State management in Flutter: een vergelijkende studie.

Angela Degryse.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Sofie Lambert
Co-promotor: Tim Alenus
Instelling: The Mobility Factory
Academiejaar: 2022–2023
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

Lijst van figuren

1

Inleiding

Cross-platform ontwikkelen wordt steeds populairder omdat er vanuit één code-base de applicatie op meerdere toestellen gedraaid kan worden. Er bestaan veel cross-platform frameworks en Flutter is één van de bekendste die in 2018 door Google ontwikkeld werd.

1.1. Probleemstelling

The Mobility Factory (TMF) is een coöperatie die IT-oplossingen biedt aan bedrijven die elektrische autodelen diensten aanbieden. De applicaties worden door de TMF-ontwikkelaars in Flutter geïmplementeerd. Omdat de applicaties van TMF steeds groter en complexer worden, werkt de applicatie steeds minder performant. Daarom kwam de vraag om de performantie van hun applicatie te optimaliseren. Een mogelijke reden van minder performante applicatie is door het slecht beheren van states. In Flutter zijn er tal van mogelijkheden voor het beheren van states. Afhankelijk van de complexiteit van de applicatie moet er overwogen worden welke state management benadering het best bij de applicatie past en of het de meest performante optie is.

1.2. Onderzoeksvraag

De hoofdonderzoekvraag van dit onderzoek is de volgende:
Op welk manier kan states bijgehouden worden in Flutter en welk past het bij bij de applicatie van The Mobility Factory?

Daarnaast kan de hoofdonderzoekvraag verder verdeeld worden in de volgende deelonderzoeksvragen:

- Welk benadering van state management is het performantste qua cpu-gebruik, opstartsnelheid, geheugengebruik...?

- Hoeveel moeite zal het The Mobility Factory kosten om de verschillende benaderingen van state management te implementeren? Hoe complex is het om te integreren in de bestaande code?

1.3. Onderzoeksdoelstelling

Het hoofddoel van dit onderzoek is om de developers van The Mobility Factory te helpen om een beslissing te maken of het veranderen van state management benadering een invloed kan hebben op de performantie van de applicatie. Ook zal er moeten overwogen worden of complexiteit om het te integreren in de bestaande code de moeite waard is. Dit wordt gedaan door middel van een vergelijkende studie. Eerst wordt er een literatuurstudie gedaan over de benaderingen van state management en daarna worden ze met elkaar vergeleken met behulp van een proof-of-concept.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:
In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.
In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.
In Hoofdstuk **??**, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

In dit onderdeel wordt dieper ingegaan op wat flutter is hoe het werkt. Daarna wordt er onderzocht welke state management benaderingen er zijn en hoe ze werken en worden ze met elkaar vergeleken.

2.1. Flutter

Flutter is een open-source framework ontwikkeld door Google coor het bouwen van multi-platform applicaties vanuit één codebase. Dit betekent dus dat ontwikkelaars in een korte tijd applicaties kunnen ontwikkelen voor mobiele, web, desktop en ook voor embedded systemen. Bovendien wordt flutter code vertaald naar native machine code, wat zorgt voor snelle applicaties en annimaties. [1]

Flutter applicaties worden geschreven met Dart, een programmeertaal die ook eigendom is van Google. Dart ondersteunt AOT (ahead of time) compilation en JIT (just in time) compilation. AOT verbetert de opstarttijd en performantie van de application en dankzij JIT compilation wordt hot reload mogelijk. Dit wilt zeggen dat enkel de gewijzigd stukken code opnieuw moet gecompileerd worden in plaats van de volledige applicatie. Het is dus mogelijk om wijziginen in bijna real time te testen. [2]

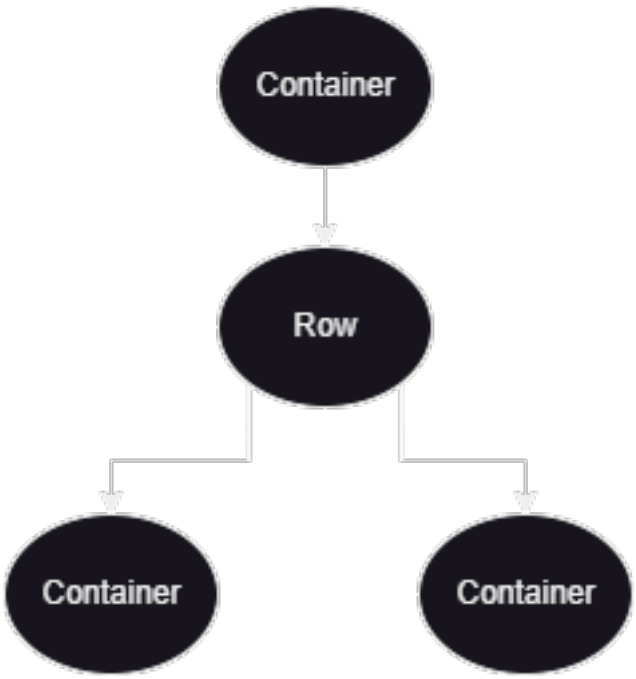
2.2. Flutter Architectuur

Voor dat er over de state management binnen flutter besproken kan worden, wordt er eerst een paar begrippen uitgelegd binnen Flutter. Ook wordt er in dit deel uitgelegd hoe Flutter werkt intern.

[1] https://flutter.dev
[2] https://docs.dart.dev

2.2.1. Widgets

In Flutter wordt alle componenten in de app als "Widget" beschouwd. Wat een widget is is eigenlijk een Dart klasse die beschrijft hoe een stuk van de user interface eruit ziet. De UI van de app wordt dus met andere worden gebouwd door widgets in ouder-kind relatie te combineren. Elk widget wordt in een parent widget genest en context kan doorgegeven worden van ouder naar kind. Alle widgets zitten dus in een widget boom, met elk ouder widget één of meerdere kind widgets.



Figuur (2.1)
Voorbeeld van een widget boom

Elk widget is onderanderlijk. Applicaties werken hun ui bij als reactie op gebeurtenissen (zoals een gebruikersinteractie) door het framework te vertellen dat een widet in de hiërarchie te vervangen door een andere widget. De ouder widgets worden met de nieuwe vergeleken door het framework en wordt de gebruikersinterface efficiënt bijgewerkt.

2.2.2. States

Elk widget kan states hebben. Flutter beschrijft een state als informatie die enerzijds synchroon kan worden gelezen wanneer de widget wordt gebouwd en anderzijds kan veranderen tijdens de levensduur van de widget. [3]

[3] https://api.flutter.dev/flutter/widgets/State-class.html

2.2.3. Widget states

Nu dat er een beeld is over wat states zijn binnen Flutter, kan er over de 2 meest gebruikte soorten widgets besproken worden.

- stateless widget
- stateful widget

2.3. State management

Zoals eerder vermeld wordt Flutter ontwikkeld om performant te zijn. Maar een mogelijke reden die de performantie van de applicatie kan beïnvloeden is het slecht bijhouden van states binnen de applicatie.
State management is een techniek, of meerdere, die gebruikt wordt om veranderingen in de applicatie te beheren. Het gebruik van de juiste state management kan zorgen voor betere leesbaarheid van code en ook voor een performanter applicatie.

2.3.1. setState

todo

2.3.2. Provider

todo

2.3.3. RiverPod

todo

2.3.4. GetX

todo

2.3.5. Redux

todo

2.3.6. Bloc

todo

2.3.7. MobX

todo

3

Methodologie

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at

lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

4

Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem. Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

A

Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1. Introductie

Smartphones zijn een integraal onderdeel van ons dagelijks leven geworden. Daarom worden er steeds meer vraag om mobiele applicaties te ontwikkelen. Ook wordt cross-platform ontwikkelen steeds aantrekkelijker voor ontwikkelaars omdat ze in een kortere tijd applicaties kunnen ontwikkelen voor meerdere platformen.
The Mobility Factory (TMF) is een coöperatie die IT-oplossingen biedt aan bedrijven die elektrische autodelen diensten aanbieden. De applicaties worden door de TMF-ontwikkelaars in Flutter geïmplementeerd. Omdat de applicaties van TMF steeds groter en complexer worden, werkt de applicatie steeds minder performant. Daarom kwam de vraag om de performantie van hun applicatie te optimaliseren.
In Flutter zijn er tal van mogelijkheden voor het beheren van states. Afhankelijk van de complexiteit van de applicatie moet er overwogen worden welke state management benadering het best bij de applicatie past en of het de meest performante optie is. In dit onderzoek wordt eerst de verschillende soorten state management benaderingen grondig besproken en daarna worden ze vergeleken met elkaar op basis van prestatie en code complexiteit aan de hand van een proof-of-concept. Hieruit wordt de onderzoeksvraag dus gecreëerd: *Op welk manier kan states bijgehouden worden in Flutter en welk past het bij bij de applicatie van The Mobility Factory?*. De hoofdonderzoeksvraag kan verder verdeeld worden in de volgende deelonderzoeksvragen:

- Welk benadering van state management is het performantste qua cpu-gebruik, opstartsnelheid, geheugengebruik...?

- Hoeveel moeite zal het The Mobility Factory kosten om de verschillende benaderingen van state management te implementeren? Hoe complex is het om te integreren in de bestaande code?

A.2. State-of-the-art

Flutter is een open source raamwerk van Google voor het bouwen van multi-platform applicaties vanuit één codebase, of met andere woorden een cross-platform raamwerk. Flutter werd in 2018 uitgebracht en wordt ondertussen door meer dan twee miljoen ontwikkelaars gebruikt om meer dan 350 duizend Flutter applicaties te ontwikkelen. [1]

A.2.1. Application performance, een onderdeel van User Experience

Applicatieperformantie is de werkelijke performantie die wordt ervaren door de eindgebruikers van de applicatie, zoals de gemiddelde responstijd bij normale belasting of piekbelasting (Rouse, 2014).
Door een onderzoek van Google wordt er vastgesteld dat naarmate de laadsnelheid oploopt, hoe meer geneigd de gebruikers zijn om de website te verlaten (An, 2010). Bij een laadsnelheid van meer dan 3 seconden van een webpagina, vertrekt 53% van de gebruikers bij het inladen van de webpagina.
Volgens Nielsen (2010), een UX-expert, is responsiviteit van een applicatie een basisregel voor het ontwerp van gebruikersinterfaces. De gebruikers moeten snel door de applicatie kunnen navigeren. Volgens Nielsen is een laadsnelheid tussen 1 en 10 seconden aanvaardbaar. Een laadsnelheid van langer dan 10 seconden kan ervoor zorgen dat de gebruiker afgeleid wordt waardoor het moeilijker is om geavanceerde taken tot een goed einde te brengen tegen dat de applicatie reageert.

A.2.2. Technieken om applicatie performantie te verbeteren

Flutter werd ontwikkeld om performant te zijn. Maar de prestatie van een Flutter applicatie kan sterk beïnvloed worden door hoe de applicatie ontworpen en geïmplementeerd wordt. [2]
Dit zijn 3 voorbeelden om de prestatie van een Flutter applicatie te verbeteren:

1. **Memory management**
   Het memory management in Flutter wordt afgehandeld door de Dart Virtual Machine (DVM), die een garbage collector heeft die automatisch ongebruikte geheugen terughaalt. Er kunnen echter nog steeds geheugenlekken optreden als objecten niet op de juiste manier uit het geheugen worden vrijgegeven, waardoor ze in de heap blijven hangen en onnodige bronnen verbruiken.

[1] https://flutter.dev/
[2] https://docs.flutter.dev/perf/best-practices

Om geheugenlekken te vermijden moet de Dispose-methode gebruikt worden om onnodige objecten te verwijderen. (Sabbagh, 2023)

2. **Netwerk optimalisatie**
   Om netwerk prestatie te verhogen moet er gebruik gemaakt worden van efficiënte netwerk libraries zoals Dio. Ook door de gegevens te cachen op het apparaat van de gebruiker kan er onnodige netwerkverzoeken vermeden worden en wordt de latentie verminderd. [3]

3. **State management**
   Binnen dit onderzoek wordt er gefocust op dit onderdeel. Dit wordt verder beschreven in het volgende puntje 2.4

A.2.3. State management binnen Flutter

Flutter beschrijft een state als informatie die enerzijds synchroon kan worden gelezen wanneer de widget wordt gebouwd en anderzijds kan veranderen tijdens de levensduur van de widget. [4]
States worden niet van parent widget naar child widget doorgegeven. De state van een widget wordt beheerd door zichzelf of door een parent widget en als de state van de parent widget wijzigt, moet alle widgets onder die parent widget ook herladen worden. Daarom kan het apart beheren van states zorgen voor efficiëntere updates en betere prestaties van de applicatie, aangezien enkel betreffende widgets heropgebouwd moeten worden. Deze functie is vooral gunstig voor complexe applicaties, waar het herbouwen van de volledige widgetstructuur tijdrovend kan zijn en kan leiden tot ondermaatse prestaties. Het is dus van groot belang dat de juiste benadering van state management wordt toegepast, zodat het optimaal presteert en een naadloze gebruikerservaring garandeert. (siddhardha, 2023)
Dit zijn de mogelijke benaderingen van state management in Flutter:

- SetState
- Provider
- GetX
- RiverPod
- Redux
- BLoC / Rx
- MobX

[3] https://www.linkedin.com/pulse/flutter-performance-optimization-techniques-best-practices/
[4] https://api.flutter.dev/flutter/widgets/State-class.html

A.2.4. Relevante studies

State management wordt door het Flutter-team als complex bestempeld en er wordt veel gediscussieerd onder de Flutter-ontwikkelaars over wat de beste aanpak is.
Een vergelijkbaar onderzoek dat reeds werd gedaan over state management is door De Vriest (2019). Hoewel zijn onderzoek maar 3 jaar oud is, is Flutter heel sterk geëvolueerd sindsdien. Zijn onderzoek concludeerde dat Provider het meest performant is voor een simpel CRUD-applicatie. Maar in zijn onderzoek waren er tekortkomingen, zoals het enkel testen op een Android-toestel en ook werden data niet van een API opgehaald, wat niet realistisch is voor de realiteit.
Een andere vergelijkbaar onderzoek werd door Slepnev (2020) geschreven. In zijn onderzoek werd de verschillende benaderingen van state management gegroepeerd en vergeleken. In zijn onderzoek werden er criteria opgesteld om het best passende benadering te kiezen voor de verschillende use cases.

A.3. Methodologie

Eerst zal een vergelijkende studie uitgevoerd worden door de verschillende benaderingen van state management op te sommen en hun voor- en nadelen te vergelijken. Uit de gevonden state management benaderingen zullen er vier uitgekozen worden om toe te passen in het proof-of-concept.
In het methodologiegedeelte zullen verschillende versies van een bepaalde feature van de applicatie van TMF geïmplementeerd worden, waarbij voor elke versie een andere benadering van state management wordt toegepast. De applicatie van TMF die nagemaakt zal worden is een beheerapplicatie om reservaties en gebruikers te beheren.
De prestaties van de applicatie zullen getest worden op basis van CPU-gebruik, laadsnelheid en nodige opslag voor de applicatie. Er wordt ook een afweging gemaakt tussen de prestaties die worden geleverd door state management en de complexiteit van de integratie ervan in de bestaande code.
Het proof-of-concept zal geïmplementeerd worden met Flutter op Visual Studio Code. De applicatie van TMF wordt voornamelijk gebruikt op de web, daarom zal de performantie van de proof-of-concept getest worden als een webapplicatie en ook op een fysieke iOS-toestel. De code complexiteit van de state managements zullen vergeleken worden op basis van aantal nodige lijnen code. Om de performantie te meten zal er automated integration testen geschreven en uitgevoerd worden aan de hand van Flutter Driver. Bij elk uitvoering van de testen zullen de performantie gemeten worden. [5]

[5] https://docs.flutter.dev/cookbook/testing/integration/profiling

A.4. Verwachte resultaten

Uit eerdere onderzoeken blijkt dat Provider het best benadering van state management is op vlak van complexiteit en ook op prestatie vlak. Dit is ook de state management benadering die aanbevolen wordt door Flutter. Maar omdat de applicatie van TMF complexer is, wordt er verwacht dat Bloc een betere benadering van state management zal zijn.

A.5. Discussie, conclusie

Er zijn talloze manieren om de prestaties van een Flutter-applicatie te verhogen. Een van de manieren is om het meest geschikte state management toe te passen, afhankelijk van de complexiteit van de geïmplementeerde applicatie.
Het doel van deze studie is om TMF-ontwikkelaars een idee te geven van welke state management opties beschikbaar zijn in Flutter en wat het meest performant is voor hun applicatie.

Bibliografie

An, D. (2010). Find out how you stack up to new industry benchmarks for mobile page speed. Verkregen juli 21, 2023, van https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/

De Vriest, J. (2019). *Een vergelijkende studie tussen verschillende benaderingen van State Management in Flutter* [Bachelor's Thesis]. Hogeschool Gent. Verkregen augustus 5, 2023, van https://scriptie.hogent.be/2019-2020/339_201640029_PBA-TIN_scriptie.pdf

Nielsen, J. (2010). Website Response Times. Verkregen juli 21, 2023, van https://www.nngroup.com/articles/website-response-times/

Rouse, M. (2014). Application performance. Verkregen juli 21, 2023, van https://www.techopedia.com/definition/30457/application-performance

Sabbagh, A. (2023). Flutter Performance Optimization Techniques Best Practices. Verkregen augustus 16, 2023, van https://blog.devgenius.io/identifying-and-handling-memory-leaks-in-flutter-apps-1ad5e7d499e7

siddhardha. (2023). My Experience with Flutter State Management. Everything You Need to Know About it. Verkregen augustus 16, 2023, van https://levelup.gitconnected.com/everything-you-need-to-know-about-flutter-state-management-from-my-experience-19f56729f3c4#:~:text=State%2C%20in%20Flutter%2C%20refers%20to,itself%20or%20its%20parent%20widget.

Slepnev, D. (2020). *State Management approaches in Flutter* [Bachelor's Thesis]. South-Eastern Finland University of Applied Sciences. Verkregen augustus 15, 2023, van https://www.theseus.fi/bitstream/handle/10024/355086/Dmitrii_Slepnev.pdf