**Documentatie App GreijdanusAlarm**

**Requirements**:

* Een alarm ongeveer 1á 2 minuten voor de les
* Een alarm precies op het einde van de les
* Voor leerlingen en collega’s
* Voor onderbouw havo / vmbo want hebben hetzelfde rooster
* 50 minuten rooster moet kunnen

Overig / optioneel:

* Uiterlijk maakt niet uit
* Alle telefoons gelijktijdig
* Inloggen met leerlingennummer
* Ouderavonden

**Ontwerp**

Ik ben begonnen met naar van Eijsden toe te gaan en te vragen wat er in de app moest komen en hoe het eruit moet zien. Zoals in de requirements te zien is maakte het uiterlijk niet zo veel uit. Handig voor mij want zo creatief ben ik niet. Toen heb ik bedacht hoe de app moest gaan werken. Ik werk met een variabele waar een compleet weekrooster in staat. Deze variabele bestaat uit vijf instances van DagRooster. Hierin zit weer een variabele waar in staat of hij op een bepaalde tijd les heeft of niet. Voor tijden maak ik gebruik van de library Joda-Time. Ik heb geprobeerd het met standaard Java te doen, maar dit bleek al snel vrij lastig en onnodig ingewikkeld te zijn. Op bepaalde punten was het zelfs onmogelijk. Ik gebruik standaard Java om de dag van de week te bepalen. Hier gebruik ik een Enum voor. Als het een dag in het weekend is returnt de class Dagen null. Dan weet de app dat hij niks hoeft te doen. Dan wordt gekeken naar wanneer de leerling of leraar les heeft en op die tijden, twee minuten voor en precies op het einde, registreer ik de callback class notificationReceiver bij de AlarmManager class van Android. Zodra de NotificationReceiver een call krijgt controleert hij eerst of de tijd die hij mee gekregen heeft niet te veel verschilt van de tijd waarop hij een alarm zou moeten geven. Dit bleek belangrijk te zijn en een bug die ik lang had op te lossen. Hieronder meer daarover. Toen ben ik pas bezig gegaan met de UI. Ik heb gekeken naar wat ik nodig had en er een zo simpel maar handig mogelijke UI bij gemaakt. Een HoofdScherm en een InstellingenScherm. Het hoofdscherm doet bijna alles. Het InstellingenScherm verwerkt de instellingen. Dit zijn de uren die de gebruiker ingeeft dat hij les heeft. Deze worden opgeslagen in een aantal variabelen en weggeschreven naar het telefoongeheugen. Dit gebeurd in JSON formaat en hier gebruik ik de library JSON-Simple voor. Als laatste heb ik nog een DownloadTask class. Deze is nodig om een JSON bestand van mijn dropbox (op dit moment) af te halen waarin staat of op deze dag er een vijftig minuten rooster is. Hier houd de DagRooster class dan rekening mee. Halverwege heb ik besloten dat ik het buildproces en vooral het dependency beheer niet handmatig wil doen. Hiervoor gebruik ik Maven. Dit is een tool gemaakt door Apache en werkt prima voor mij. Dat is ongeveer de werking van mijn app tot nu toe.

**Testplan**

Helaas is een Android app niet anders te testen dan door hem te gebruiken. De Java Unit testen werken niet voor Android. Dus, ik heb hem maar op de handmatige manier getest. Het was zeer waarschijnlijk meer werk geweest om geautomatiseerde testen te maken dan om de app zelf te maken. Dit vond ik nogal nutteloos. Ik heb in HoofdScherm een variable ingebouwd waarmee ik kan zeggen dat de app in debug mode moet draaien. Dat betekent dat hij de tijden in DagRooster aanpast en om de paar minuten een melding geeft. Het resultaat van deze testen was in het begin rampzalig. Zoals te verwachten. Het werkte totaal niet. Ik ben toen begonnen met het inbouwen van debug berichten d.m.v. System.out.println(); Dit is niet helemaal de manier waarop Google het wil hebben, maar zo is de output van LogCat beter te filteren voor mij. De uiteindelijke conclusie van de testen is dat de app prima werkt. Er zijn nog wel een aantal problemen, die komen zo aan bod.

**Evaluatie**

Mijn grootste probleem was de ontdekking dat Android event-driven is zoals het heet. Oftewel: als er iets gebeurd in Android dan reageert het OS hierop. Het is niet zo dat je kunt zeggen van: wacht 5 minuten en geef dan een melding. Als je je Android device vergrendeld dan duurt het nog even en Android gaat naar slaapstand. Behalve de Systeemprocessen (waaronder AlarmManager gelukkig) gaan alle andere processen op “pauze”. Een Sleep() functie gaat dus zelf even “slapen”. Dit is heel goed voor de batterij maar niet al te handig voor de programmeurs van apps. Dit heb ik overwonnen door AlarmManager te gebruiken. Verder was het probleem dat ik een aantal meldingen gewoon niet kreeg. Ik had geen idee waarom. Ik heb mijn oom eens gevraagd of hij ernaar wou kijken en dat wou hij zeker. Hij is met zeven jaar ervaring een Java expert. Dit was zinvol, hij heeft mij verteld dat als ik iets registreer bij AlarmManager, ik elke instantie van Intent uniek moet maken. Dit is gedaan door er een setAction() call voor te plaatsen waar een string in zit plus een nummer. Dit nummer is elke keer anders omdat de calls naar AlarmManager worden gedaan vanuit een for-loop. Mijn tweede probleem kwam zo ongeveer op het einde: de UI paste niet op elk scherm. Bleek te komen doordat de standaard Android CheckBoxes een vrij grote witrand hebben. Een eigen CheckBox gemaakt en het probleem was zo ongeveer opgelost. Wel moest ik rekening houden met het feit dat je een mobiel of tablet kunt draaien. Hier moest ik een speciale lay-out voor maken. Het derde probleem was het downloaden van het bestand met de vijftig minuten dagen. Een bestand downloaden, hoe moeilijk kan het zijn? Nou, wel aardig lastig. Na veel error meldingen heb ik ontdekt dat Android het weigert om vanuit de main thread (UI thread) een bestand te downloaden. Dit heb ik opgelost door AsyncTask te gebruiken. Dat was even puzzelen maar het is gelukt, omdat de code niet verder mag voor het bestand ook echt binnen is. Ik heb ook nog geprobeerd om van Maven naar Gradle te switchen. Gradle is de build tool die Android sinds kort standaard gebruikt. Bleek heel erg veel werk te zijn, en ik vond het niet echt nodig. De branch voor Gradle staat nog op GitHub.

Wat mij uiteindelijk wel mee viel was het bedenken van de werking van de app en de manier waarop ik het om zou zetten naar code. Ik heb eerst een ander systeem gebruikt dan wat ik nou doe. Deze valt wel terug te vinden op GitHub, waar overigens al mijn broncode voor de app op staat, inclusief de workspace van Intellij IDEA. Dit is de IDE die ik sinds een poosje gebruik voor Java.

Helaas zijn er ook nog een paar bugs die ik nog niet heb kunnen oplossen, of waar ik nog niet aan toe gekomen ben:

* Op het HoofdScherm hoor in het tekstvak onder de rechter tijd de tijd te komen met wanneer de gebruiker weer les heeft. Naar mijn idee werkt deze wanneer hij er zin aan heeft.
* Ik heb nog wat problemen met de UI. Hij werkt niet perfect op elk schermformaat.
* Je moet de app nog elke dag dat je hem gebruikt even opstarten zodat hij zichzelf in kan stellen. Dit moet automatisch worden, maar dat wordt nog even werk.
* De laatste bekende tot nu toe is dat hij bij het opstarten alles doet. Of je de app nou voor het eerst opstart of terug komt omdat je bijvoorbeeld even een appje hebt gestuurd, hij doet zo goed als alles elke keer. Dit wordt nog even puzzelen om het netjes op te lossen, maar het zal wel lukken.

**Conclusie**

Al met al was het een leuke, leerzame maar ook lastige opdracht. Het heeft me heel wat uren alleen maar nadenken gekost. Dit vind ik niet erg, ik hou wel van een uitdaging, vooral met programmeren. Verder heb ik met van Eijsden afgesproken dat ik, zolang ik het kan, de app blijf onderhouden. Dit betekent dus ook dat de bovengenoemde bugs ooit opgelost zullen worden. Ik stop er niet mee omdat we dit project in moeten leveren.

Als laatste:

De broncode staat op GitHub:

<https://github.com/berryh/GreijdanusAlarm>

De laaste versie is voor nu te downloaden via deze link:

<http://bit.ly/greijdanusalarm>