruiken die op elk Android-toestel zit. Wij hebben hier niet voor gekozen, omdat veel minder ervaren Android-gebruikers vaak niet weten dat deze knop een menu kan openen in veel apps.

### 3.3.2 De opbouw van pakketten

Een van de basisonderdelen van de app zijn de pakketten die over en weer gestuurd worden en daarmee communicatie tussen de meerdere apparaten mogelijk maken. Het idee was om verschillende soorten pakketten, met dus varierende inhoud, over het netwerk te kunnen versturen. Deze collectie aan pakketten moest dus ook makkelijk uit te breiden zijn in het geval dat er meer functionaliteit vereist was.

Waar we voor hebben gekozen is om een standaard klasse te maken, welke UDPPacket heet. Dit is een abstracte klasse die een aantal methoden en eigenschappen bevat waardoor het een begin is van een volwaardig pakket. Dit pakket zelf bevat nog geen informatie die echt zinvol is om te versturen, het bevat enkel gegevens om een semi-betrouwbaar transport mogelijk te maken.

De klassen die informatie toevoegen aan een pakket hoeven zich op deze manier alleen maar zorgen te maken over hun eigen data, gezien de rest van de pakketinformatie al wordt geregeld door de UDPPacket klasse.

Het uitlezen en inlezen van alle pakketgegevens is nu ook erg eenvoudig. Alle pakketten moeten de methoden readPacket en writePacket implementeren. Deze methoden kunnen uit een stream data uitlezen of juist data wegschrijven. Deze stream wordt eerst doorgegeven aan de UDPPacket, waar de basis pakketinformatie kan worden geregeld, en daarna kan de decorator van de UDPPacket zijn eigen informatie toevoegen.

Dit model is gebaseerd op netwerkcode in oude versies van het spel Minecraft. Hierin werd op vergelijkbare wijze gewerkt met pakketten. Ook is deze vorm van aanpak grofweg terug te vinden in het ISO-OSI model. Daar komt bij dat dit een goed voorbeeld was van een situatie waarin een bepaald basis-object gespecialiseerd moest worden.

Bij het ontwikkelen is direct voor deze aanpak gekozen omdat dit verreweg de best manier leek om meerdere pakketten mogelijk te maken. Het is erg lastig dit te implementeren zonder de inheritance die nu ingezet is, en zou een heleboel extra code opleveren die ook nog erg rommelig zou zijn.

# 4 Reflectie

## 4.1 Reflecties per groepslid

### 4.1.1 Bart van de Put

Persoonlijk ben ik best heel tevreden over dit project. Een positieve ervaring hierbij vond ik de samenwerking binnen onze groep. We hebben vaak samen afgesproken in Nijmegen om samen verder te werken aan de app. Dit werkte erg prettig omdat je zo makkelijk kon overleggen of hulp vragen als iets niet lukte. Wel denk ik dat we af en toe een wat duidelijkere taakverdeling hadden moeten maken.

Een echte negatieve ervaring heb ik eigenlijk niet gehad. Een lastig puntje was de weinige ervaring die we met Android hadden. Hierdoor liep je vaak tegen specifieke dingen aan die je dan eerst weer helemaal moest uitzoeken. Dit is niet erg, want zulke dingen uitzoeken kan erg leuk zijn, maar bij dit project waar we niet erg veel tijd hadden was het niet echt handig dat we nog geen voorkennis hadden.

Verder hebben we nog wat problemen ondervonden met het Eduroam netwerk, wat ons ook wat vertraging heeft opgeleverd. Maar uiteindelijk hebben we dit prima kunnen oplossen door zelf een router mee te nemen.

Al met al vond ik het leuk om zo in groepsverband aan de app te werken, heb ik denk ik veel geleerd, vooral met betrekking tot Android. Mijn algehele ervaring van het project is daarom positief.

### 4.1.2 Ronnie Swanink

Werken aan het project was een leuke, maar rommelige, ervaring. Het beginnen met Android ging een beetje stroef, ook al had ik al wel ervaring met programmeren in Java. Toen we eenmaal door de Setting Sun opdracht heen waren gekomen lukte het allemaal wel om met Android te werken. Aan het eind van het ontwikkelen aan onze app werkt alles best goed voor de dingen die we erin hebben verwerkt, maar ik het wel het idee dat het vooral een samentrekking is van allemaal dingetjes die we van internet geplukt hebben, omdat er gewoon weinig tijd was om echt bekend te raken met het systeem.

Een negatief punt was onze 'planning'. Aanvankelijk kwamen we eigenlijk niet samen om aan dingen te werken, dit is gelukkig later beter geworden toen we regelmatig als groep aanwezig waren om samen aan het project te werken. De communicatie ging op zich wel goed, maar hier is zeker verbetering in mogelijk. Uiteindelijk is het ons wel gelukt om alles netjes op tijd af te krijgen.

Dat brengt me bij een positief punt, namelijk dat we het goed voor elkaar hebben gekregen om alles op tijd af te hebben. Aanvankelijk waren er een paar opstartproblemen, maar toen het project eenmaal van de grond was liep alles wel lekker door. Uiteindelijk hebben we dan ook een naar mijn idee mooi project neergezet.

We hebben helaas een redelijke tegenvaller gehad met dat het Eduroam netwerk onze manier van pakketten versturen niet ondersteunde voor draadloze apparaten. Toen we dit eenmaal hadden vastgesteld hebben we hier over overlegd en hebben we een oplossing gevonden voor dit probleem.

Over het algemeen ben ik tevreden met het project, als groepje is het waarschijnlijk verstandig om aan meer communicatie te doen en een daadwerkelijke planning te maken. Ik heb een hoop opgestoken over Android, maar nog steeds vind ik het een wazig platform.

### 4.1.3 Thomas de Bel

Het werken met Android was voor mij een nieuwe ervaring. Hierdoor was het soms een hoop gedoe om ogenschijnlijk makkelijke dingen voor elkaar te krijgen. Ondanks dat dit af en toe een beetje frustrerend kon zijn, was het vaak leuk om deze puzzeltjes op te lossen. Ik denk dat ik uiteindelijk een hoop heb geleerd over het programmeren voor Android en ben tevreden met het eindresultaat. Toch was het fijn geweest als er hier en daar vanuit het vak wat meer theoretische ondersteuning zou

zijn geweest. We zijn als groep vaak bij elkaar gekomen. In de vroege fase van het vak, was dit bij elkaar komen soms wat rommelig. Ik wist vaak niet wat ik moest doen. Dit kwam deels doordat er gedoe was met Eduroam, waardoor we niet verder konden, maar ook deels doordat we geen concrete afspraken hadden gemaakt over wie wat deed. Dit leidde ertoe dat er soms twee groepsleden tegelijk aan hetzelfde bezig waren, of dat er iemand gewoon niks kon doen, omdat er niet echt een overzicht was. Dit is iets dat voor een volgende project beter van tevoren afgesproken zou moeten worden. In een latere fase van het project ging dit allemaal een stuk beter. Toen we eenmaal een basis stond, hadden we een duidelijke richting en wist iedereen wat hem te doen stond. Iedereen heeft uiteindelijk een goede bijdrage kunnen leveren aan het project.

#### 4.1.4 Tom Nikken

Ik vond dit een leerzaam project. In het begin had ik erg veel moeite met dat wij als groep geen echte werkstructuur of werkverdeling hadden. Ik had erg weinig houvast om aan het project te werken. Daarbij kwam dat ik het eerste college over Android erg snel vond gaan. Ik heb de slides terug gekeken, maar zonder uitleg waren die vaak niet voldoende. Ik heb geprobeerd op internet informatie op te zoeken over Android, maar ik vond alleen informatie die niet toereikend genoeg was. Het volgende college over Android heb ik daardoor niet goed begrepen en daardoor had ik een achterstand op het gebied van Android. Toen we aan het project begonnen, zorgde dit voor veel frustratie.

Na een aantal weken kwam ik erachter dat je het beste in de sectie 'Training' en 'API Guides' van de Android developer site kunt zoeken als je met Android wilt leren werken en dat de referentie hiervoor te gedetailleerd is. Deze ervaring kan ik goed gebruiken voor toekomstige projecten. Ook ging de samenwerking een stukje beter. Nadat we op de universiteit hadden gewerkt, spraken we af wie wat ging doen. Vanaf dat moment ging het werken aan het project een stuk beter.

We hebben tijdens het samenwerken veel bij elkaar gezeten. Ik heb dit ervaren als een prettige manier van samenwerken, omdat je makkelijk kan communiceren. Voor een volgend project zou ik een duidelijke werkverdeling maken waar iedereen het mee eens is.

### 4.2 Analyse

We hadden allemaal nauwelijks tot geen ervaring met Android. Ook vanuit het vak kwam weinig theoretische onderbouwing. Er waren slechts twee korte colleges waar een paar basale aspecten van programmeren in Android werden uitgelegd. Verder is het duidelijk dat we in het begin geen eenduidige richting hadden in wat we moesten doen. Er was geen werkverdeling en er werd af en toe maar wat gedaan. Naarmate de weken vorderden is het ons wel gelukt hier meer structuur in te krijgen.

De tegenvaller met het Eduroam netwerk was een redelijk groot probleem voor het project, maar dit is toch snel opgelost door de groep. Dit was te danken aan dat er verderop in het project wel op regelmatige basis werd afgesproken en er dus snel overlegd kon worden. Deze manier van afspreken was echter niet gepland, het kwam er meer op neer dat er gewoon heel vaak werd afgesproken op korte termijn. Dit leverde situaties op waarin groepsleden niet aanwezig waren omdat er afspraken overlapten.

### 4.3 Conclusie

Bij een volgend project met Android is het voor ons belangrijk om eerst wat meer bekend te raken met wat het doet en hoe het werkt, want dit was naar ons idee niet heel duidelijk behandeld. Het is misschien een goed idee om de volgende keer meer tijd te nemen om beter bekend te worden met een nieuwe programmastructuur. We hadden onszelf bijvoorbeeld wat huiswerk kunnen geven om wat research te doen. Vervolgens hadden we elkaar een soort kleine uitleg kunnen geven over wat we op internet gevonden hadden. Hiermee kom je ook allemaal een beetje op hetzelfde niveau te zitten. Dit had vooral in de beginfase een hoop tijd kunnen besparen.

De structuur is ook iets dat we een volgende keer beter kunnen doen. Aanvankelijk hadden we geen planning, wat als gevolg had dat ons project aan het begin een rommel was. Later kwam de groep wel regelmatiger samen. Het maken van een algemene planning aan het begin van het project, en eventueel het vergaderen over of de groep nog op schema ligt is dan wel een idee voor de toekomst.

De communicatie tussen groepsleden liet op momenten nog te wensen over, het zou in de toekomst wel verstandig zijn om hier goed op te letten.