Tour de icapps

Realisatierapport

Stage ITFactory

Luca Van Genechten 3APPAI01

Academisch Jaar 2022-2023

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

LID VAN ASSOCIATIE KU LEUVEN

THOMAS MORE

INHOUDSTAFEL

1 INTRO

In dit document vindt u de uitleg van alle prestaties die ik heb gerealiseerd tijdens mijn stage bij icapps. Deze stage vond plaats van 28 februari tot 27 mei 2023.

Om te beginnen zal ik een kleine uitleg geven over het bedrijf waar ik stage heb gelopen. Daarna zal ik het stageproject zelf introduceren, evenals een overzicht van de architectuur en de technologieën die gebruikt zijn om het project te realiseren. Ik zal ook inzicht geven in de stage en hoe ik die heb ervaren.

Maar voordat we verder gaan wil ik icapps nogmaals bedanken, niet alleen voor hun geweldige begeleiding tijdens deze stage, maar ook voor de geweldige kans die ze me hebben gegeven.

2 STAGEBEDRIJF

icapps is een vooraanstaand digitaal agentschap gevestigd in Antwerpen, België, en het heeft een opmerkelijke positie binnen de Cronos Group, een toonaangevend technologie- en innovatiecentrum. Als onderdeel van de Cronos Group profiteert icapps van een sterk netwerk van expertise en middelen, waardoor ze hun klanten geavanceerde oplossingen kunnen bieden.

Met een team van meer dan 100 hoogopgeleide professionals heeft icapps met succes een groot aantal projecten afgerond, variërend van kleinschalige applicaties tot grote bedrijfsoplossingen. Hun portfolio omvat een breed scala aan klanten uit verschillende sectoren, waaronder de financiële sector, de gezondheidszorg, de detailhandel en nog veel meer. Dankzij deze uitgebreide ervaring begrijpt icapps de unieke vereisten en uitdagingen die eigen zijn aan elke sector, wat zorgt voor op maat gemaakte en impactvolle digitale oplossingen. Tot de klanten van icapps behoren bekende bedrijven zoals Monizze, KBC, Telenet, Orange, Haven van Antwerpen, Luminus en vele anderen.

icapps heeft kantoren in zowel Antwerpen als Mechelen, maar mijn stage zal plaatsvinden in het kantoor in Antwerpen. Ik zal aan de slag gaan binnen Squad 42, een toegewijd team van developers die gespecialiseerd zijn in één bepaald framework: Flutter.

3 HET PROJECT

3.1 Reden

Binnen icapps is sporten een echte levensstijl. Veel medewerkers doen dagelijks aan wielrennen, vriendschappelijke voetbalcompetities, darts of spinning. Deze passie voor sport bracht Hannes Van den Berghe, de huidige teamleider van Squad 42 en een fervent wielrenner, ertoe om een vriendschappelijke wielerwedstrijd op te zetten binnen icapps. Deze jaarlijkse competitie, bekend als "Tour de icapps", valt samen met de Tour de France.

De Tour de icapps bestaat uit meerdere etappes die tijdens een toernooi plaatsvinden. Medewerkers kunnen worden gevraagd of ze wel of niet mee willen doen aan het toernooi, waarna alle deelnemers worden verdeeld over teams. Elke etappe bestaat uit meerdere uitdagingen, bijvoorbeeld "Meeste gefietste kilometers per team" of "Hoogste gemiddelde topsnelheid per team". Deelnemers kunnen vervolgens tijdens elke etappe zoveel activiteiten doen als ze willen.

Hannes trad altijd op als jurylid van deze wedstrijd en moest daarom alle activiteiten van de deelnemers nauwgezet bijhouden met behulp van een Excel-sheet en de Strava-applicatie. Dit proces was echter tijdrovend en belastend voor Hannes.

3.2 Doel van het project

Het doel van dit project is om een Minimum Viable Product (MVP) te ontwikkelen waarmee Hannes alle activiteiten van de spelers kan bijhouden en scoreborden voor het hele toernooi kan automatiseren.

In eerste instantie worden alle werknemers gevraagd om deel te nemen en ze kunnen de uitnodiging accepteren of weigeren. Daarna worden er willekeurig teams samengesteld met de deelnemende werknemers en krijgen ze namen toegewezen. Deze teams worden vervolgens opgeslagen en gebruikt binnen de applicatie.

Deelnemers kunnen tijdens het toernooi activiteiten voltooien die worden vastgelegd op Strava. Strava is een internetservice en -applicatie die wordt gebruikt voor het bijhouden van lichaamsbeweging en die verschillende sociale netwerkfuncties biedt. Deelnemers kunnen al hun activiteiten bekijken en uploaden naar de applicatie.

Zodra een activiteit is geüpload, wordt deze gekoppeld aan een etappe. Een etappe wordt gedefinieerd door twee data en heeft specifieke uitdagingen. Deze uitdagingen kunnen variëren in doel, zoals bepalen welk team de meeste kilometers heeft gereden tijdens een etappe, welk team de meeste hoogtemeters heeft gemaakt, enzovoort.

Om de nauwkeurigheid van geüploade activiteiten te garanderen, zijn er meestal eisen verbonden aan de uitdagingen en etappes. Bijvoorbeeld, in een etappe die gericht is op een tijdrit, moeten deelnemers een afstand van meer dan 25 kilometer hebben afgelegd. In een langere etappe die meerdere dagen beslaat, kan een vereiste zijn dat de totale afstand die een team heeft afgelegd groter moet zijn dan 250 kilometer. Activiteiten die niet voldoen aan de eisen op het niveau van de uitdaging of etappe komen niet in aanmerking voor het berekenen van punten.

Als alle uitdagingen zijn voltooid en het toernooi is afgelopen, moet een klassement met de scores van alle teams zichtbaar zijn op zowel etappe- als toernooiniveau.

4 ARCHITECTURE

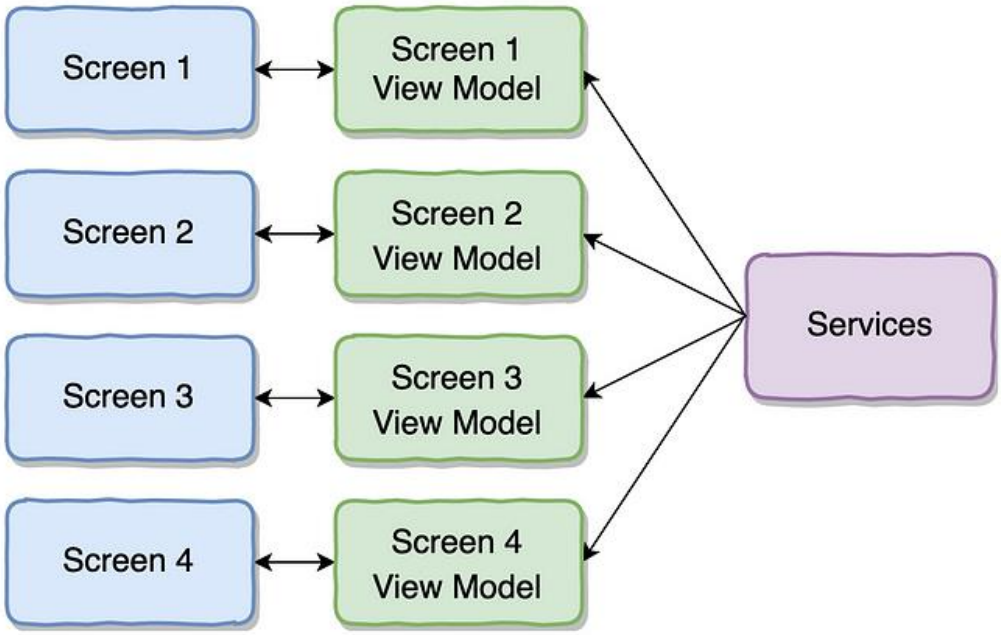
Flutter is een krachtig cross-platform framework ontwikkeld door Google waarmee developers mooie en krachtige applicaties kunnen bouwen voor verschillende platformen, waaronder webbrowsers. Flutter for Web maakt gebruik van een unieke architectuur die het beste van het reactieve UI-framework van Flutter combineert met de mogelijkheden van het webplatform.

De kern van Flutter for Web is de Flutter engine, die verantwoordelijk is voor het renderen en updaten van de gebruikersinterface. De engine maakt gebruik van de Skia grafische bibliotheek om UI-elementen te tekenen en te manipuleren. In tegenstelling tot traditionele webontwikkelingsbenaderingen die vertrouwen op HTML, CSS en JavaScript, gebruikt Flutter for Web Dart, een statisch getypeerde programmeertaal, om de UI-componenten en logica te beschrijven. Dit stelt developers in staat om een enkele codebase te schrijven voor zowel mobiele als webapplicaties, waardoor de ontwikkelingstijd en -inspanning afnemen.

Om Flutter applicaties te laten draaien in webbrowsers, introduceert Flutter for Web een uniek concept genaamd "Hummingbird". Hummingbird is een framework dat Flutter's UI rendering en layout semantiek vertaalt naar geoptimaliseerde JavaScript code. Het overbrugt de kloof tussen Flutter's rendering engine en het webplatform, waardoor Flutter apps kunnen worden uitgevoerd in een browseromgeving. Deze architectuur zorgt ervoor dat Flutter for Web applicaties dezelfde hoge prestaties en visueel rijke gebruikerservaring behouden waar Flutter bekend om staat, terwijl ze zich naadloos aanpassen aan de mogelijkheden van het webplatform.

Over het algemeen biedt de architectuur van Flutter for Web een gestroomlijnde en efficiënte manier om cross-platform applicaties te ontwikkelen die op het web draaien. Door de kracht van Flutter's reactieve UI framework te combineren met de flexibiliteit van het webplatform, kunnen developers met gemak visueel verbluffende en responsieve webapplicaties maken. Of het nu gaat om het bouwen van een eenvoudige website of een complexe webapp, Flutter for Web biedt een robuuste architectuur die ontwikkeling vereenvoudigt en consistente gebruikerservaringen op verschillende platforms mogelijk maakt.

Tijdens de ontwikkeling van Tour de icapps heb ik ook gebruik gemaakt van de MVVM-architectuur. De Model-View-ViewModel (MVVM) architectuur is een ontwerppatroon dat veel gebruikt wordt in Flutter applicaties. In MVVM vertegenwoordigt het Model de gegevens en de bedrijfslogica, de View vertegenwoordigt de gebruikersinterface en het ViewModel fungeert als bemiddelaar tussen het Model en de View. Het ViewModel stelt gegevens en opdrachten bloot waar de View zich aan bindt, wat zorgt voor scheiding van zorgen en verbeterde testbaarheid. In Flutter kan het Model gerepresenteerd worden door datamodellen of state management oplossingen zoals Provider of Riverpod. De View komt overeen met de UI-widgets en de lay-out, terwijl het ViewModel kan worden geïmplementeerd met behulp van klassen die de toestand beheren en methoden en eigenschappen leveren die de View kan gebruiken. Door de MVVM architectuur te volgen, kunnen Flutter developers schaalbare, onderhoudbare en testbare applicaties maken met een duidelijke scheiding tussen de UI en de business logica.



Figuur 1: voorbeeld van de MVVM architectuur

4.1 Technologieën

Tour de icapps wordt beschouwd als een full stack project, met meerdere technologieën die verantwoordelijk zijn voor het draaiende houden van de front-end en back-end. In de volgende paragrafen beschrijf ik de technologieën en functies die voor elk deel van het project zijn gebruikt.

4.1.1 Back-end

De back-end van het project bestaat uit een paar onderdelen. Laten we het eerst hebben over Flutter. icapps gebruikt een aangepast sjabloon voor al hun projecten, dat een verscheidenheid aan functies bevat. Enkele van deze functies zijn automatisch gegenereerde serviceklassen, DAO-klassen voor gegevenstoegang en een pakket genaamd Drift dat kan worden gebruikt om een lokale SQLite-database te maken. Tijdens het project waren de service- en DAO-klassen enkele van de meest gebruikte klassen.

De serviceklassen maakten gebruik van 2 waanzinnig handige pakketten om API-aanroepen te doen, namelijk Dio en Retrofit. De Dio en Retrofit pakketten zijn populaire netwerkbibliotheken voor Flutter die handige en efficiënte manieren bieden om HTTP verzoeken te doen en API communicatie af te handelen. Dio is een krachtig pakket dat een eenvoudige maar functierijke API client biedt voor het afhandelen van netwerkaanvragen. Het ondersteunt verschillende verzoekmethodes, zoals GET, POST, PUT en DELETE, en biedt functies zoals interceptors, verzoekannulering en formuliergegevensverwerking. Dio ondersteunt ook geavanceerde functies zoals het uploaden en downloaden van bestanden, time-out configuratie van verzoeken en loggen van verzoeken en reacties.

Aan de andere kant is Retrofit een pakket dat geïnspireerd is op de populaire Java bibliotheek met dezelfde naam. Het vereenvoudigt het proces van het definiëren van RESTful API endpoints in Flutter door gebruik te maken van annotaties en automatische codegeneratie. Met Retrofit kunnen developers de

API-interface definiëren met aangepaste request-methodes en -parameters, en het genereert automatisch de benodigde code voor het maken van de requests. Retrofit ondersteunt query parameters, request headers en request/response serialisatie met verschillende data formaten zoals JSON en XML. Het ondersteunt ook verschillende HTTP clients, waardoor flexibiliteit in de keuze van de onderliggende netwerk library mogelijk is.

Zowel Dio als Retrofit bieden efficiënte en flexibele oplossingen voor netwerken in Flutter applicaties. Ze helpen bij het stroomlijnen van API-communicatie, verwerken verschillende soorten verzoeken en bieden geavanceerde functies voor het beheren van netwerkinteracties. Of je nu de voorkeur geeft aan een meer handmatige aanpak met Dio of een declaratieve en op code-generatie gebaseerde aanpak met Retrofit, deze pakketten vereenvoudigen het proces van het werken met API's en verbeteren de ontwikkelervaring in Flutter.

Meestal zullen Flutter applicaties deze backend pakketten gebruiken om te communiceren met API's die hen kunnen voorzien van data van een service of een database. In Tour de icapps werkt dit echter in sommige opzichten een beetje anders. Tour de icapps gebruikt de Dio en retrofit pakketten om te communiceren met de Strava applicatie. Dit is voor het eerst te zien in het inlogscherm, waar een communicatie tot stand wordt gebracht tussen Tour de icapps en Strava met behulp van single sign-on, of SSO. Deze communicatie wordt vervolgens gebruikt voor andere doeleinden, die later in dit document worden uitgelegd.

Wat de backend van Tour de icapps echter niet doet, is gebruik maken van een API om te communiceren met de database van zijn keuze, Firestore. In plaats daarvan wordt er een directe communicatie gemaakt tussen Tour de icapps en Firestore door gebruik te maken van het cloud_firestore pakket, waardoor gegevens onmiddellijk kunnen worden opgehaald zonder dat er een tussenliggende API nodig is.

Om Firestore in de backend te integreren, moeten developers de nodige authenticatie- en autorisatiemechanismen instellen met behulp van Firebase Authentication. Dit zorgt ervoor dat de server over de vereiste referenties en permissies beschikt om toegang te krijgen tot de Firestore database. Zodra de authenticatie is geconfigureerd, kunnen developers met het cloud_firestore pakket een breed scala aan bewerkingen uitvoeren, zoals het aanmaken en bijwerken van collecties, het toevoegen en wijzigen van documenten en het uitvoeren van queries.

4.1.2 Front-end

De front-end van een Flutter-applicatie wordt gebouwd met behulp van een combinatie van widgets, lay-outs en stijlen. Flutter volgt een declaratief UI programmeermodel, wat betekent dat de gebruikersinterface wordt beschreven met code in plaats van met een visuele editor. De kern van Flutter's front-end ontwikkeling is het concept van widgets. Widgets zijn de bouwstenen van de gebruikersinterface en vertegenwoordigen alles van knoppen en tekstvelden tot complexe lay-outs en animaties.

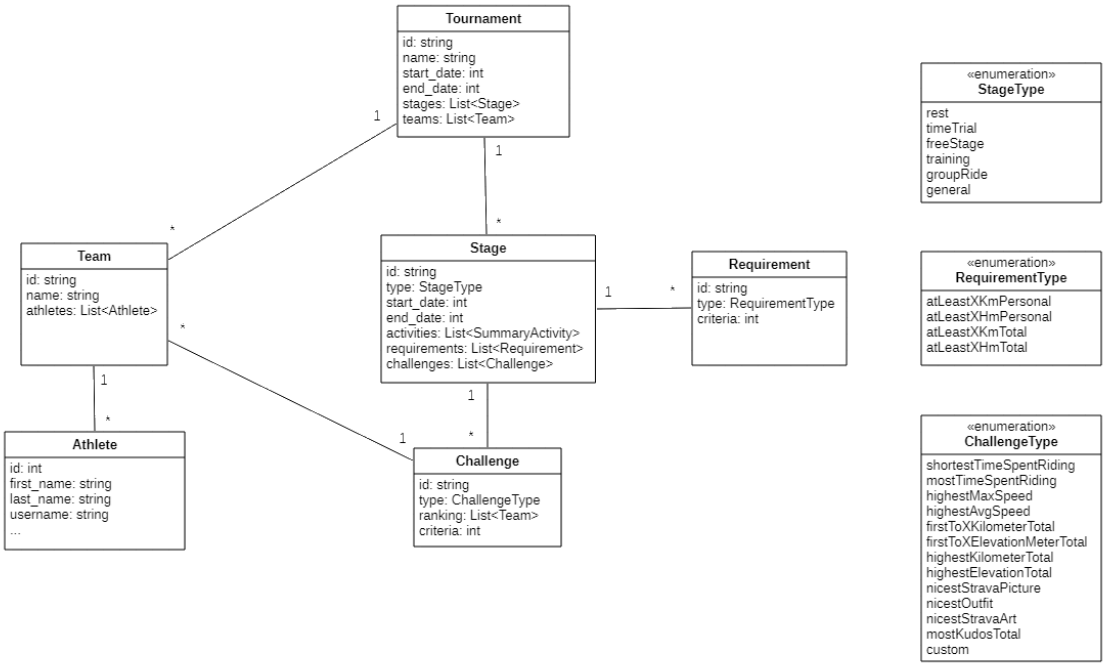
Developers kunnen een breed scala aan vooraf gebouwde widgets gebruiken die door Flutter worden geleverd, zoals Container, Column, Row, Text, Image en meer. Deze widgets kunnen worden samengesteld en genest om complexe UI-structuren te maken. Flutter biedt ook de flexibiliteit om aangepaste widgets te maken door bestaande widgets te subclassen of geheel nieuwe widgets te maken. Widgets in Flutter zijn reactief en stateful, wat betekent dat ze kunnen

worden bijgewerkt en reageren op veranderingen in de toestand van de applicatie.

Om de lay-out van de gebruikersinterface te ontwerpen, biedt Flutter een flexibel systeem dat de widgetboom wordt genoemd. De widgetboom geeft de hiërarchische structuur van de widgets weer, waarbij elke widget een ouder heeft en mogelijk meerdere kinderen. Flutter biedt verschillende lay-out widgets zoals SizedBox, Expanded en Flexible die precieze controle geven over de rangschikking en grootte van widgets binnen een lay-out.

Styling en thematisering in Flutter worden bereikt door het gebruik van eigenschappen en de ThemeData klasse. Eigenschappen zoals kleur, lettergrootte, opvulling en marge kunnen worden aangepast om het gewenste visuele uiterlijk te bereiken. Flutter ondersteunt ook hot-reloading, waardoor developers onmiddellijk veranderingen in de UI kunnen zien als ze wijzigingen aanbrengen in de code.

Over het algemeen wordt de front-end van een Flutter-applicatie gebouwd door widgets samen te stellen en te nesten om de gewenste gebruikersinterface te creëren, lay-outs te definiëren om de widgets te structureren en stijlen en thematisering toe te passen om een visueel aantrekkelijke en consistente look te krijgen. De declaratieve aard van Flutter's UI programmeermodel, samen met de rijke set van vooraf gebouwde widgets en flexibele lay-out systeem, maakt front-end ontwikkeling in Flutter efficiënt en krachtig.



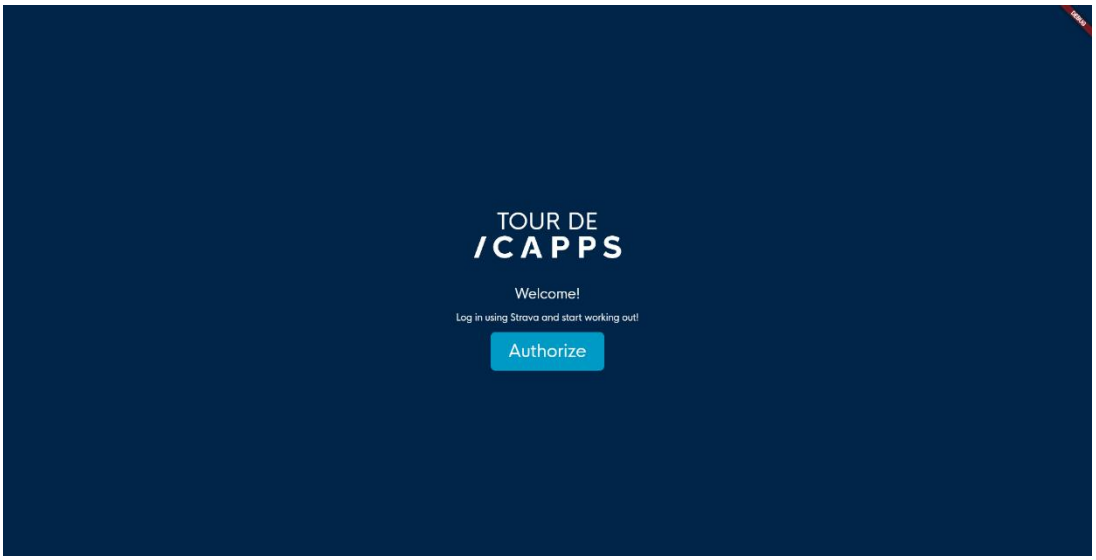
Figuur 2: klassendiagram van de applicatie

5 OVERZICHT VAN ALLE PAGINA'S

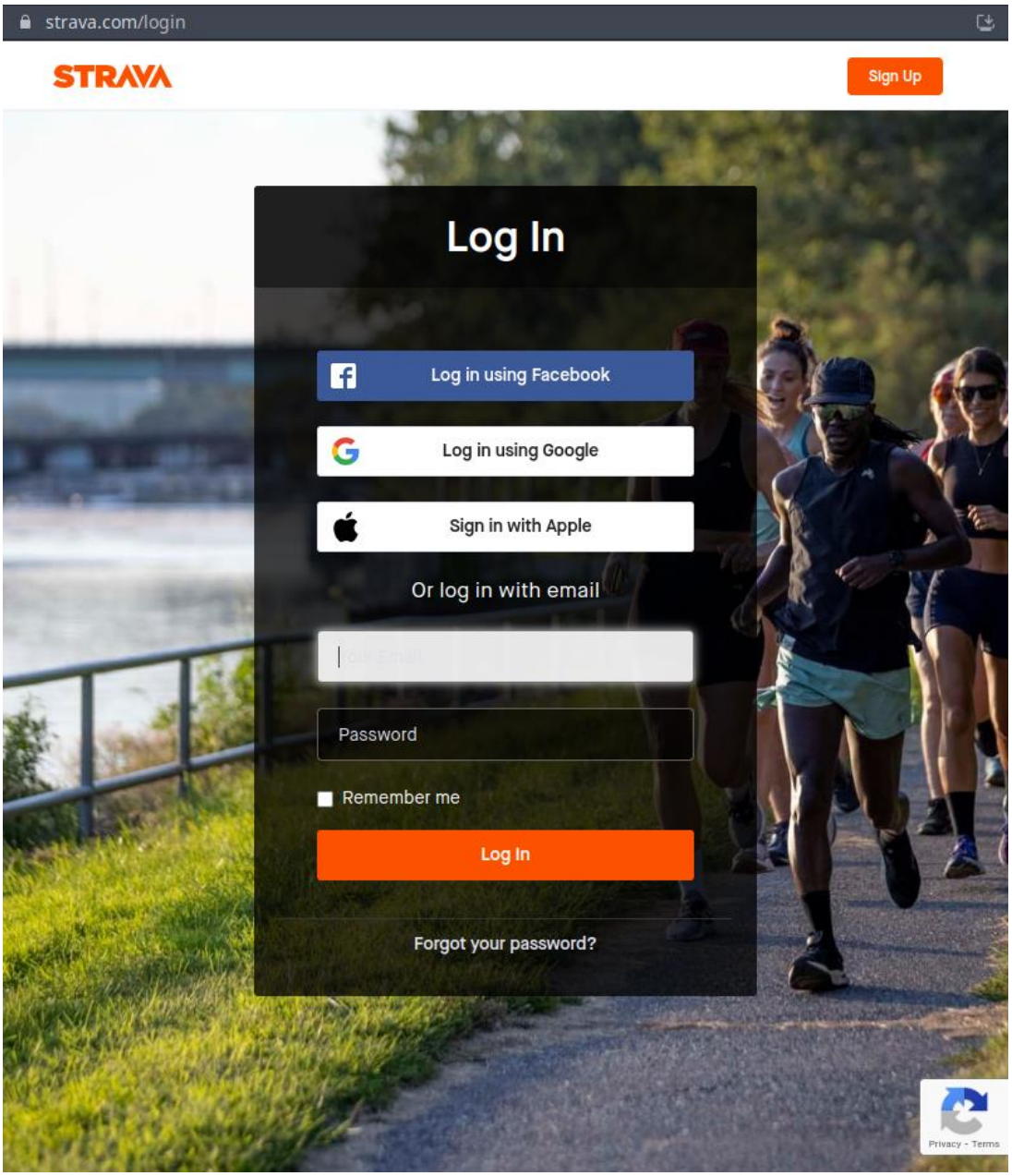
In het volgende deel van het document zal ik elke pagina afzonderlijk bespreken en het doel, de verschillende functies en de werking ervan uitleggen. Om te beginnen bekijken we de eerste pagina die een gebruiker te zien krijgt zodra hij naar de website van de applicatie navigeert.

5.1 Login scherm

Het inlogscherm is de allereerste pagina die een gebruiker te zien krijgt. Om de applicatie volledig te kunnen gebruiken, moet een gebruiker kunnen inloggen bij Strava om activiteiten op te halen en te uploaden. Dat is waar deze pagina om de hoek komt kijken. De enige interactie die een gebruiker zal hebben met Strava zal op deze pagina plaatsvinden. Zodra ze op de knop "Authorize" klikken, verschijnt er een scherm waarin een gebruiker zich bij Strava kan aanmelden met de methode van zijn voorkeur, of dit nu Google, Facebook of via e-mail is. Strava en de applicatie gebruiken vervolgens een proces dat single sign-on (SSO) heet om de gebruiker veilig aan te melden. Wanneer het verificatieproces is voltooid, wordt de gebruiker doorverwezen naar de pagina voor het selecteren van een team, of naar de startpagina als de gebruiker zich al eerder bij een team had aangesloten.



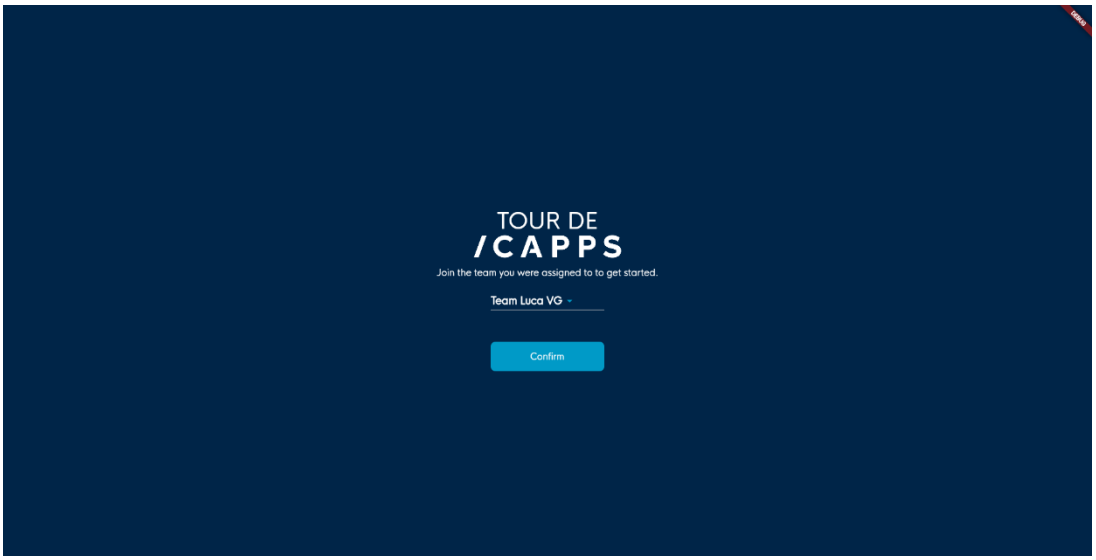
Figuur 3: login scherm



Figuur 4: pop-up om de gebruiker te authenticaten

5.2 Team selectie scherm

Zodra de gebruiker zich heeft aangemeld met SSO, wordt hij doorgestuurd naar de volgende pagina. Het enige doel van deze pagina is dat de gebruiker na het inloggen een team kan selecteren om lid van te worden. Deze pagina wordt maar één keer getoond, wanneer een gebruiker inlogt en zijn Strava ID nog niet is gekoppeld aan een team. Hier kan een gebruiker kiezen bij welk team hij zich wil aansluiten, met een keuze uit alle geregistreerde teams in de applicatie. Zodra een team is geselecteerd, kan de gebruiker zijn keuze bevestigen en verder gaan naar het beginscherm van de applicatie.



Figuur 5: team selectie scherm

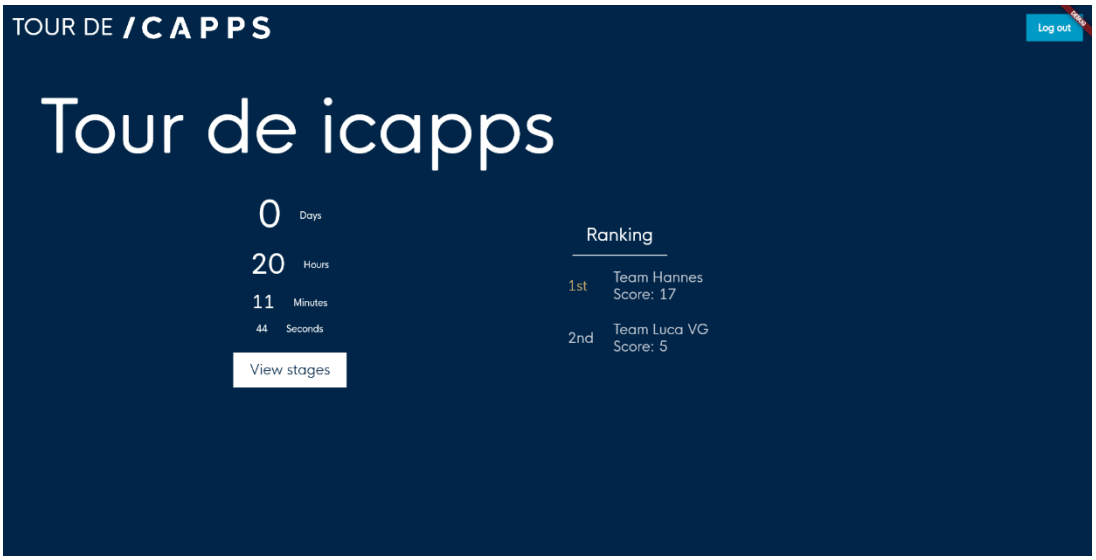
5.3 Home scherm

Het startscherm is het scherm dat alle gebruikers te zien krijgen zodra ze zijn ingelogd en een team hebben geselecteerd om lid van te worden. Het beginscherm gebruikt een paar widgets om verschillende informatie weer te geven. Als een gebruiker inlogt voordat een toernooi is begonnen, wordt hij begroet met een timer aan de linkerkant van het scherm. Deze timer geeft aan hoeveel dagen, uren, minuten en seconden er nog resteren tot het toernooi begint. Zodra deze timer 0 bereikt, wordt hij automatisch vervangen door een tekstbericht dat aangeeft dat het toernooi is begonnen.

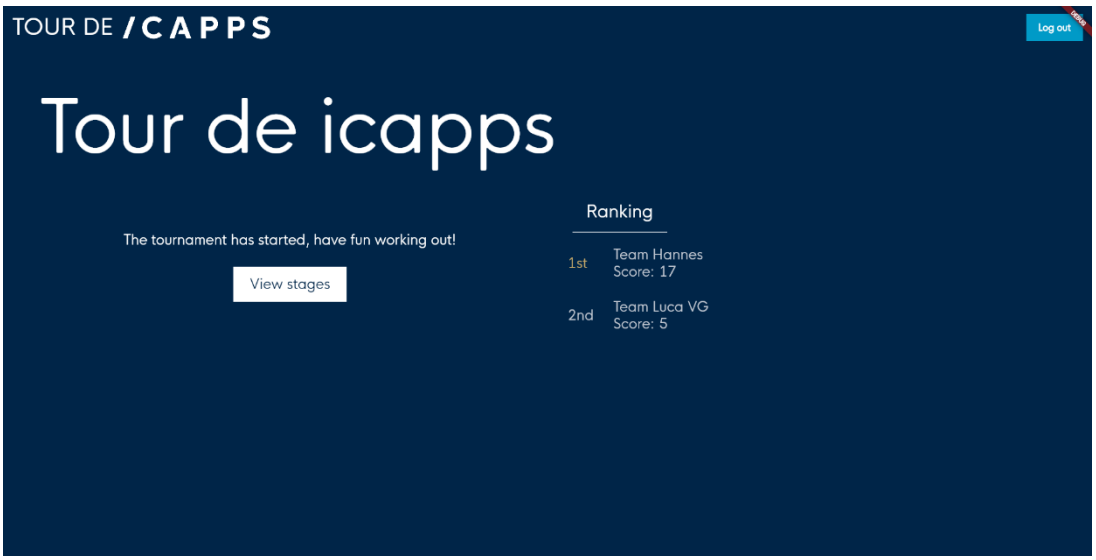
Onder de timer zie je een knop met de tekst "View stages". Als de gebruiker op deze knop klikt, wordt hij doorgestuurd naar het volgende scherm van de applicatie, het toernooischerm.

Aan de rechterkant van het scherm wordt een ranglijst getoond met de scores die elk team in totaal heeft behaald in de loop van het toernooi. Deze scores worden berekend door een ranking te berekenen voor elke uitdaging in het toernooi. Voor elke uitdaging kan een score worden ontvangen van 3 tot 1, waarbij de eerste plaats 3 punten krijgt, de tweede plaats 2 punten en de derde plaats 1 punt. Alleen de top 3 teams per uitdaging kunnen punten ontvangen. De totale scores per team worden dan berekend en hier weergegeven in een ranglijst.

Een ander ding dat we op dit scherm kunnen zien is de navigatiebalk bovenaan het scherm. Op de huidige pagina hebben we 2 knoppen: het logo van de app linksboven en de knop "Log out" rechtsboven. Het logo leidt de gebruiker terug naar deze pagina en is toegankelijk op elke pagina van de applicatie. De knop "Log out" is alleen toegankelijk op de home pagina van de applicatie.



Figuur 6: home scherm met aftellende timer



Figuur 7: home scherm zonder aftellende timer

5.4 Tournament scherm

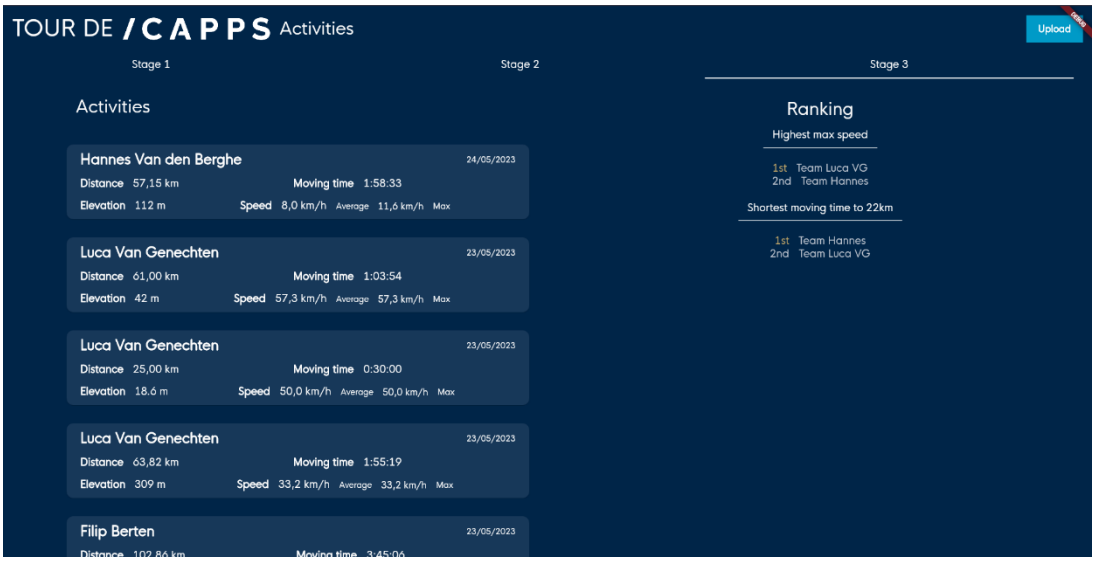
In het toernooischerm wordt alle informatie getoond over de geüploade activiteiten, de uitdagingen per stage en de klassementen voor die uitdagingen. Wanneer we dit scherm voor het eerst openen, worden we begroet met een tabbladweergave van ons toernooi. Elke fase van het toernooi wordt weergegeven met een eigen tabblad, waar alle geüploade activiteiten per fase worden weergegeven in een scrolbare lijst. Alle stadia worden hier weergegeven, behalve de stadia die zijn gemarkeerd als "rust"-stadia, waar geen activiteiten hoeven te worden gedaan. Door op een tabblad bovenaan het scherm te klikken, kan de gebruiker tussen stadia navigeren.

Aan de linkerkant van het scherm zien we elke individuele activiteit die per gebruiker is geüpload. Elke activiteit toont ook relevante informatie, zoals de afstand van de activiteit, de tijd die in beweging is doorgebracht, het aantal hoogtemeters dat is afgelegd en de gemiddelde en maximale snelheid waarmee

de activiteiten zijn uitgevoerd. Dit is allemaal informatie die rechtstreeks van Strava wordt gehaald.

Aan de rechterkant van het scherm zien we de klassementen voor elke uitdaging in een etappe. In het onderstaande voorbeeld kunnen we zien dat deze etappe in het bijzonder slechts 2 uitdagingen heeft: "Highest max speed" en "Shortest moving time to 22km".

Op de navigatiebalk bovenaan het scherm zien we dat het logo van de applicatie blijft om naar het beginscherm te navigeren, maar er is ook een titel voor de huidige pagina toegevoegd. Aan de rechterkant van de balk zien we dat de knop "Log out" van de vorige pagina nu verdwenen is en in plaats daarvan een knop "Upload" toont. Als we op deze knop klikken, kunnen we naar de volgende pagina navigeren.



Figuur 8: tournament scherm

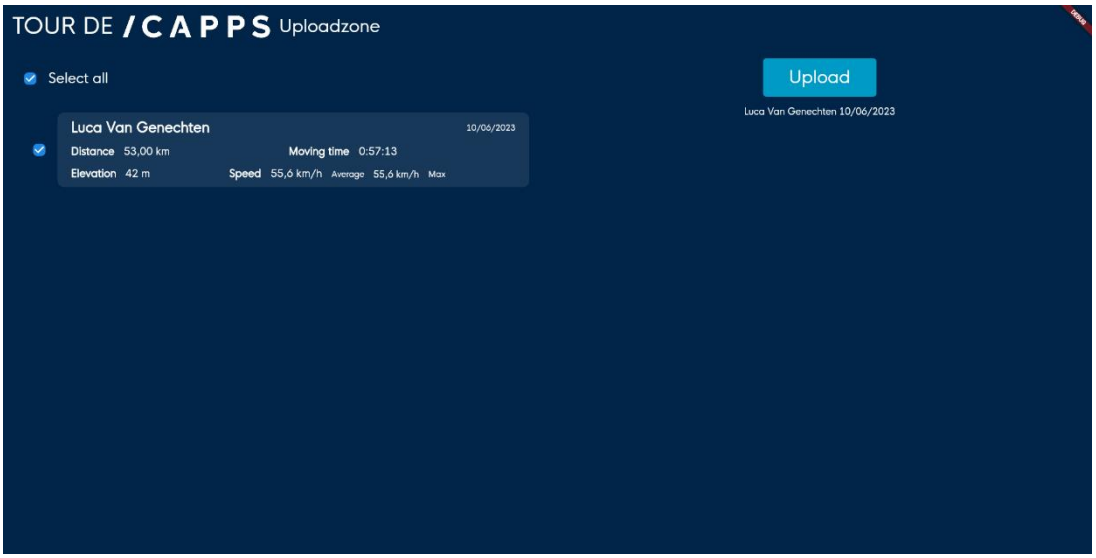
5.5 Upload scherm

Als we op de volgende pagina komen, zien we dat we nu de uploadzone van de applicatie hebben bereikt. Op deze pagina heeft een gebruiker verschillende interacties. Aan de linkerkant van het scherm zien we een lijst met alle fietsactiviteiten die de gebruiker heeft gedaan in de periode van het huidige lopende toernooi. De applicatie toont alleen "Ride", "Gravel Ride" of "Mountain Bike Ride" activiteiten, omdat dat de enige activiteiten zijn die geldig zijn voor dit toernooi. Al deze activiteiten zijn opgehaald uit Strava met behulp van verschillende API-aanroepen.

Door op elke activiteit te klikken, zien we dat het selectievakje wordt ingeschakeld en de activiteit aan de rechterkant wordt toegevoegd. Met de handige knop "Select all" kunnen we alle activiteiten in de lijst selecteren of deselecteren.

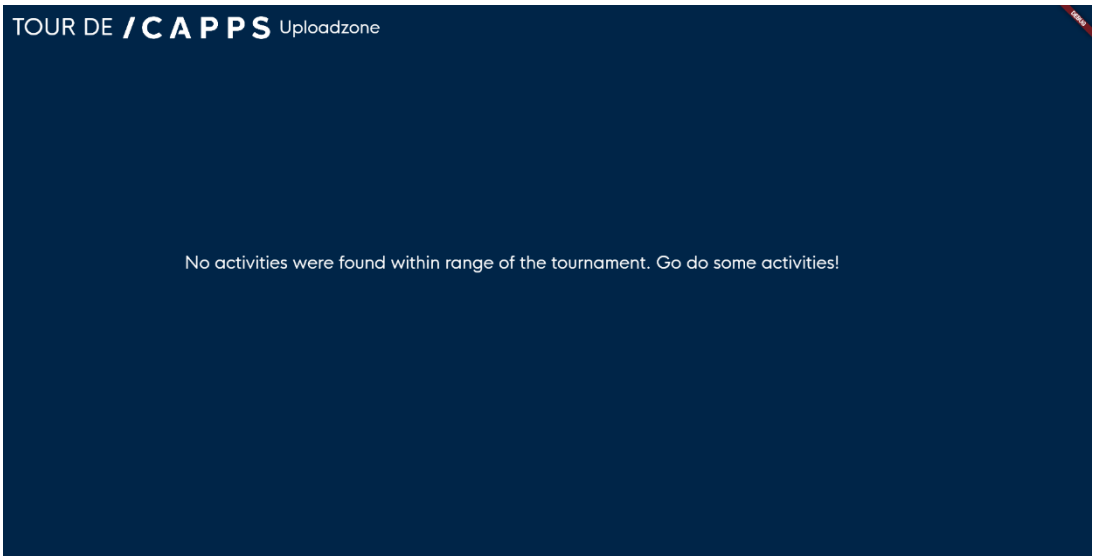
Zodra de gebruiker heeft gekozen welke activiteiten hij wil uploaden, kan hij klikken op de knop "Upload" aan de rechterkant van het scherm. Dit start het proces van het toevoegen van elke gekozen activiteit aan de bijbehorende fase. Als er een fout optreedt tijdens het proces, wordt dit ook weergegeven in een Toastr pop-up onderaan de pagina.

Bij het uploaden van een activiteit controleert de applicatie de datum waarop de activiteit is uitgevoerd. Als de activiteit tussen de begin- en einddatum van een fase in het toernooi valt, wordt de activiteit aan die fase toegevoegd. Als een gebruiker echter te laat is met het uploaden van zijn activiteit, krijgt hij 3 dagen speling. Als een activiteit die tijdens een etappe plaatsvond binnen 3 dagen na het einde van die etappe wordt geüpload, zal deze nog steeds correct worden geüpload. Anders krijgt de gebruiker een foutmelding.



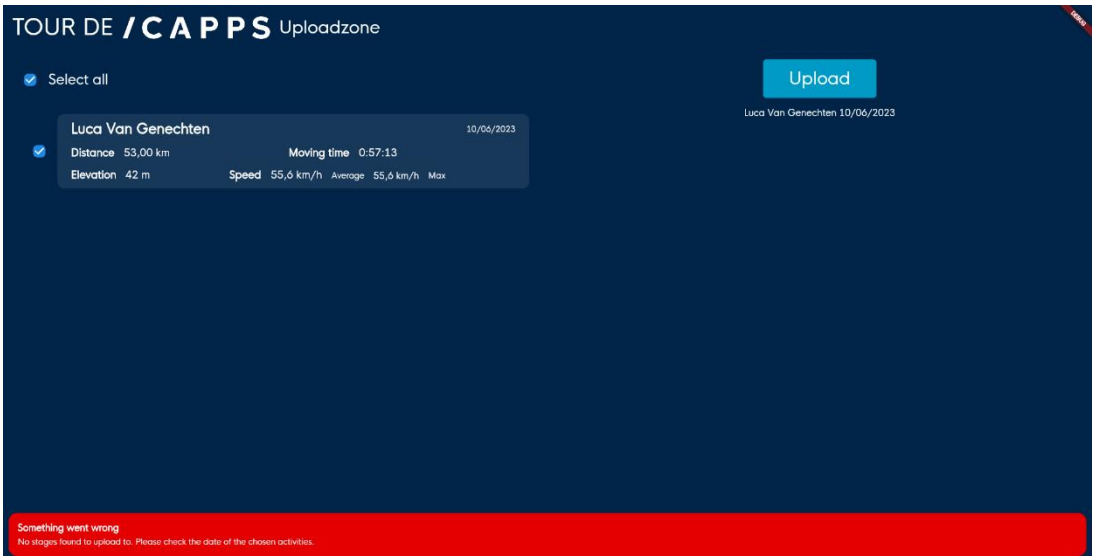
Figuur 9: upload scherm met activiteiten

Als dit scherm wordt bereikt zonder dat er enige registratie is van activiteiten tijdens de periode van het toernooi, wordt de volgende melding getoond. De gebruiker kan in dit geval geen activiteiten uploaden.



Figuur 10: upload scherm zonder activiteiten

Als de gebruiker een activiteit probeert te uploaden die niet plaatsvond tussen de begin- en einddatum van een fase, krijgt de gebruiker de volgende foutmelding en wordt de activiteit in kwestie niet geüpload.



Figuur 11: upload scherm zonder foutmelding

6 INTEGRATIES

6.1 Single Sign-On (SSO)

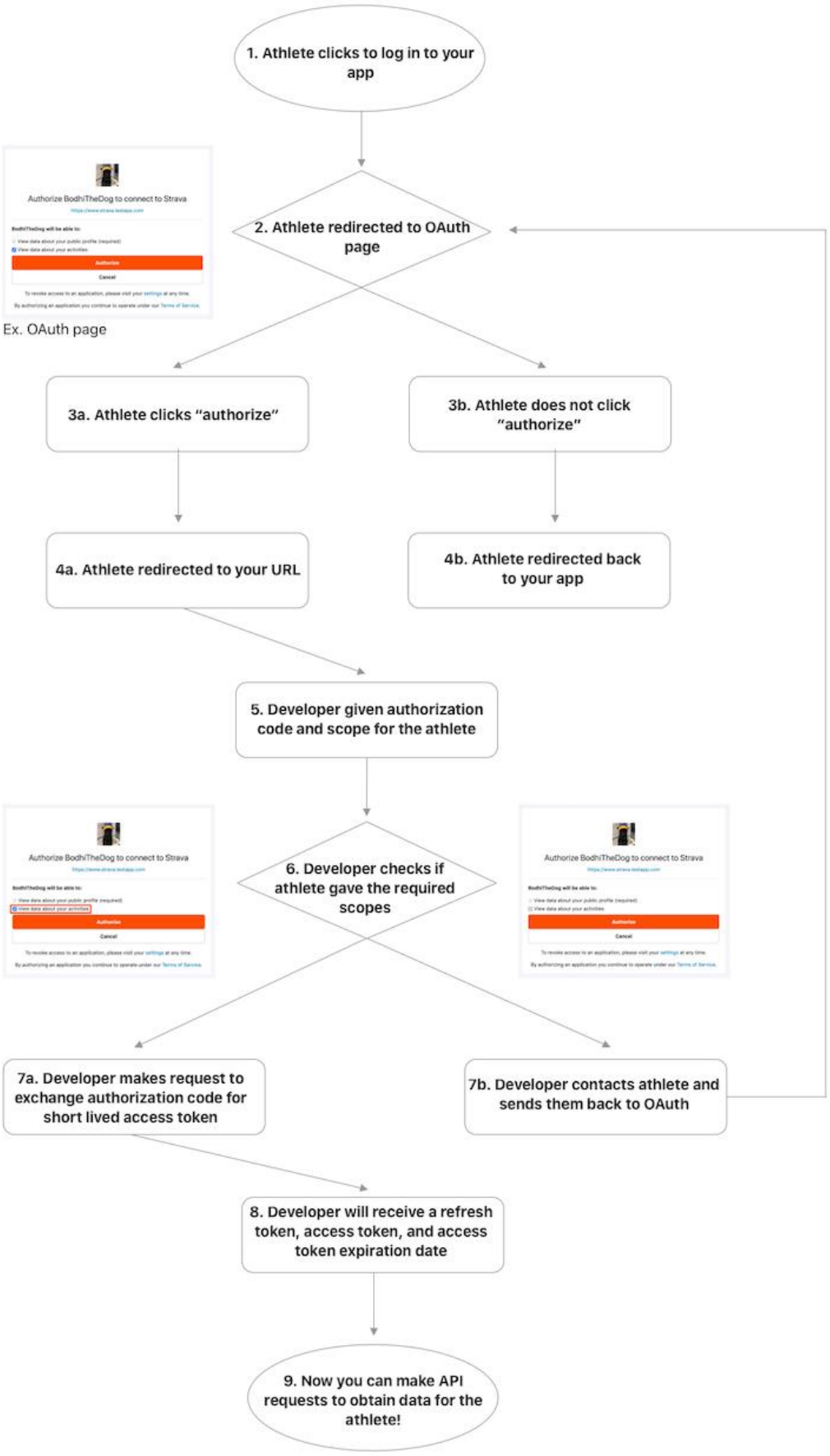
Wanneer een gebruiker inlogt in de Flutter app, wordt hij doorgestuurd naar de Strava OAuth authenticatiepagina. De app stuurt een verzoek naar de autorisatieserver van Strava, inclusief de benodigde credentials en scopes die de gevraagde permissies aangeven. Deze scopes worden bepaald door de developers van de app en bepalen hoeveel toegang de applicatie heeft tot de gegevens van de geverifieerde gebruiker. In het geval van de Tour de icapps heeft de applicatie toegang tot alle openbare profielgegevens en alle activiteiten.

Strava toont de gebruiker vervolgens een inlogpagina, waar hij zijn Strava-gegevens kan invoeren. Zodra de gebruiker succesvol inlogt, valideert de server van Strava de gegevens en genereert een autorisatiecode. Deze code wordt dan teruggestuurd naar de backend server van de Flutter app of direct naar de app, afhankelijk van de implementatie.

De Flutter app kan dan de autorisatiecode uitwisselen met Strava's token endpoint om een access token en een refresh token te verkrijgen. Het access token vertegenwoordigt de autorisatie van de gebruiker om toegang te krijgen tot zijn Strava-gegevens, terwijl het refresh token de app in staat stelt om een nieuw access token te verkrijgen zonder interactie van de gebruiker.

Met het access token kan de Flutter app namens de gebruiker geautoriseerde API verzoeken doen aan de Strava servers. Hierdoor kan de app activiteitsgegevens ophalen, nieuwe activiteiten uploaden en andere Strava-gerelateerde bewerkingen uitvoeren. Het access token heeft een vervaltijd, maar het kan worden vernieuwd met het eerder verkregen refresh token om toegang te behouden.

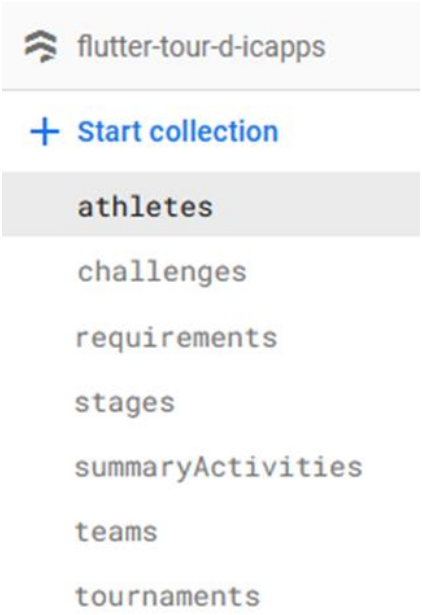
Op de volgende pagina wordt een grafiek getoond die de authenticatiestroom van een SSO-sessie met Strava visualiseert.



Figuur 12: Strava SSO authenticatie flow

6.2 Firestore

In de Tour de icapps is Firebase een kernonderdeel van de applicatie. Meer specifiek werd de Firestore-database gebruikt als belangrijkste opslagmethode voor dit project. Firestore slaat alle gebruikersgegevens op en dient als configuratiepunt voor het maken, bijwerken en verwijderen van gegevens met betrekking tot een toernooi. In het volgende deel zal ik de Firestore database, waarin alle gegevens van de applicatie worden opgeslagen, bespreken en elke verzameling in detail uitleggen.



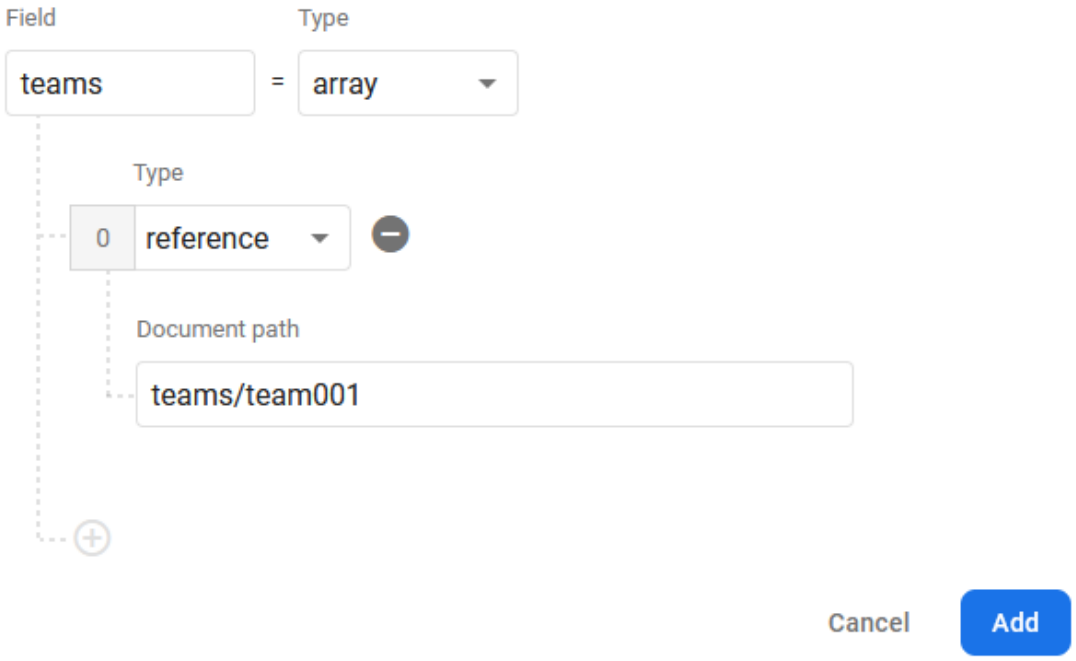
Figuur 13: alle collecties die zijn opgeslagen in Firestore

6.2.1 Collectie tabellen

In het onderstaande gedeelte worden alle bestaande collecties binnen de Firestore database getoond met hun respectievelijke namen, velden, datatypes per veld en de beschrijving die aangeeft welke functie elk veld heeft. Er zijn bepaalde collecties die automatisch gegenereerd worden of data bevatten die ontvangen worden van publieke API's zoals Strava, zoals de Athletes collectie. Een link naar de datamodellen die overeenkomen met deze verzamelingen zal worden opgenomen.

In deze collecties zullen er verschillende gevallen zijn van Lists die gebruikt worden. Omdat Firestore geen relationele database is en gemaakt is in NoSQL, kunnen deze lijsten niet rechtstreeks gedeclareerd worden. In plaats daarvan gebruikt Firestore een techniek die "referenties" heet. Elke referentie verklaart een verbinding tussen documenten opgeslagen in Firestore, gedefinieerd door het veldtype "referentie" en een documentpad gedeclareerd als "collectie/documentId".

Bijvoorbeeld, om een team binnen de collectie "teams" met document ID "team001" te linken aan een toernooi, kan je een veld aanmaken in een toernooidocument van het type "array", gevolgd door een veld van het type "reference" met de waarde "teams/team001", zoals hieronder getoond.



Figuur 14: voorbeeld van het verklaren van een reference

Naam collectie	Naam veld	Data type	Omschrijving
Athletes	Een gedetailleerde uitleg van de DetailedAthlete dataobjecten die zijn opgeslagen in deze collectie kun je vinden in de officiële Strava documentatie.		
SummaryActivities	Een gedetailleerde uitleg van de SummaryActvitity dataobjecten die zijn opgeslagen in deze collectie kun je vinden in de officiële Strava documentatie.		
Tournaments	ID	String	Een unieke identifier, automatisch gegenereerd door Firestore.
	Name	String	Een naam gegeven aan een toernooi
	Stages	List<Stage>	Een lijst van stages verbonden aan een toernooi
	Teams	List<Team>	Een lijst van teams verbonden aan een toernooi

	Start_date	Int	Een epoch tijdstempel die de begindatum van het toernooi aangeeft
	End_date	Int	Een epoch tijdstempel die de einddatum van het toernooi aangeeft
Stages	ID	String	Een unieke identifier, kan automatisch gegenereerd worden door Firestore
	Type	Enum/String	Een enum dat aangeeft welk type stage de atleten moeten afleggen: Rust: een ruststage geeft aan dat er die dag geen wedstrijd is. Tijdrit: een tijdrit stage geeft aan dat er een tijdrit wordt gereden. Vrije stage: een vrije stage geeft aan dat de atleten vrij zijn om te rijden zoals ze willen, er zijn geen specifieke doelen. Training: een trainingsstage is een etappe waarin de atleten vooral voor hun plezier kunnen rijden om zich voor te bereiden op komende stages. Meestal zitten hier leuke challenges aan vast. Groepsrit: een speciaal soort stage waarbij alle deelnemende atleten samen rijden. Algemeen: een algemene stage wordt over het algemeen gezien als een startstage, met

			challenges voor de warming-up.
	Start_date	Int	Een epoch tijdstempel die de begindatum van het toernooi aangeeft
	End_date	Int	Een epoch tijdstempel die de einddatum van het toernooi aangeeft
	Activities	List <SummaryActivity>	Een lijst van alle activiteiten en hun atleten gekoppeld aan een stage
	Challenges	List<Challenge>	Een lijst van alle challenges gekoppeld aan een stage
	Requirements	List<Requirements>	Een lijst van alle requirements gekoppeld aan een stage
Requirements	ID	String	Een unieke identifier, kan automatisch gegenereerd worden door Firestore
	Criteria	Int	Een integer dat samen met de requirement kan worden doorgegeven
	Type	Enum/String	Een enumtype dat aangeeft aan welk type vereisten een activiteit moet voldoen om in aanmerking te komen voor scoreberekening. Dit type wordt hieronder verder uitgelegd.
Challenge	ID	String	Een unieke identificatie, kan automatisch worden gegenereerd

	Criteria	Int	Een optioneel geheel getal dat kan worden meegegeven met de challenge
	Ranking	List<Team>	Een lijst van teams met de huidige rangschikking van alle teams binnen een challenge.
	Type	Enum/String	Een enumtype dat aangeeft aan welk type challenge een activiteit moet voldoen voordat deze in aanmerking komt voor scoreberekening. Dit type wordt hieronder verder uitgelegd.
Teams	ID	String	Een unieke identifier, kan automatisch gegenereerd worden door Firestore
	Name	String	Een naam gegeven een groep van atleten
	Athletes	List<Athlete>	Een lijst met atleten die bij een Team horen

6.2.2 Requirement type

Het requirement type is een enumerable die weergeeft welke berekening moet worden uitgevoerd om een activiteit, of lijst van activiteiten, als geldig te beschouwen. Zodra deze activiteiten geldig zijn bevonden, kunnen ze worden gebruikt om de rankings voor elke uitdaging te berekenen. In de onderstaande tabel wordt elk type in detail beschreven.

type	omschrijving
atLeastXKmPersonal	Om een enkele activiteit als geldig te beschouwen, moet de totale afgelegde afstand tijdens de activiteit ten minste X kilometer zijn.
atLeastXHmPersonal	Om een enkele activiteit als geldig te beschouwen, moet het totaal aantal hoogtemeters dat tijdens de activiteit is afgelegd ten minste X meter zijn.

atLeastXKmTotal	Alle activiteiten gekoppeld aan een etappe worden eerst gegroepeerd per team, waarna de som van de afgelegde afstand door het hele team wordt berekend. Als de som groter is dan het criterium X, dan wordt die lijst van activiteiten als geldig beschouwd.
atLeastXHmTotal	Alle activiteiten gekoppeld aan een etappe worden eerst gegroepeerd per team, waarna de som van de hoogtemeters gedaan door het hele team wordt berekend. Als de som groter is dan het criterium X, dan wordt die lijst van activiteiten als geldig beschouwd.

6.2.3 Challenge type

Het challenge type is een opsomming die aangeeft welke berekening moet worden uitgevoerd om de rangschikking voor elke uitdaging te berekenen. Voordat elke berekening wordt uitgevoerd, worden alle activiteiten in een etappe gegroepeerd per team. Daarna wordt elke lijst van activiteiten per team door de berekeningen gehaald en vervolgens in een klassement geplaatst.

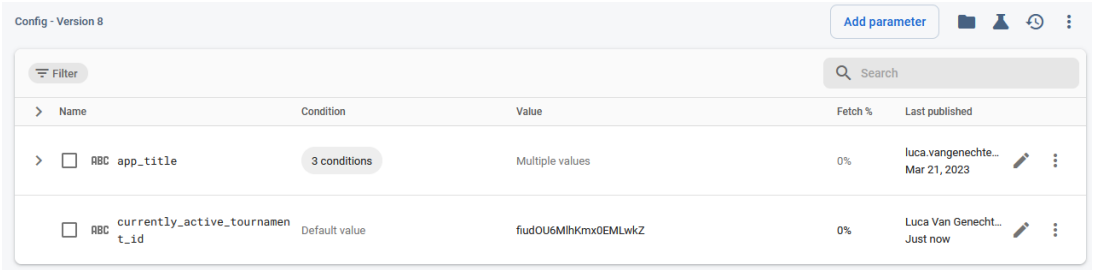
type	omschrijving
shortestTimeSpentRiding	Het gemiddelde van alle rijtijd per activiteit. Het team met de laagste gemiddelde tijd staat op de eerste plaats.
mostTimeSpentRiding	De som van alle rijtijd per activiteit. Het team met de langste tijd staat op de eerste plaats.
highestMaxSpeed	Het gemiddelde van alle maximale snelheden per activiteit. Het team met de hoogste maximale snelheid staat bovenaan.
highestAvgSpeed	Het gemiddelde van alle gemiddelde snelheden per activiteit. Het team met de hoogste gemiddelde snelheid staat bovenaan.
firstToXKilometerTotal	Eerst wordt de som van alle afgelegde afstanden per activiteit berekend. Daarna wordt de rijtijd berekend. Als de totale afgelegde afstand het criterium X bereikt, wordt het team met de minste rijtijd als eerste gerangschikt.
firstToXElevationMeterTotal	Eerst wordt de som van alle hoogtemeters per activiteit berekend. Daarna wordt de rijtijd berekend. Als de totale afgelegde afstand het criterium X bereikt, wordt het team met de minste rijtijd als eerste gerangschikt.
highestKilometerTotal	De som van alle afgelegde afstanden per activiteit. Het team met de meeste kilometers staat op de eerste plaats.

highestElevationTotal	De som van alle hoogtemeters die per activiteit zijn afgelegd. Het team met de meeste hoogtemeters staat bovenaan.
nicestStravaPicture	Een aangepaste uitdaging. Hieraan is geen klassement gekoppeld, maar de winnaar wordt uitgeroepen door middel van stemmen.
nicestOutfit	Een aangepaste uitdaging. Hieraan is geen klassement gekoppeld, maar de winnaar wordt uitgeroepen door middel van stemmen.
nicestStravaArt	Een aangepaste uitdaging. Hieraan is geen klassement gekoppeld, maar de winnaar wordt uitgeroepen door middel van stemmen.
mostKudosTotal	De som van alle verzamelde kudo's per activiteit. Het team met de meeste kudo's staat bovenaan.

6.3 Remote Config

Firebase Remote Config is een krachtige functie van het Firebase-platform waarmee developers het gedrag en uiterlijk van hun mobiele of webapplicaties dynamisch kunnen aanpassen zonder dat daarvoor app-updates nodig zijn. Door gebruik te maken van Remote Config kunnen developers key-value paren definiëren in de Firebase console, die vervolgens worden opgehaald door de app tijdens runtime.

Dit maakt on-the-fly configuratiewijzigingen mogelijk, zoals het aanpassen van feature flags, inhoudsvariaties of UI-instellingen, zonder dat de app opnieuw gecompileerd of uitgerold hoeft te worden. Met Firebase Remote Config kunnen developers efficiënt A/B-tests uitvoeren, gepersonaliseerde ervaringen leveren en snel itereren op app-configuraties om de betrokkenheid en tevredenheid van gebruikers te verbeteren. In het geval van Tour de icapps kan het worden gebruikt om de huidige actieve toernooi ID on the fly te wijzigen.



Figuur 15: screenshot van Firebase remote config

6.4 Crashlytics

Firebase Crashlytics is een robuuste crashrapportagetool van Firebase waarmee developers app-crashes in real-time kunnen monitoren en analyseren. Wanneer Crashlytics wordt geïntegreerd in een mobiele of webapplicatie, registreert het automatisch crashrapporten en biedt het uitgebreide inzichten in de hoofdoorzaken van crashes.

Het biedt gedetailleerde crashlogs, stack traces en relevante metadata, zodat developers snel problemen kunnen identificeren en prioriteren voor debugging

en oplossing. Crashlytics kan worden gebruikt om crashvoorvallen te traceren, inzicht te krijgen in de specifieke omstandigheden die tot crashes leiden en uiteindelijk de stabiliteit van apps en de gebruikerservaring te verbeteren.

Crashlytics werkt echter niet op Flutter for Web omdat het specifiek is ontworpen voor native mobiele platformen. Met Flutter for Web kunnen developers webapplicaties bouwen met behulp van het Flutter-framework, en het vertrouwt op andere onderliggende technologieën dan native mobiele platformen. Als gevolg hiervan heeft Crashlytics geen directe ondersteuning of integratie voor Flutter-gebaseerde webapplicaties, en alternatieve crash-rapportage oplossingen moeten mogelijk worden overwogen voor web-specifieke omgevingen.

7 CI/CD

7.1 Bitbucket

Bitbucket is een webgebaseerd versiebeheerplatform dat is ontworpen om het beheer van broncode-repositories te vereenvoudigen en te stroomlijnen. Het biedt een samenwerkingsomgeving voor softwareontwikkelingsteams om wijzigingen in de code effectief op te slaan, bij te houden en te beheren.

Bitbucket ondersteunt zowel Git als Mercurial, zodat developers zelf hun versiebeheersysteem kunnen kiezen. Met Bitbucket kunnen developers repositories maken om hun code op te slaan, aftakken om aan nieuwe functies of bugfixes te werken en pull requests maken voor code-reviews.

Het biedt krachtige samenwerkingsfuncties zoals inline commentaar, geïntegreerde issue tracking en integratie met populaire ontwikkeltools. Bitbucket kan worden gebruikt voor versiebeheer, samenwerking in code en het beheren van de levenscyclus van softwareontwikkeling. Het stelt teams in staat om efficiënt samen te werken, wijzigingen bij te houden, de kwaliteit van de code te waarborgen en de continue integratie en oplevering van softwareprojecten te vergemakkelijken.

7.2 Firebase

Firebase Hosting is een krachtige hostingservice van Firebase die het deployment- en hostingproces voor webapplicaties vereenvoudigt. Met Firebase Hosting kunnen developers hun statische of dynamische webcontent, inclusief HTML, CSS, JavaScript en assets, eenvoudig en met één opdracht uitrollen naar een wereldwijd content delivery network (CDN). Het CDN zorgt voor een snelle en betrouwbare levering van content aan gebruikers wereldwijd, wat resulteert in een lage latentie en betere prestaties.

Firebase Hosting biedt ook functies zoals aangepaste domeininstellingen, beheer van SSL-certificaten en URL-omleidingen. Het kan worden gebruikt om zowel applicaties met één pagina als applicaties met meerdere pagina's te hosten, waardoor het geschikt is voor een breed scala aan webprojecten.

In termen van Flutter for Web werkt Firebase Hosting naadloos en efficiënt. Het ondersteunt het hosten van op Flutter gebaseerde webapplicaties, waardoor developers hun Flutter for Web-projecten moeiteloos kunnen implementeren en kunnen profiteren van de schaalbaarheid en prestatievoordelen die Firebase Hosting biedt.

7.3 Jenkins

Jenkins is een open-source automatiseringsserver die een robuust framework biedt voor het bouwen, testen en implementeren van softwareprojecten. Het stelt developers in staat om verschillende stadia van de levenscyclus van softwareontwikkeling te automatiseren, zoals het bouwen van code, het uitvoeren van tests en het implementeren van applicaties.

Jenkins werkt op basis van een gedistribueerde architectuur, waarbij jobs of taken worden uitgevoerd op externe servers die agents of slaves worden genoemd. Het ondersteunt een breed scala aan plugins en integraties, waardoor naadloze integratie met verschillende ontwikkeltools, versiebeheersystemen en implementatieplatforms mogelijk is. Jenkins kan worden gebruikt voor continue

integratie (CI), waardoor developers hun codewijzigingen vaak kunnen integreren en problemen vroegtijdig kunnen detecteren.

Daarnaast faciliteert Jenkins continuous delivery en continuous deployment (CD), waarbij de release en deployment van software wordt geautomatiseerd. Met zijn uitgebreide plugin-ecosysteem en flexibiliteit is Jenkins een veelzijdige automatiseringsserver die kan worden aangepast aan de specifieke behoeften van softwareontwikkelingsteams.

7.4 Compatibiliteit

De compatibiliteit tussen Bitbucket, Jenkins en Firebase Hosting zorgt voor de naadloze vorming van een robuuste CI/CD-pijplijn. Bitbucket dient als versiebeheerplatform, waarmee teams hun broncode-repositories kunnen beheren en eraan kunnen werken.

Jenkins, als automatiseringsserver, integreert met Bitbucket om builds, tests en implementaties te triggeren op basis van codewijzigingen. Jenkins-pijplijnen kunnen worden ingesteld om Bitbucket-repositories te controleren, automatisch de CI/CD-workflow te starten wanneer wijzigingen worden gedetecteerd en taken uit te voeren zoals het bouwen en testen van de code.

Zodra de code de vereiste tests doorstaat, kan Jenkins de applicatie naadloos uitrollen naar Firebase Hosting. Firebase Hosting biedt een betrouwbaar en schaalbaar hostingplatform voor webapplicaties en zorgt voor een snelle en wereldwijde levering van content via zijn content delivery network (CDN). Door de sterke punten van Bitbucket, Jenkins en Firebase Hosting te combineren, kunnen developers een end-to-end CI/CD-pijplijn opzetten die het ontwikkelproces vereenvoudigt, de kwaliteit van de code verbetert en de implementatie van hun webapplicaties versnelt.

8 SCHEMA

Tijdens de eerste 3 weken van mijn stage zal mijn focus vooral liggen op het opbouwen van mijn kennisbasis. Ik zal proberen zoveel mogelijk over Flutter voor het web op te nemen en een duidelijk beeld te krijgen van wat me te wachten staat tijdens de realisatiefase. Deze periode zal bestaan uit het leren over icapps als bedrijf en het werken aan een startersproject genaamd de Flutter Beer App.

De Flutter Bier App dient als startpunt voor nieuwe medewerkers en stagiaires om zich vertrouwd te maken met het Flutter framework. Het leert ook het gebruik van verschillende tools, scripts en templates die worden aangeboden door icapps. Ik heb drie weken de tijd gekregen om dit zandbakproject te verkennen, met specifieke kleine doelen voor ogen om de basis Fluttertechnieken te leren.

Zodra ik deze technieken onder de knie heb, ga ik verder met het opzetten van het Tour de icapps project. icapps biedt een starter template voor al hun nieuwe applicaties, die een uitgebreide toolkit bevat met hulpprogramma's zoals een debug scherm, project structuur en een lokale opslag database. De Bier App is gebouwd op hetzelfde starterproject.

Na het afronden van de eerste 3 weken, begin ik aan het Tour de icapps project zelf tijdens de realisatiefase. De realisatiefase zal worden opgedeeld in vijf sprints over een periode van 10 weken, waarbij de focus ligt op de volgende gebieden:

1. Algemene opzet van het project:
   - Configuratie van Strava integratie
   - Firebase configuratie
   - Single Sign-On implementatie
2. Strava interactie:
   - Activiteiten ophalen van ingelogde gebruikers
3. Etappes en toernooien opzetten:
   - De toernooistructuur definiëren in Firebase
   - De toernooistructuur weergeven in de applicatie
4. Beginscherm & scoreberekening:
   - Een startpagina maken om gebruikers te begroeten en een countdown tot het volgende toernooi weer te geven
   - Een rankingsysteem implementeren om scores voor het hele toernooi weer te geven
   - Een ranglijstsysteem ontwikkelen om scores per fase weer te geven
5. Kalender & landing:
   - Een kalenderfunctie implementeren om alle etappes, hun vereisten en uitdagingen weer te geven
   - De laatste aanpassingen doen om ervoor te zorgen dat het project veilig landt
   - Documentatie voor het project voltooien
   - Deze vijf sprints vormen de kern van de realisatiefase en geven richting aan de ontwikkeling van het Tour de icapps-project.

9 CONCLUSION

Als ik terugkijk op dit project, blijf ik achter met een heel prettig gevoel. Het eindresultaat van de stage is een volledig functionele webapplicatie, volledig ontwikkeld met Flutter, met een aangenaam maar professioneel interface. De meeste gewenste functionaliteiten zijn aanwezig en de feedback die ik onderweg heb gekregen heeft me geholpen om deze applicatie om te vormen tot een volwaardige Flutter for Web applicatie.

Persoonlijk denk ik dat dit project een geweldige afsluiter was voor mijn middelbare schoolcarrière, omdat ik mezelf hiermee kon bewijzen als een onafhankelijke en professionele ontwikkelaar. Het doel van dit project was niet alleen om Hannes een goed alternatief te bieden voor handmatig werk met Strava, maar ook om te dienen als een bron van kennis. Het werd een kans om niet alleen meer te leren over Flutter for Web en zijn vele toepassingen, maar ook om de kennis die ik onderweg opdeed te verspreiden.

Dit project heeft me veel nieuwe dingen geleerd over Flutter waar ik voorheen geen idee van had, en ik ben enthousiast om in de toekomst meer met deze technologie te kunnen werken. Naast het verbeteren van verschillende harde vaardigheden, bleek deze stage ook een waardevolle leerervaring te zijn voor verschillende soft skills, aangezien dit een van de eerste ervaringen was die ik heb gehad als Flutter-ontwikkelaar in een groot bedrijf. Door de geweldige begeleiding die ik kreeg van mijn supervisors en de verschillende developers om me heen, en de vele teambuilding oefeningen onderweg, heeft het me echt geholpen om te groeien.

Natuurlijk ben ik niet de enige die tevreden moet zijn met het eindresultaat, de klant, Hannes, moet ook tevreden zijn. Ik hoop echt dat ik een levensvatbaar product heb kunnen maken dat nog vele jaren gebruikt kan worden binnen icapps.