Matrixrekenen op zijn best

AssertTrue

Marco Hernandez
Roland Leferink
Joren Vrancken
Gijs Hendriksen

17 juni 2016

Voorwoord

In dit verslag zullen wij een uitleg geven over onze app Matrix Calculator. Deze hebben wij gemaakt voor de cursus Research & Development. Het bevat een uitgebreide omschrijving van de app en keuzes die we hebben gemaakt tijdens de development. Eerst geven we een korte beschrijving van de app en het proces, vervolgens gaan we dieper in op de onderdelen van de app zelf en tenslotte geven we een korte reflectie.

Beschrijving

Inleiding

Onze Matrix Calculator is gemaakt om een gebruiker snel en gemakkelijk een matrix in te laten voeren en daar een aantal berekeningen op uit te laten voeren. Dit zijn dus ook de belangrijkste eigenschappen van onze app:

- Het gemakkelijk invoeren en wijzigen van matrices, en deze opslaan om ze later makkelijk te kunnen gebruiken.
- Het uitvoeren van berekeningen op deze matrices.
- Het laten zien van de benodigde stappen van deze berekeningen, zodat het algoritme duidelijk wordt voor de gebruiker.

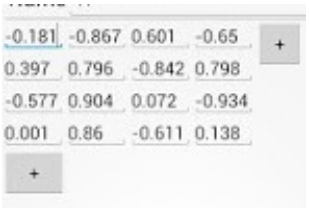
Onze app is zo gebruiksvriendelijk, professioneel en informatief mogelijk ontworpen en gebouwd.

Productverantwoording

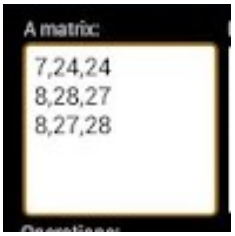
Op Google Play zijn al meerdere Matrix Calculator apps te vinden. Wij hebben een aantal van deze apps bestudeerd en vonden dat er meerdere dingen waren die beter konden. Hieronder hebben wij een paar punten waarvan wij vinden dat wij het beter hebben gedaan, inclusief screenshots van de betreffende apps.

Zo waren er apps die maar een totaal van drie matrices ondersteunden. Dit vonden wij een beperking, omdat je soms gewoon meer matrices nodig hebt, bijvoorbeeld als je meerdere berekeningen snel achter elkaar wil uitvoeren zonder de matrices te verliezen. Daarom hebben wij ervoor gekozen om een vrijwel ongelimiteerde hoeveelheid matrices toe te staan.

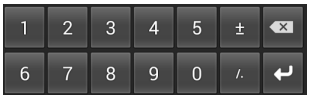
Daarnaast waren bepaalde apps aesthetisch niet aantrekkelijk. Zo waren er apps waar het invoeren van de matrix gebeurde in een redelijk saai, wit schermpje waar de EditTexts heel dicht op elkaar staan, ook al was er meer ruimte.



Bovendien waren er apps waar het invullen van matrices in één tekstveld ingevuld moesten worden, waarbij elke regel op een nieuwe lijn stond en de kolommen gescheiden worden door komma's. Dit vonden wij al helemaal gebruiksonvriendelijk. Wij hebben ons best gedaan om een nette, intuïtieve en makkelijk te gebruiken invoer te maken voor onze app.



Ook hebben bepaalde apps ervoor gekozen om een eigen toetsenbord te maken in hun app, in plaats van het standaard Android-toetsenbord. Dit is redelijk onhandig naar onze mening, en wij wilden liever standaard Android features gebruiken, omdat dat bijna altijd beter is dan het implementeren van een eigen toetsenbord.

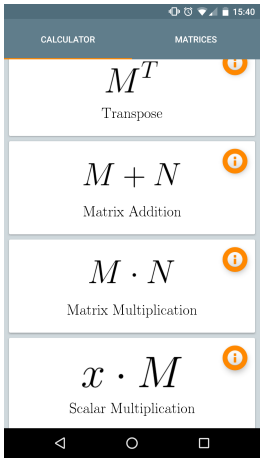


Ten slotte hebben wij de feature toegevoegd om de stappen van de berekeningen te laten zien na het uitvoeren, waarvan wij van mening zijn dat het een heel belangrijke feature is. Voor zover wij weten zijn wij de eerste Matrix Calculator app die dit ondersteunt, en daardoor zijn wij zeer waardevol voor bijvoorbeeld studenten die de basis van matrixberekeningen willen leren.

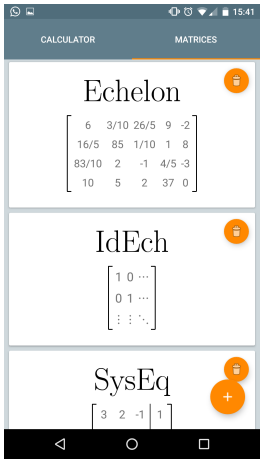
Specificaties

Zoals gezegd zijn de belangrijkste eigenschappen van onze app het invoeren van matrices, en het uitvoeren en zorgvuldig laten zien van de berekeningen. Deze features zijn opgebouwd uit vijf onderdelen:

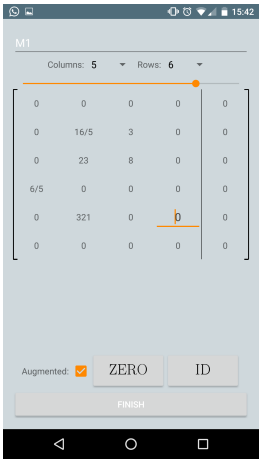
- **Calculations tab**
  We hebben een tabblad met negen verschillende berekeningen, die de gebruiker uit kan laten voeren op bepaalde matrices. Deze berekeningen zijn duidelijk aangegeven met hun wiskundige notatie en de naam van de berekening. Daarnaast hebben ze ook allemaal een eigen informatie, die te zien is door op het icoontje rechtsboven te klikken.



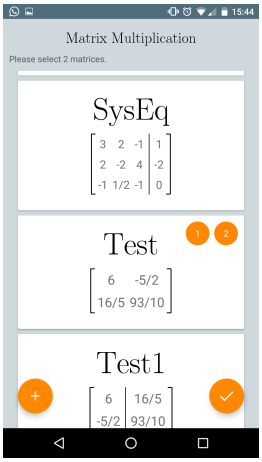
- **Matrices tab**
  We hebben een ander tabblad, waar de opgeslagen matrices worden weergegeven en waar deze weer gewijzigd en nieuwe matrices gecreëerd kunnen worden. Bij elke matrix staat de naam erboven. Daarnaast hebben we twee maatregelen getroffen om de matrices compacter te maken als ze te groot waren. Ten eerste wordt de lettergrootte kleiner naarmate de matrix groter wordt. Ten tweede laten we grotere matrices standaard in een ingeklapte vorm zien. Dat wil zeggen dat we puntjes laten zien aan de rechter- en onderkant, tot de gebruiker op de matrix klikt. Dan wordt de matrix weer uitgeklapt. Matrices kunnen ook verwijderd worden, door op het prullenbakje te drukken. Dit laat ook een Snackbar zien, waarmee de actie ongedaan gemaakt kan worden.



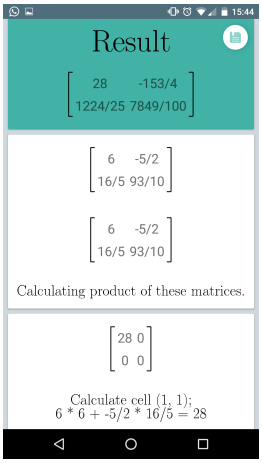
- **Wijzigen/creëren matrix**
  Als een bestaande matrix gewijzigd of een nieuwe gecreëerd wordt, krijgt de gebruiker een scherm te zien met de optie om de matrix naar wens te wijzigen. Bovenaan staat de naam van de matrix. Daarna zijn er opties om het aantal kolommen en het aantal rijen aan te passen, tot een maximum van tien. Vervolgens zie je de matrix zelf. Je kunt op een cel drukken om deze te wijzigen. Bij het groter maken van de matrix behoudt deze zijn huidige data. Wat extra features zijn twee knoppen om de matrix leeg te maken (ZERO) en om de identiteitsmatrix te maken (ID), en de mogelijkheid om de matrix geaugmenteerd te maken. Dit activeert een slider boven de matrix, om de lijn in de matrix te verschuiven. De 'finish'-knop slaat de matrix op in de database, en intern zodat de matrix ook tijdelijk gebruikt kan worden.



- **Kiezen matrix**
  Als de gebruiker een berekening heeft uitgekozen, krijgt hij een lijst te zien met alle matrices. Net als bij de library is hier ook gedacht aan het compact houden van de matrices, met behulp van lettergrootte en de mogelijkheid tot inklappen. De gebruiker moet, afhankelijk van de berekening, een matrix en eventueel een getal of een tweede matrix selecteren. Als hieraan is voldaan, krijgt de gebruiker de optie om de berekening uit te voeren. Daarnaast is er ook de mogelijkheid om vanaf dit scherm nog matrices toe te voegen. In dat geval krijg je de optie om de matrix op te slaan in de database, mocht je dat willen.



- **De berekening**
  Na het kiezen van de matrices wordt de berekening uitgevoerd en getoond. Prominent bovenaan staat het resultaat van de berekening. Als dit resultaat bestaat uit een enkele matrix, is er de mogelijkheid deze op te slaan met het 'save'-icoontje rechtsboven. Onder het resultaat wordt eerst aangegeven welke berekening er precies wordt uitgevoerd. Daarna wordt stap voor stap duidelijk uitgelegd welke berekeningen zijn uitgevoerd om bij het resultaat te komen. Mocht de berekening om bepaalde redenen niet uitgevoerd kunnen worden, dan wordt er een informatieve foutmelding weergegeven.



Ontwerp

Globaal ontwerp

Wij gebruiken ten eerste het Model View Controller model. Dit is erg goed toe te passen op onze app omdat we heel goed onderscheid kunnen maken tussen de berekeningen (het model) en het interface (View en Controller). Wel hebben wij er voor gekozen om de View en Controller niet afzonderlijk te implementeren, omdat dit vooral met Android specifieke klassen en methodes gaat.

Het Model bestaat vooral uit losse onderdelen, die niet veel met elkaar te maken hebben. Denk, bijvoorbeeld, aan een klasse die een matrix implementeert en een klasse voor de communicatie met de database. Deze krijgen pas een daadwerkelijke betekenis als ze gecombineerd worden met het interface.

Wij hebben het interface opgesplitst in vier verschillende secties. De algemene sectie, één voor beide tabbladen en een laatste voor de stappen die gebruikt worden in verschillende computaties. De algemene klassen zijn klassen die gebruikt worden door beide tabbladen. Dit betreft klassen als views om matrices te tonen. De tabblad secties bevatten klassen die specifiek zijn voor het desbetreffende tabblad. Hier vallen de fragments onder. Als laatste hebben we een sectie speciaal voor verschillende views die gebruikt worden tijdens het laten zien van de resultaten van de computaties. Hier zitten zeer specifieke views in voor ieder soort stap die gemaakt wordt tijdens de computaties.

Detailontwerp

- **Computations klasse**
  De computaties zijn de backbone van de app. De computations klasse bevat alle afzonderlijke algoritmes die bij de verschillende computaties horen. De computations klasse doet op zichzelf niks, maar wordt, als het nodig is, aangeroepen door het interface.

- **EditMatrix klassen**
  Een van de meest belangrijke features in onze app, is de mogelijkheid om de gebruiker hun eigen matrix te laten invullen. Er zijn meerdere klassen die hierover gaan. We hebben een redelijk unieke oplossing gevonden voor de invoer van matrices. Voor de input van de matrix zelf, hebben we ervoor gekozen dat ieder nummer wordt getoond in een TextView. Dan wanneer er wordt gedrukt op het nummer verandert deze in een EditText en kan deze aangepast worden. Omdat EditText een subklasse is van TextView, is dit erg handig en efficient qua recycling. De augmented line is geimplementeerd met een horizontale slider waarmee je boven de matrix de index van de augmented line kan selecteren.

- **MatrixView klasse**
  De matrixview is de view die gebruikt wordt iedere keer dat er een matrix getoond wordt. De matrixview wordt zowel in de computaties als in de opgeslagen matrices tab getoond. Naast de nummers in de matrix zelf, laat de matrixView meer zien zoals blokhaken aan beide zijkanten. Daarnaast wordt ook de grote van de cijfers kleiner bij bredere matrices. Het verkleinen van de cijfer grote vonden wij alleen niet genoeg. Matrices die groter zijn dan 2x2 worden standaard ingeklapt, dit houdt in dat alleen de rechter bovenhoek van de matrix te zien is en puntjes aan de zijkanten om aan te geven dat de matrix eigenlijk groter is. Als er dan op de matrix gedrukt wordt, wordt de matrix uitgeklapt. Bij de computaties staat dit standaard uit. Voor efficientie, worden matrixView zoveel mogelijk gerecycled.

- **DatabaseHandler klasse**
  Android heeft meerdere manier om data permanent op te slaan (op disk en niet in RAM). Omdat wij speciale data hebben om op te slaan en ook veel data hebben om op te slaan, hebben wij gekozen voor een achterliggende SQLite database. SQLite is de database standaard van Android, en zit grotendeels geintegreerd in het Android systeem. Voor onze database benodigdheden hebben wij een klasse geschreven om alle communicatie met de achterliggende database af te handelen, de DatabaseHandler klasse. Iedere keer als er data geschreven naar of gelezen uit de database wordt, wordt er een methode aangeroepen in de DatabaseHandler klasse.

- **Step en Stepview klassen**
  De step klassen zijn de link tussen de model en het interface. Terwijl een computatie wordt uitgerekend, wordt er een interne lijst opgevuld met stappen die uitgevoerd worden. Deze stappen bevatten de data en de uitleg met wat er gebeurd tijdens de computaties. Iedere stap heeft een bijhorende view. Deze view bevat de layout voor de specifieke stap. Iedere stap heeft zijn eigen aparte view omdat er in de verschillende computaties verschillende soorten stappen worden uitgevoerd, die verschillende data hebben.

Ontwerpverantwoording

Zoals al eerder besproken, zijn het uitleggen van de berekeningen en het invoeren van matrices de twee belangrijkste features. Hier hebben we dan ook het meest over nagedacht.

- **Computatie stappen systeem**

  We hebben vanaf het begin al het idee om het computatie systeem met stappen te laten werken. We wilde het eerst modulair maken. Dat we een stepView hadden voor iedere stap, waarbij we alleen de nodige informatie aan zouden passen. Helaas bleek tijdens de implementatie dat de verschillende soorten data zoveel van elkaar verschilde, dat het niet reeel was om het verder door te ontwikkelen. Daarna hebben we het modulaire ontwerp weggegooid en hebben gekozen om voor iedere soort stap een aparte view te maken. Dit was uiteindelijk meer werk, maar wel een beter en mooier ontwerp.

- **Matrix invoer systeem**
  Zoals we ook al hebben toegelicht in de productverantwoording, waren er meerdere apps waar de matrixinvoer saai of onhandig was. Wij hebben dus ons best gedaan om deze zo simpel en gemakkelijk mogelijk te maken. Ons eerste idee was om alle cellen een EditText te maken. We kwamen er echter al snel echter dat dit niet erg mooi was, en bovendien werkte het niet optimaal. We hebben toen besloten om een normale matrix weer te geven, en een cel alleen bewerkbaar te maken als er op gedrukt wordt. Dit maakt het duidelijk welke cel er wordt bewerkt op het moment, en het maakt het scherm een stuk rustiger, omdat er een stuk minder lijntjes staan om de EditTexts weer te geven.
  We hebben wel geprobeerd de views zo veel mogelijk te recyclen, om de app zo efficiënt mogelijk te maken. Dit kan helaas niet altijd, maar het is wel een stuk beter dan elke keer nieuwe views initialiseren. Daarnaast hebben we nog meerdere maatregelen getroffen om het scherm zo rustig en leeg mogelijk te maken. Zo hebben we bijvoorbeeld de slider van het geaugmenteerd maken van de matrix weggelaten als deze optie niet geselecteerd is. Daarnaast is het tekstveld voor de naam ook niet zichtbaar als de matrix niet opgeslagen hoeft te worden.

Reflectie

- **Pluspunten**
  Vooral de samenwerking was een erg groot pluspunt. We kenden elkaar al voor het project, en Marco en Joren hadden al veel samen geprogrammeerd, daarom was het samenwerken erg gezellig. We waren na het maken van de dam app allemaal goed op elkaar ingespeeld.
  Geen van ons had hiervoor met Android-Studio gewerkt, daarom was Anroid apps maken volledig nieuw voor ons. Aan het begin ging het nogal stroef met Android-Studio, maar toen we eenmaal door hadden hoe alles in elkaar zat, gingen we erg snel vooruit. Wij waren ook erg blij met de dam app als oefening. Door de dam app kregen we een redelijk simpel project om mee te oefenen, in plaats van meteen in het diepe gegooid te worden.
  Ook de zelfstandigheid vonden wij erg fijn. We hadden een jaar lang programmeer opdrachten gemaakt en waren nu klaar voor een echt project. We mochten alles zelf bedenken en moesten ook onze eigen problemen oplossen, dat vonden wij erg fijn. Het was heel leerzaam voor ons.

- **Minpunten**
  We hebben geen grote minpunten. We hebben hier en daar een kleine bug niet kunnen fixen, maar verder zijn we van mening dat we een goed werkend product hebben ingeleverd. Wel vinden we het jammer dat alle verslagen ingeleverd moesten worden tijdens de tentamen week.

- **Positieve ervaring**
  Werken met Git was een heel positieve ervaring. De meeste van ons hadden al eerder gewerkt met Git, maar nog niet op deze schaal, alleen persoonlijk of voor kleine projecten. Dit was een groter project met echt een team. Het was aan het begin wel wennen en we hebben zo nu en dan ook een paar keer fouten gemaakt met Git, zoals het overschrijven van bestanden met nieuwe versies of merges die verkeerd afliepen. Al met al was het erg leerzaam en zijn wij erg tevreden over hoe Git het versioning heeft gerevolutioneerd.

- **Negatieve ervaring**
  Uiteindelijk konden we niet alle bugs fixen. We hebben een heel stabiel product geleverd, maar hier en daar zitten nog kleine puntjes die volgens ons zelf verbeterd konden worden.
  Er zitten bijvoorbeeld nog een aantal toetsenbord gerelateerde problemen in de matrix invoer, die de gebruiksvriendelijkheid ten nadele komen. Hier hebben we erg veel tijd in gestoken maar het lukte ons uiteindelijk niet om het perfect te krijgen zoals we het wilden.

Concluderend, vonden wij het een erg leuk, leerzaam en uitdagend vak. Maar die uitdaging is juist wat het vak leuk maakt. Voor een volgende keer gaan we beter gebruik maken van Git zodat het pushen en pullen van Gitlab zonder problemen verloopt.