


# Efficiënter Hybride Applicaties Ontwikkelen

## Eindverslag.


**Auteur:** Jordy Robin ten Den
**Studentnummer:** 450425
**Opleiding:** HBO ICT SE
**Onderwijsinstelling:** Saxion Hogeschool Enschede
**Datum:** 13-06-2021
**Begeleidend docent:** Eelco Jannink
**Eerste begeleider:** Harm-Jan Hazelhorst
**Tweede begeleider:** Jos Hekman
**Versie:** 1.1

# Versiebeheer

| Datum | Opmerking | Auteur | Versie |
|---|---|---|---|
| 03-03-2021 | Opzet van document | Jordy ten Den | 0.01 |
| 04-03-2021 | Urenvergelijking toegevoegd | Jordy ten Den | 0.02 |
| 08-03-2021 | Structuur van document wijzigen | Jordy ten Den | 0.03 |
| 11-03-2021 | Toevoegen van Bijlage A, B en C | Jordy ten Den | 0.04 |
| 12-03-2021 | Toelichten van vragen en antwoorden van enquête | Jordy ten Den | 0.05 |
| 15-03-2021 | Uitschrijven van hoofdstuk aanpak | Jordy ten Den | 0.06 |
| 16-03-2021 | Criteria opstellen o.b.v. enquête en interviews | Jordy ten Den | 0.07 |
| 17-03-2021 | Bijlagen toevoegen, verder uitwerken hoofdstuk onderzoeksfase. | Jordy ten Den | 0.08 |
| 18-03-2021 | Uitwerken van hoofdstuk 3. | Jordy ten Den | 0.09 |
| 19-03-2021 | Bijlage C toegevoegd. | Jordy ten Den | 0.10 |
| 22-03-2021 | Samenvattingen geschreven over interviews. | Jordy ten Den | 0.11 |
| 23-03-2021 | Afronding fase 1 – filtering o.b.v. criteria medewerkers | Jordy ten Den | 0.12 |
| 24-03-2021 | Opstellen van beoordelingstabel | Jordy ten Den | 0.13 |
| 26-03-2021 | Afronding fase 1 onderzoek | Jordy ten Den | 0.14 |
| 29-03-2021 | Uitwerken fase 2 onderzoek | Jordy ten Den | 0.15 |
| 31-03-2021 | Afronding fase 2 onderzoek | Jordy ten Den | 0.16 |
| 01-04-2021 | Bijlage D en F toegevoegd | Jordy ten Den | 0.17 |
| 02-04-2021 | Aanpak hoofdstuk update | Jordy ten Den | 0.18 |
| 04-04-2021 | Uitwerken fase 3 onderzoek | Jordy ten Den | 0.19 |
| 05-04-2021 | Gereedmaken eerste tussentijdse oplevering | Jordy ten Den | 0.20 |
| 06-04-2021 | Afronding fase 3 onderzoek | Jordy ten Den | 0.21 |
| 07-04-2021 | Bevindingen gestart, structuur iets aangepast | Jordy ten Den | 0.22 |
| 08-04-2021 | Opzet realisatiefase geschreven | Jordy ten Den | 0.23 |
| 09-04-2021 | Feedback van procesbegeleider verwerken | Jordy ten Den | 0.24 |
| 13-04-2021 | Feedback van begeleidend docent verwerken | Jordy ten Den | 0.25 |
| 14-04-2021 | Feedback van begeleidend docent verwerken | Jordy ten Den | 0.26 |
| 19-04-2021 | Schrijven in kop realisatiefase | Jordy ten Den | 0.30 |
| 21-04-2021 | Feedback verwerken van een lezer | Jordy ten Den | 0.31 |
| 26-04-2021 | Toevoegen/schrijven van wekelijkse reflecties | Jordy ten Den | 0.32 |
| 28-04-2021 | Realisatiefase bijwerken | Jordy ten Den | 0.33 |
| 04-05-2021 | Toevoeging van Bijlage G, schrijven in realisatiefase | Jordy ten Den | 0.34 |

| | | | |
|---|---|---|---|
| 11-05-2021 | Bevindingen uitbreiden en begrippenlijst uitbreiden | Jordy ten Den | 0.35 |
| 13-05-2021 | Herschrijven van Figuren en tabellen bijschriften | Jordy ten Den | 0.36 |
| 15-05-2021 | Voorwoord toegevoegd | Jordy ten Den | 0.37 |
| 23-05-2021 | Toevoegingen realisatiefase, Samenvatting toegevoegd | Jordy ten Den | 0.38 |
| 28-05-2021 | Realisatiefase toevoegingen, Conceptversie | Jordy ten Den | 1.0 |
| 03-06-2021 | Verwerken feedback conceptversie | Jordy ten Den | 1.01 |
| 04-06-2021 | Verwerken feedback conceptversie, toevoegingen aanpak | Jordy ten Den | 1.02 |
| 08-06-2021 | Verwerken feedback lezers | Jordy ten Den | 1.03 |
| 09-06-2021 | Toevoegingen discussie, feedback procesbegeleider verwerken | Jordy ten Den | 1.04 |
| 11-06-2021 | Controle gehele document | Jordy ten Den | 1.05 |
| 13-06-2021 | Definitieve versie document | Jordy ten Den | 1.1 |

# Begrippenlijst

**Framework(s) –** Binnen de gebruikte context is dit een omhulsel of extra pakket bovenop de te gebruiken programmeertalen. Dit pakket is ervoor bedoelt om het ontwikkeltraject van een applicatie te versnellen door het gebruik van voor gedefinieerde componenten.

**API –** Application Programming Interface, een API wordt door Mozilla Developers beschreven als: “Een verzameling van functies en regels die bestaan binnen een softwareprogramma waardoor interactie mogelijk wordt met software”. (Mozilla Developers, 2019)

**Native Applicatie(s) –** Native Applicatie wordt door P. Mishra, S. Sachdeva, A. Kumar en A. Kumar beschreven als: “Native mobiele applicaties zijn applicaties welke ontwikkeld zijn met gebruik van een SDK (Software Development Kit) in een platform specifieke programmeertaal. Deze applicaties zijn gelimiteerd tot één platform en hebben de hoogste prestaties en snelheid.” (Mishra et al., 2019, p. 103)

**Cross Platform Development –** Cross Platform Development wordt door P. Mishra, S. Sachdeva, A. Kumar and A. Kumar beschreven als: “Cross Platform Development verwijst naar creatie en ontwikkeling van mobiele applicaties om te functioneren op verschillende platformen (bijvoorbeeld Android, iOS en Windows.).” (Mishra et al., 2019, p. 102)

**Hybride Applicatie(s) –** Hybride Applicatie wordt door P. Mishra, S. Sachdeva, A. Kumar en A. Kumar beschreven als: “Hybride mobiele applicaties zijn applicaties waar zodra de codebase eenmaal is gerealiseerd, op ieder platform inzetbaar is door deze in te pakken in een platform specifiek omhulsel.” (Mishra et al., 2019, p. 103)

**User Interface (UI) –** Volgens Encyclo is een User Interface het deel van een softwareprogramma of applicatie wat de interactie tussen de gebruiker en de computer mogelijk maakt. (Encyclo, z.d.)

**User Experience (UX) –** Volgens Encyclo is User Experience alles wat richt op interactie van een gebruiker in een website of applicatie**.** (Encyclo, z.d.-a)

**Variabele (Informatica) –** Variabele wordt door S. Sau beschreven als: “Een variabele de naam van het gereserveerde gebied dat in het geheugen van de applicatie is toegewezen.” (Sau, 2019)

**Build(en) –** Het compileren van de broncode tot een uitvoerbare applicatie voor een specifiek platform.

**Boilerplate –** Een template of eerste versie van software of framework, om een aantal stappen van de initialisatie van een nieuw project te kunnen overslaan.

**Learning Curve –** De leer curve van een ontwikkeltechniek of framework, op basis van de huidige kennis van een medewerker.

**PWA (Progressive Web Applications)** – Een PWA is een applicatie welke ontwikkeld wordt d.m.v. web technologieën. De intentie van zo’n applicatie om bruikbaar te zijn op ieder platform doormiddel van een webbrowser zoals Google Chrome of Safari.

**Release Candidate –** Een release candidate is in de gebruikte context een versie van een framework, ontwikkeltechniek of native runtime welke kandidaat is om uitgebracht te worden als stabiele versie.

**Native Runtime –** Het achterliggende stuk software wat ervoor zorgt dat webontwikkeltechnieken omgezet worden in of native functies voor verschillende 'native' platformen zoals Android of iOS.

**Dedicated Development Team –** Een toegewijd team met ontwikkelaars welke constant (door)ontwikkelen aan één stuk software of product.

**Plain JavaScript –** De programmeertaal JavaScript zonder bijkomstigheid van een framework.

**Merge Request –** De aanvraag om afgetakt stuk code samen te voegen met een stabiele versie van software.

**View Width** – Een eenheid gebruikt in de programmeertaal CSS, om een element te schalen op basis van de grote van het scherm van het apparaat waar het element op weergegeven wordt.

**Features** – In de gebruikte context betekent dit dat de applicatie voorzien is van een bepaalde functionaliteit.

**Media Query –** Een in CSS geschreven stuk code, waarin op basis van een viewport, specifieke styling toegepast kan worden voor verschillende apparaten zoals; smartphones, tablets en laptops.

**Viewport –** Een viewport is het gedeelte van een webpagina waar een eindgebruiker interactie mee kan hebben. Een viewport van een smartphone is kleiner dan die van een laptop, gezien het verschil in schermformaat.

**Mismatch van parameters –** Bij het aanroepen van een functie, kan er om data gevraagd worden welke meegestuurd dient te worden. Als de meegestuurde data niet klopt met wat er in de functie verwacht wordt, ontstaat er een mismatch.

**IDE (Integrated Development Environment) –** De ontwikkelomgeving van een ontwikkelaar, daar waar de broncode in toegevoegd en gewijzigd wordt.

**Live/Hot Reloading -** Herladen van een applicatie o.b.v. aanpassingen welke plaatsvinden in de broncode, rechtstreeks op het apparaat waar de applicatie op weergegeven wordt tijdens ontwikkeling.

**Refactor –** Broncode herschrijven om een betere aansluiting te ontwikkelen voor de implementatie of herbruikbaarheid van dat specifieke stuk code.

**Minimum Viable Product (MVP)** – Een MVP is volgens (Betekenis MVP, z.d.) “Een vroege, uitgeklede versie van een product waarmee bekenen wordt of dat product rendabel is.”

# Tabellen- en Figuurlijst

## Figuren

## Tabellen

# Voorwoord

Voor u ligt het eindverslag ‘efficiënter hybride applicaties ontwikkelen’. Dit verslag is geschreven in het kader van mijn afstuderen aan de opleiding HBO ICT Software Engineering aan het Saxion te Enschede en in opdracht van het bedrijf CODE14. Vanaf februari 2021 tot en met april 2021 ben ik bezig geweest met de research en vanaf mei 2021 tot juli 2021 met de realisatie van de VYBER-applicatie. Gedurende de gehele afstudeerperiode is dit document tot stand gekomen.

Tijdens mijn vooropleiding applicatieontwikkelaar bij Landstede te Zwolle ben ik in aanraking gekomen met het ontwikkelen van (web)-applicaties. Hierbij heb ik kennis opgedaan van de basis van het ontwikkelen van applicaties. Na het afronden van de mbo-opleiding was ik niet overtuigd dat mijn vakkennis op niveau was voor de commerciële arbeidsmarkt. Mede hierdoor heb ik na een tussenjaar genomen te hebben, de keuze gemaakt om verder te studeren en te beginnen met een hbo-opleiding in hetzelfde vakgebied. Door mijn affiniteit met applicatieontwikkeling en in het specifiek front-end development heb ik tijdens mijn studieperiode voor de minor Creative Design & Technology gekozen en in het laatste jaar Advanced Application Development gevolgd als specialisatie.

Tijdens de gehele studieperiode heb ik met verschillende ontwikkeltechnieken geleerd om applicaties te ontwikkelen. Maar in het bijzonder mijn kennis verbreed ten aanzien van de ontwikkeling van hybride applicaties doormiddel van web-ontwikkelingstechnieken. Omdat deze methode voor het ontwikkelen van cross-platform applicaties mij enorm aansprak.

Ik wil graag mijn bedrijfsbegeleiders Harm-Jan Hazelhorst, Jos Hekman en begeleidend docent Eelco Jannink bedanken voor hun begeleiding, feedback en adviezen tijdens het afstudeertraject. Verder wil ik de collega’s bij CODE14 bedanken voor de medewerking tijdens het onderzoek en de vele spar- en brainstormsessies. Zonder hun had ik dit onderzoek nooit kunnen voltooien.

Tot slot wil ik mijn ouders, zussen en vriendin bedanken voor morele ondersteuning tijdens mijn afstudeerperiode. Zij hebben mij het afgelopen jaar de motivatie gegeven om ook de laatste fase van mijn studie, goed af te ronden.

Ik wens u veel leesplezier toe.

Jordy ten Den

Nijverdal, 12-06-2021

# Samenvatting

**Vraagstuk**

CODE14 wil efficiënter en onderhoudsvriendelijker te werk gaan als het aankomt op het realiseren van mobiele applicaties. Het bedrijf is zich ervan bewust dat er technieken bestaan welke de mogelijkheid bieden om van één codebasis, meerdere applicaties voor verschillende platformen (bijv. iOS, Android) te realiseren. Er is een onderzoek gedaan naar bestaande ontwikkeltechnieken voor het realiseren van dergelijke applicaties.

**Onderzoek**

Aan het begin van de onderzoeksfase is er informatie ingezameld over alternatieve ontwikkelmethoden, zoals softwarepakketten welke aangeven zonder enige programmeerkennis de mogelijkheid te bieden om (mobiele) applicaties te realiseren. Hieruit bleek dat dit soort ontwikkelmethoden nog in een relatief onvolwassen stadium zitten. Mede hierdoor is de keuze gemaakt door te gaan met een onderzoek naar een ontwikkeltechniek welke het beste aansluit op de interne kennis van CODE14.

Het onderzoek is opgedeeld in drie fases. Tijdens de eerste fase is er intern een enquête rondgegaan en zijn er interviews afgenomen bij ontwikkelaars om kennisgebieden te pijlen. Vervolgens zijn er op basis van drie artikelen over het onderwerp, een lijst met geschikte ontwikkeltechnieken opgesteld. In de tweede fase zijn er criteria opgesteld o.b.v. de verkregen informatie uit de eerste fase van het onderzoek. Daarna zijn de gevonden ontwikkeltechnieken o.b.v. deze criteria gefilterd. In de derde en laatste fase zijn er in de drie overgebleven ontwikkeltechnieken prototypes uitgewerkt. Tot slot zijn deze op basis van criteria beoordeeld, hieruit is de best scorende ontwikkeltechniek meegenomen naar de realisatiefase.

**Realisatie**

Tijdens de realisatiefase is er een applicatie ontwikkeld voor een investeringsproject van CODE14, genaamd VYBER (Vyber Cycles, z.d.). Er zijn al een tweetal bestaande native applicaties ontwikkeld welke te vinden zijn op de app- en play store (iOS en Android). CODE14 heeft weinig interne kennis van de gebruikte ontwikkeltechnieken van de bestaande applicaties, dus voor onderhoudsvriendelijkheid en om de bevindingen van het onderzoek aan te tonen is tijdens de afstudeerperiode de ontwikkeling van deze applicatie in gang gezet conform het advies.

**Advies**

Uit het onderzoek is de conclusie getrokken dat van de ontwikkeltechniek Ionic i.c.m. VueJS en Capacitor de beste aansluiting biedt op de eisen en wensen van CODE14. Dit framework biedt aansluiting op technologieën zoals Overlay (Overlay tech, z.d.) en Storybook (Storybook, z.d.). Ionic heeft een grote community, een eigen ecosysteem, actieve doorontwikkeling en heeft toegang tot alle benodigde functies voor het gemiddelde CODE14 hybride project.

Hierbij zou er gebruik gemaakt kunnen worden van Overlay (Overlay tech, z.d.). Dit is software welke op basis van een applicatie ontwerp broncode genereerd, wat vervolgens gebruikt kan worden tijdens de ontwikkeling van een applicatie welke geschreven wordt in één specifiek framework, namelijk VueJS (You, 2014).

Ook is er een aansluiting gevonden op de huidige ontwikkeltechnieken, waardoor ontwikkelde componenten universeel herbruikbaar zijn tussen verschillende projecten, dit kan gerealiseerd worden doormiddel van een softwarepakket genaamd Storybook (Storybook, z.d.). Wanneer een component volledig uitgewerkt is, kan deze opgeslagen worden in dit softwarepakket. Mocht een gelijkwaardig component dan bij een ander project benodigd zijn, kan deze hieruit worden gehaald en hoeft deze niet in zijn geheel herschreven worden.

**Conclusie**

Met het behalen van de doelstellingen, welke waren geformuleerd in de (deel)vragen, is het mogelijk gebleken om efficiënter hybride applicaties te ontwikkelen. Ook is gebleken dat er over het algemeen efficiënter ontwikkeld kan worden. Er zal constant moeten worden gekeken naar vernieuwingen in het kader van (mobiele) applicatieontwikkeling, omdat de bestaande technologieën met een rap tempo vernieuwen.

# Inhoudsopgave

# 1. Inleiding

## 1.1 Onderwerp

Smartphones en mobiele applicaties zijn niet meer weg te denken uit het dagelijkse leven. Uit onderzoek van het Centraal Bureau voor de Statistiek (2019) blijkt dat 87% van de Nederlandse inwoners tussen de 16 en 75 jaar in 2018 een smartphone gebruikt.

In een tijd met steeds verdergaande digitalisering, neem als voorbeeld online winkelen, is er steeds meer vraag naar de ontwikkeling van mobiele applicaties voor bedrijven. Het proces van het realiseren van een (mobiele) applicatie kan enorm tijdrovend zijn. Bovendien willen opdrachtgevers vaak dat hun applicatie uiteindelijk voor meerdere platformen beschikbaar is, wat er vaak voor zorgt dat niet één, maar meerdere applicaties ontwikkeld moeten worden. Om deze redenen zijn softwareontwikkelaars steeds vaker zoekende naar efficiëntere methoden om applicaties te ontwikkelen.

Toen hybride applicatie frameworks zoals React Native (React Native, 2021), Ionic (Ionic, 2021) en Flutter (Flutter, 2021) op de markt kwamen kregen softwareontwikkelaars de mogelijkheid om binnen één codebase een applicatie te ontwikkelen voor de meeste grotere besturingssystemen, neem hierbij als voorbeeld iOS, Android en Windows. Veel van deze hybride frameworks worden gebaseerd op web-ontwikkelingstechnieken zoals HTML, CSS en Javascript.

## 1.2 Vraagstuk

CODE14 is één van de softwareontwikkelaars welke zoekt naar een efficiëntere manier om mobiele applicaties te ontwikkelen. CODE14 wil graag weten wat de mogelijkheden zijn op gebied van hybride applicaties. Hiermee moet duidelijk worden wat de toepasbaarheid en het toekomstperspectief van de hybride ontwikkeltechnieken zijn.

**De onderzoeksvraag**

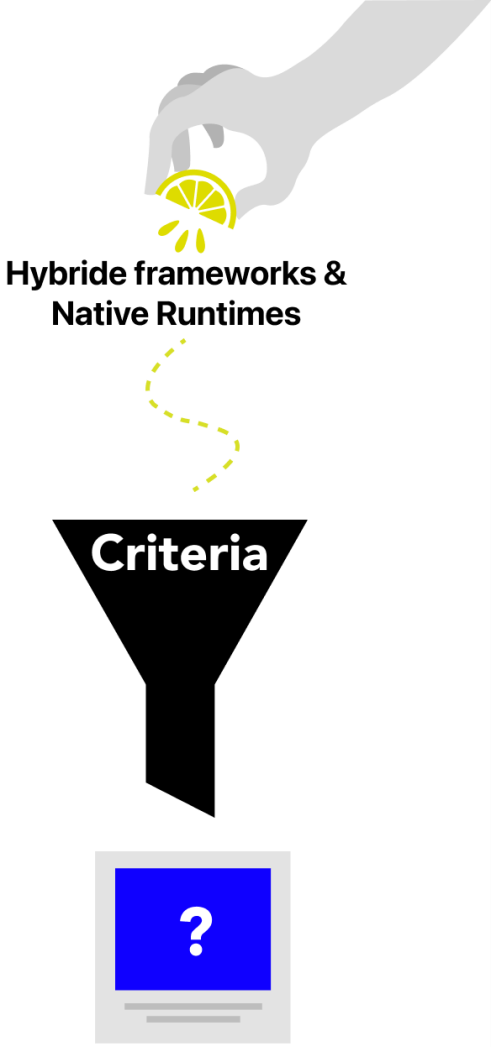
Hoe kan CODE14 op efficiëntere wijze toekomstgerichte hybride applicaties ontwikkelen en onderhouden? En kunnen Low-/No code platformen hierbij van toegevoegde waarde zijn?

**De deelvragen**

- Wat zijn de mogelijkheden van hybride ontwikkeling t.o.v. native ontwikkeling?
- Wat zijn de eisen waar een hybride ontwikkeltechniek aan moet voldoen voor de gemiddelde CODE14 hybride applicatie?
- Welke ontwikkeltechnieken worden het meest gebruikt binnen CODE14, als het gaat om front-end ontwikkeling?
- Wat zijn de voor- en nadelen voor CODE14 welke hybride ontwikkeling met zich meebrengt?
- Hoe kan er gezorgd worden dat de gemiddelde ontwikkelaar werkzaam bij CODE14, de voorkant van een hybride applicatie kan opbouwen?
- Hoe kan er binnen een aansluitende ontwikkeltechniek gezorgd worden dat er herbruikbare code geschreven wordt?

Daarnaast wil CODE14 het huidige investeringsproject van VYBER, de twee native mobiele applicaties voor iOS en Android, volledig over hebben naar een hybride ontwikkelingstechniek. Dit komt doordat de huidige native applicaties moeilijk onderhoudbaar zijn en hier aanzienlijk op bespaart kan worden. Daarnaast zijn de bestaande native applicaties opgezet door oud-medewerkers en is de kennis over native ontwikkeling binnen CODE14 mede hierdoor, schaars.

# 2. Aanpak




## 2.1 Opstartfase

In de opstartfase van de afstudeerperiode is er een plan opgesteld om beroepsproducten te realiseren, waarmee de benodigde beroepscompetenties aangetoond kunnen tonen. Aan het eind van de opstartfase zal er een begin gemaakt worden aan de onderzoeksfase, zoals het opstellen van vragen voor interviews en voor de medewerkers enquête. Tijdens deze fase zal er ook een ontmoetingsgesprek plaatsvinden met de begeleidend docent en de bedrijfsbegeleider(s).

## 2.2 Onderzoeksfase

De onderzoeksfase is opgedeeld worden in vier fases, in de eerste fase is er informatie vanuit de medewerkers opgevraagd. Tijdens de onderzoeksfase is er gebruik gemaakt van het timeboxing principe, hierbij wordt er van tevoren een tijdsbestek gekoppeld aan een onderwerp, waarbij er bij geen doorslaggevende bevindingen, er over wordt gegaan naar een volgend onderwerp.

### 2.2.1 Fase 1 - Informatie inzamelen van medewerkers. (Veldonderzoek)

*Figuur 1 - Afbakenen van frameworks d.m.v. criteria*

Het is van belang om de medewerkers te betrekken bij het onderzoek, aangezien de uitkomst mogelijk een impact kan hebben op de gebruikte technologiestack. Er is een enquête opgesteld waarbij de kennisgebieden van de medewerkers inzichtelijk zijn gemaakt, daarnaast zijn er een aantal vragen gesteld om voorkeuren en hun visie inzichtelijke te krijgen. Mede om de gegeven antwoorden van de enquête te valideren zijn er interviews afgenomen met een drietal personen welke werkzaam zijn binnen CODE14, daarnaast is er binnen één interview specifiek ingegaan op de business aspecten welke het onderzoek met zich mee brengt.

### 2.2.2 Fase 1 - Hybride ontwikkel frameworks verzamelen (Literatuuronderzoek)

Om een duidelijk beeld te krijgen van de huidige ontwikkeltechnieken en frameworks welke gebruikt worden voor hybride applicatieontwikkeling, zal er eerst een lijst met frameworks worden opgesteld o.b.v. literatuuronderzoek. Hierbij wordt er nog niet gekeken naar toepasbaarheid binnen de organisatie.

### 2.2.3 Fase 2 - Criteria opstellen o.b.v. literatuuronderzoek en veldonderzoek

Er is een lijst met criteria opgesteld voor de afbakening van de frameworks welke voortgekomen zijn uit het literatuuronderzoek. Er zijn twee lijsten met criteria opgesteld, één lijst met criteria

vanuit CODE14 (Interviews, gesprekken en enquête) en één lijst met criteria voortkomend uit eigen ervaring en het voorafgaande literatuuronderzoek, deze is gebruikt om de frameworks een cijfer te geven op verschillende belangrijke aspecten. In de criteria vanuit CODE14 zitten een aantal must-have's, dit is criteria waar een hybride framework minimaal aan moest voldoen. Om deze reden zijn de gevonden frameworks eerst gefilterd op basis van deze criteria en zijn de overgebleven ontwikkeltechnieken meegenomen naar een tweede filter fase, waarbij er dieper is in gegaan op de specificaties.

### 2.2.4 Fase 2 – Filteren van frameworks

Op basis van de criteria zijn de gevonden frameworks gefilterd. Hierbij is er alleen gekeken naar de belangrijkste criteria, waar een framework absoluut aan moet voldoen. De overgebleven frameworks zijn vervolgens meegenomen worden naar de volgende fase.

### 2.2.5 Fase 3 – Prototypes

In totaal zijn er drie frameworks overgebleven. In deze frameworks is er een klein prototype ontwikkeld welke aan moet tonen dat het framework aan de belangrijkste criteria vanuit CODE14 voldoet. Er is praktijkervaring opgedaan, welke vervolgens in de eindbeoordeling van deze drie frameworks is meegenomen.

### 2.2.6 Fase 3 – Frameworks beoordelen

Er is een lijst met belangrijke onderwerpen opgesteld o.b.v. veld- en literatuuronderzoek. Hierbij is er gekeken naar belangrijke onderwerpen voor het ontwikkelen van hybride applicaties zoals; Initialisatiefase, documentatie, community, platformen, learning curve, toepasbaarheid, toegankelijkheid en prestatie.

### 2.2.7 Afronding - Low code technieken

Er is gekeken of er bij het overgebleven framework een toepassing mogelijk is voor een low code oplossing. Hierbij is er gekeken of dit efficiënter is dan de huidige workflow, of dit past binnen de huidige technologiestack en wat hiervoor benodigd is.

### 2.2.8 Presentatie Development Board

Nadien van de onderzoeksfase zijn de bevindingen gepresenteerd voor het Development Board van CODE14. Hierbij is in een presentatie van 10 tot 15 minuten verteld hoe het onderzoek is uitgevoerd, wat de bevindingen zijn en wat het advies is om de komende hybride projecten op te bouwen. Aan het eind is er groen licht gegeven om de realisatiefase in te gaan met de geadviseerde ontwikkeltechniek

## 2.3 Realisatiefase

### 2.3.1 Requirements

Er zijn requirements opgesteld aan de hand van documentatie welke beschikbaar was voor het VYBER-project. De opgestelde requirements zijn gecontroleerd door de bedrijfsbegeleider en goed bevonden.

### 2.3.2 Schatting

Na het opstellen van de requirements is er een document opgesteld om het bestaande ontwerp op te delen in deeltaken. Op basis van de opgedeelde taken zijn tijdsinschattingen gemaakt om een beeld te krijgen van de verwachte ontwikkeltijd.

### 2.3.3 Projectmanagement

CODE14 werkt volgens de Agile projectmanagement methodiek. Wekelijks zijn er teammeetings ingepland om de voortgang te bespreken en een verwachting te scheppen voor de komende week. De procesbegeleider houdt het issue-bord nauw in de gaten. Kwaliteit wordt gewaarborgd door de controles welke uitgevoerd worden op de code, dit wordt gedaan in de vorm van mergerequests waar een ontwikkelaar van team invest aan wordt gekoppeld. Hierbij wordt er gestuurd om zoveel mogelijk mondeling contact te hebben.

### 2.3.4 Initiële opzet

Het project is opgezet o.b.v. bestaande programmeer paradigmen. De basis opzet van de stylesheets waar variabelen, functies en mixins in staan. Verder is de mappen structuur overzichtelijk ingedeeld, zoals dit ook in andere projecten gedaan is. Met een aparte map voor componenten, waar onderliggend weer verdeling is voor de onderwerpen waar de componenten in gemaakt worden

### 2.3.5 Werkvolgorde

**1) Visueel**

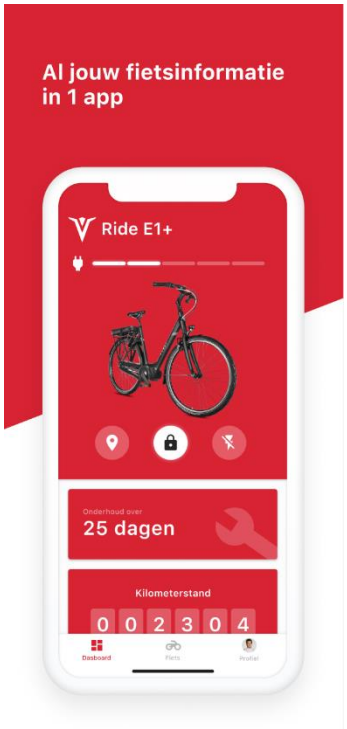
In de eerste weken wordt er puur visueel ontwikkeld. Dit is omdat er enigszins nog aanpassingen gedaan kunnen worden aan het ontwerp, in samenwerking met design expert Christian Lemmers, zal er gekeken worden naar het huidige ontwerp.

**2) Gebruikers functies**

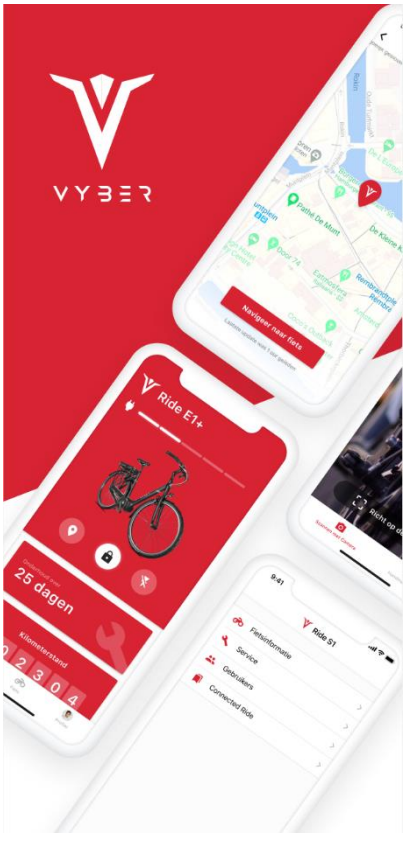
Na het uitwerken van het visuele ontwerp zijn de gebruikers functies uitgewerkt. Denk hierbij aan functionaliteiten zoals: Vyber-ID Registreren, Account aanmaken, Inloggen, Profielgegevens inzien en bijwerken, Profielfoto toevoegen, etc.

**3) E-bike functies**

Hierna is er gewerkt aan de functies voor de fietsen, het correct weergeven van de status en statistieken van de fiets, zoals: GPS-locaties, batterij percentage, kilometerstand, status van onderhoud, het op slot zetten van de fiets en het inschakelen van de kill-modus.




*Figuur 4 - E-bike dashboard van de vyber applicatie*




*Figuur 2 - E-bike overzicht, dashboard, QR scanner en locatie van vyber applicatie*


*Figuur 3 - Vyber kill-modus inschakelen*

# 3. Onderzoek

In dit hoofdstuk worden de gebruikte methoden in kaart gebracht en welke stappen er ondernomen zijn om tot het eindresultaat te komen.

## 3.1 Huidige technologieën

Er is gekeken naar welke technologieën CODE14 al gebruikt om efficiënter te werk te gaan. Vanuit het front-end team van CODE14 kwamen de volgende dingen naar voren:

**Deelvraag: Welke ontwikkeltechnieken worden het meest gebruikt binnen CODE14, als het gaat om front-end ontwikkeling?**

1) Herbruikbare componenten d.m.v. Storybook (Storybook, z.d.).
2) Een **NuxtJS boilerplate** voor tijdswinst in het opzetten van een project.
3) De meeste medewerkers weten hoe ze basis-styling uit **Figma** (Figma, z.d.) ontwerpen kunnen halen.
4) Hybride applicaties zijn in het verleden gemaakt in **Ionic i.c.m. Angular** en **Cordova**.
5) Veel VueJS gerelateerde frameworks zoals **Gridsome, Nuxt en Vue i.c.m. Laravel**.

## 3.2 Fase 1 – Informatie verzamelen

### 3.2.1 Interviews met medewerkers

De vragen uit de enquête zijn gebaseerd op de eerste interviews welke met een drietal werknemers is afgenomen. Hiermee wordt een eerste inzicht verkregen in de technologiestack, waarom er voor hybride ontwikkeling wordt gekozen, maar ook wat de voor- en nadelen van hybride ontwikkeling zijn.

Er zijn in totaal drie mensen geïnterviewd, als eerst Leon Kosters (Lead UX/UI Design, Front-end Developer), vervolgens Mathijs Pluimers (Mede-eigenaar, algemeen directeur) en tot slot Christian Lemmers (UI/UX Designer en Front-end Developer).

#### 3.2.1.1 Interview met Leon Kosters

*In **Bijlage A** is het volledige interview met Leon te vinden, deze is opgenomen en vervolgens getranscribeerd.*

**Waarom het interview met Leon?**

Leon is de lead-designer van CODE14 en doet daarbij ook front-end ontwikkeling. Leon heeft zes jaar ervaring bij CODE14 en daarmee grotendeels van de veranderingen in de technologiestack meegemaakt. Mede hierom is Leon de perfecte kandidaat om persoonlijk te interviewen. Daarnaast is hij zonder enige programmeerervaring binnengekomen als Product Designer en weet hij hoe groot de learning curve kan zijn van een ontwikkeltechniek. Leon is ook onderdeel

van het management van CODE14 en heeft hij een heel goed inzicht in wat er allemaal speelt binnen de organisatie

**Samenvatting**

Leon geeft meerdere malen aan dat er veel interne kennis van het ontwikkel framework VueJS (You, 2014) is. Daarnaast wordt Nuxt (*Nuxt.Js - The Intuitive Vue Framework*, z.d.) sinds kort gezien als de nieuwe standaard binnen CODE14. In het verleden is de keuze voor hybride ontwikkeling vaker gevallen op Ionic, deze ontwikkeltechniek kwam destijds qua technologiestack het dichtst bij de interne kennis. Leon is van mening dat Cordova/Capacitor al grotendeels van de native functionaliteiten afvangen en kon tijdens het interview niet op een native functie komen welke niet ondersteund zou worden.

***Keuzes in het verleden***

De keuze die in het verleden zijn gemaakt om voor Ionic te kiezen zijn op basis van community en toepasbaarheid binnen de organisatie. Hierbij is geen rekening gehouden met learning curve en is er niet om input gevraagd vanuit de organisatie over het onderwerp 'hybride frameworks'.

Leon geeft aan dat bij alle huidige mobiele applicaties die tot nu toe gerealiseerd zijn, 80% van de tijd gestoken wordt in het uitwerken van de user interface en dat hier een maatwerk API aan verbonden zit. Er wordt relatief weinig gebruik gemaakt van de native functies, wanneer dit wel gebeurt zijn het volgens hem oppervlakkige dingen zoals camera, lokale opslag, QR/barcode scanner en bluetooth, welke relatief eenvoudig implementeerbaar zouden moeten zijn in bestaande hybride frameworks.

***No-/Low code platformen***

Leon is van mening dat No-/Low code platformen nog niet volwassen genoeg zijn om volledig in afhankelijkheid in gebruik te nemen, zelf werkt hij veel met Framer om prototypes uit te werken van gemaakte ontwerpen, dit vertaalt op de achtergrond naar React code. Hij geeft hierbij aan dat React (Native) mogelijk interessant kan zijn in de toekomst, maar niet op korte termijn. Aangezien de learning curve daarvoor intern te groot zou zijn en het een hele omslag is als je kijkt naar de huidige technieken. Daarbij geeft hij wel aan te weten dat React Native daadwerkelijk naar een native applicatie compileert en dat dit bij Ionic een WebView is. Maar omdat de doorontwikkelingen omtrent hardware van smartphones enorm snel gaat, vind Leon dit geen valide argument om de omslag te maken naar React Native.

Wel vond Leon het een interessante theorie om low code platformen te combineren met het ontwikkelen van hybride applicaties. Dit zou positieve impact kunnen hebben op de snelheid waarmee individuele componenten opgebouwd kunnen worden. Zonder dat dit afdoet aan de aanpasbaarheid van zo'n component, aangezien je alle broncode hebt.

***Belangrijkste punten uit het interview***

- Zo dicht mogelijk bij de huidige stack blijven.
- Gericht op minimaal 1 á 2 jaar keuze voor nieuwe projecten, wel langer uitvoerbaar.
- Aansluiting op kennis van meerdere medewerkers.
- Lage learning curve, zodat het framework snel opgepakt kan worden in de technologiestack.

#### 3.2.1.2 Interview met Mathijs Pluimers

*In **Bijlage B** is het volledige interview met Mathijs te vinden, deze is opgenomen en vervolgens getranscribeerd.*

**Waarom het interview met Mathijs?**

Mathijs Pluimers is mede-eigenaar van CODE14 en is voornamelijk bezig met de financiële tak van het bedrijf. Aangezien hij de groei en ontwikkelingen van CODE14 heeft meegemaakt, kan hij op business level goed inschatten wat de impact van de keuze voor hybride applicatie zou kunnen zijn.

**Deelvraag: Wat zijn de voor- en nadelen voor CODE14 welke hybride ontwikkeling met zich meebrengt?**

**Samenvatting interview**

Mathijs ziet de keuze voor hybride ontwikkeling vooral terug in de urenbesparing, onderhoudbaarheid en daling van de complexiteit. Daarnaast is het voor het uitwerken van business cases van belang dat de gekozen ontwikkeltechniek minimaal 5 jaar lang werkend blijft, ook op oudere smartphones. Hij ziet dat de trend in hybride ontwikkeltechnieken stijgt en dat dit samenhangt met de upgrades op hardware gebied van de apparaten waar deze applicaties op draaien. Er kan veel tijd bespaard worden door de keuze op hybride te laten vallen. Maar hiervoor zal er in de uitwerking van de verschillende business cases gekeken moeten worden naar de toepasbaarheid van zo'n ontwikkeltechniek en dit brengt risico's met zich mee. Het is daarnaast ook zo, dat door het gebruik van hybride ontwikkeltechnieken de complexiteit daalt. Waardoor er meer medewerkers onderhoud kunnen plegen aan de broncode van een hybride applicatie, dit maakt de personele inzetbaarheid van CODE14 flexibeler. Daarnaast, wordt er tijdens de sollicitatieprocedure voor een front-end ontwikkelaar gefilterd of de sollicitant enige kennis heeft van VueJS.

***Belangrijke punten uit het interview***

- Zo dicht mogelijk bij de huidige stack blijven.
- Business cases moeten minimaal 3 tot 5 jaar te onderhouden zijn.
- Aansluiting op kennis van meerdere medewerkers voor flexibiliteit en daling van complexiteit.

#### 3.2.1.3 Interview met Christian Lemmers

*In **Bijlage C** is het volledige interview met Christian te vinden, deze is opgenomen en vervolgens getranscribeerd.*

**Waarom het interview met Christian?**

Christian werkt al twee jaar voor CODE14 als product ontwerper en front-end developer. Hij heeft in de afgelopen twee jaar veel ervaring opgedaan met de wisselingen in de technologiestack wanneer het aankomt op front-end development en is hij onderdeel van het Development Board van CODE14. Dit Development Board is bedoeld voor onderzoek van verschillende onderwerpen welke directe betrekking hebben op de ontwikkelwijze van CODE14, low-/no code platformen is hierin een onderdeel van Christian.

**Samenvatting**

***Overeenkomsten met Leon***

Christian zijn antwoorden komen grotendeels overeen met de antwoorden welke Leon gaf. Het liefst blijft Christian zo dicht mogelijk bij de huidige technologiestack voor een snelle integratie binnen het bedrijf, daarnaast is flexibiliteit daar ook een belangrijk onderdeel bij. Wel geeft Christian aan dat hij in tegenstelling tot Leon minder aandacht schenkt aan de native stijlgidsen van platformen zoals iOS en Android.

Ook is Christian net als Leon van mening dat een hybride framework niet per definitie UI componenten hoeft aan te leveren. Immers is er binnen CODE14 nog nooit voor een dergelijke oplossing gekozen, wel reikt Christian net als Leon de optie aan om NuxtJS en Capacitor te gebruiken. Om nog dichter bij de huidige technologiestack te blijven en de flexibiliteit en vrijheid te hebben qua UI elementen, aangezien NuxtJS dit niet aanlevert.

***Eigen ervaring met hybride***

Christian geeft aan enige ervaring te hebben met het hybride framework Ionic, dit in combinatie met Angular. Hierbij was de uitdaging voornamelijk zijn gebrek aan kennis over Angular en het om de componenten van Ionic heen bouwen. Hij heeft nog geen ervaring met Ionic Vue, aangezien dit nog in geen enkel CODE14 project gebruikt is.

***No-/Low code platformen***

Christian geeft aan nog weinig ervaring te hebben met no- of low code platformen, maar heeft wel eens een deskresearch uitgevoerd naar dit soort platformen. Aangezien de productontwerpen voornamelijk in Figma (Figma, z.d.) gemaakt worden, heeft hij wel eens gezocht naar plugins voor Figma welke Vue code uit zouden spuwen. Net zoals Leon is Christian wel bang dat de code welke zo'n oplossing zou aanrijken, onbruikbaar zou zijn, gezien een design relatief complex kan worden en er veel verschillende manieren zijn om iets in het ontwerp aan te vliegen.

***Criteria welke voortvloeit uit dit interview***

- Zo dicht mogelijk bij de huidige stack, liefst Vue gerelateerd.
- Aansluiting op kennis van meerdere medewerkers.
- UI Componenten zijn zeker geen vereiste.
- Mogelijkheden tot de 'standaard' native functies zoals: Camera, LocalStorage, Bluetooth en mogelijk NFC? Etc.
- Verwisseling mogelijk tussen hybride- en web ontwikkeling.

### 3.2.2 Medewerkers enquête

Er is een enquête opgesteld om de kennisgebieden van de medewerkers van CODE14 in kaart te brengen. Deze gegevens zijn van groot belang voor het inkaderen van de beschikbare ontwikkeltechnieken. De enquête is uitgezet bij 27 programmeurs, waarvan er 22 hebben gereageerd.

Om een zo goed mogelijk beeld te krijgen van het belang ten aanzien van hybride frameworks, zijn er zowel open- als gesloten vragen gesteld in de enquête. In ***Bijlage D*** is een toelichting opgenomen over de relevantie van de vragen. Alle respondenten zijn developer of programmeur bij CODE14.

#### 3.2.2.1 Criteria welke voortvloeit uit de enquête

*In **Bijlage D** zijn alle gestelde vragen en bijbehorende antwoorden te vinden.*

**Criteria 1: Zo dicht mogelijk bij de huidige technologiestack, dus kennis blijven (VueJS, NuxtJS).**

**Toelichting**

Uit de antwoorden van de medewerkers van CODE14 is te halen dat ze de frameworks zoals NuxtJS en VueJS erg prettig vinden. In totaal zegt 90,9% van de respondenten tevreden te zijn met de huidige front-end technologiestack. Daarnaast weten 5 van de respondenten ook de combinatie mogelijkheden van dit soort frameworks i.c.m. hybride framework zoals Ionic, NativeScript en Capacitor te benoemen. Verder is het duidelijk dat de meeste kennis intern bij de web-ontwikkelingstechnieken ligt en dit is ook naar voren gekomen in de enquête 12 van de 17 respondenten (70,6%) heeft wel eens met web-ontwikkelingstechnieken een mobiele applicatie gerealiseerd. Daarnaast is het ook duidelijk dat 81,8% van de respondenten wil dat een hybride ontwikkeltechniek aansluit op de huidige kennis, in totaal vonden 17 mensen (77.3%) dat interne kennis van de gebruikte technieken van belang is.

**Criteria 2: Het terugvinden van UI (User Interface) componenten is geen vereiste binnen een hybride framework.**

**Toelichting**

In totaal vinden 13 respondenten (65%) het geen vereiste dat er binnen een hybride ontwikkeltechniek al uitgewerkte UI componenten beschikbaar zijn. Dit wordt in de Interviews met Leon en Christian (**Bijlage A, Bijlage C**) ook verder toegelicht. CODE14 maakt over het algemeen altijd product ontwerpen op maat en houdt hierbij weinig rekening met de native stijlgids van een platform. De 5 respondenten in de enquête welke dit wel van belang vonden (25%) zijn over het algemeen backend ontwikkelaars welke

weinig te maken hebben met de ontwikkeling van mobiele- of hybride applicaties. Verder zijn er nog 2 respondenten (10%) welke aangeven dat ze het liever op een andere manier zien, één hiervan geeft aan het liefste herbruikbare componenten te zien.

**Criteria 3: De mogelijkheid om ontwikkelde herbruikbare componenten te ontwikkelen zou van toegevoegde waarde kunnen zijn voor efficiënter ontwikkelen.**

**Toelichting**

In de open vragen wordt er vaak aangegeven dat herbruikbare componenten van toepassing zouden kunnen zijn, zo kan er intern een soort bibliotheek opgebouwd worden met componenten welke vaak terugkomen in de ontwerpen van de designers. Zodat dit soort componenten makkelijk hergebruikt kunnen worden in verschillende projecten.

**Criteria 4: HTML, CSS en JavaScript is een pré.**

**Toelichting**

Aangezien de wens is om zo dicht mogelijk bij de huidige technologiestack te blijven en in de enquête aangegeven wordt dat iedere respondent wel kennis heeft van HTML, CSS & JavaScript (één respondent niet), zou het een voordeel zijn, wanneer een toekomstig hybride framework gebruik maakt van dit soort talen. Daarnaast sluit dit heel goed aan op de front-end stack, aangezien NuxtJS nu de standaard is voor het ontwikkelen van een webapplicatie en deze maakt ook gebruik van HTML, CSS & JavaScript.

### 3.2.3 Gesprekken op de werkvloer

Omdat er veel ontwikkelaars werken bij CODE14, is er ook mondeling contact gezocht buiten interviews en de enquête om. Hier zijn discussies ontstaan over ervaringen uit het verleden, over tevredenheid van de gemaakte keuzes, etc. Tijdens deze gesprekken komen ook een aantal belangrijke punten naar voren welke meegenomen kunnen worden in de afbakening van een hybride framework.

- **Toekomstgericht: Een ouder hybride project updaten, hoeveel tijd kost dit?**

**Toelichting**

In een gesprek met Harm-Jan Hazelhorst (bedrijfsbegeleider, mede-eigenaar en softwareontwikkelaar) is naar voren gekomen dat hij een Native Android applicatie welke ontwikkeld is in 2016, zonder al te veel moeite weer draaiend heeft weten te krijgen zonder dat er in de tussentijd onderhoud aan is verricht. Hij gaf hierbij aan dat je dit niet zo snel zal ervaren bij hybride applicaties en dat hier in het verleden ook al problemen mee zijn geweest. Er zal onderzocht moeten worden hoe onderhoudsvriendelijk een hybride framework is.

- **TypeScript gebruiken in front-end projecten, dus ook hybride applicatie projecten.**

Het voornaamste verschil tussen JavaScript en TypeScript is dat, variabelen een typehint kunnen krijgen. Zo wordt er voordat er daadwerkelijk waarde aan de variabele gekoppeld wordt, al aangegeven dat de variabele bijvoorbeeld alleen nummers of een stukje tekst kan vasthouden. Vooral wanneer er invoer van een gebruiker verwacht wordt, kunnen er hierdoor foutmeldingen voorkomen worden.

- **Meenemen van NuxtJS + Capacitor in het onderzoek**

In de interviews, enquête en gesprekken op de werkvloer is NuxtJS vaak naar voren gekomen. Er is expliciet gevraagd om deze combinatie mee te nemen in het onderzoek gezien over dit framework de meeste interne kennis is, daarnaast zijn meerdere medewerkers ervan overtuigd dat Capacitor alle benodigdheden al aanbiedt wanneer het aankomt op native functies en toekomstgerichtheid. Om deze redenen zal deze combinatie betrokken worden bij de afbakening van de gevonden frameworks.

**Deelvraag: Wat zijn de mogelijkheden van hybride ontwikkeling t.o.v. native ontwikkeling?**

Tegenwoordig is de ontwikkeling van cross platform applicaties binnen één codebase al in een vergevorderd stadium. Het is kosten besparend wanneer er maar in één codebase gewerkt hoeft te worden. Daarnaast is er toegang tot een groot deel native functies zoals camera, bluetooth, etc. Toch zijn er een aantal dingen waardoor de keuze op native ontwikkeling kan vallen, zoals AR (Augmented Reality) of VR (Virtual Reality). Ook krijgen de nieuwste telefoons tegenwoordig snellere schermen (90-, 120- of 144Hz) schermen, waardoor native applicaties vloeiender weergegeven worden op dit soort smartphones, in tegenstelling tot hybride applicaties welke tot op heden tot 60Hz kunnen draaien. Toch is er door dit soort aspecten een duidelijke scheidingslijn te zien, wanneer er voor native of voor hybride gekozen zou moeten worden.

Voordat er een enquête wordt opgesteld wordt er gekeken naar relevante hybride ontwikkel frameworks. Dit gebeurt omdat er dan voorkennis is en hiermee er dieper kan worden ingegaan op de gestelde vragen. Voor de afbakening van de te onderzoeken ontwikkeltechnieken zijn er interviews afgenomen. Daarnaast is er een enquête opgesteld welke inzicht moet geven in de voorkeuren en kennisgebieden van de programmerende werknemers van CODE14. De data welke hieruit voortvloeide is gebruikt om criteria op te stellen voor de uiteindelijke afbakening van de ontwikkeltechnieken en frameworks.

### 3.2.4 Alternatieve ontwikkelmethoden

#### 3.2.4.1 Low-/No code

Volgens Leon Kosters (Lead Product Designer CODE14) en Christian Lemmers (Senior Product Designer) hebben de alternatieve ontwikkelmethoden de toekomst. Zij verwachten beiden dat in een tijdsbestek van 5 jaar het overgrote deel van de mobiele applicaties welke in productie zijn gezet, gerealiseerd zijn met behulp van of door een low- of no code platform. Immers geeft Leon in zijn interviews duidelijk aan dat de volledige overgang naar dit soort platformen naar zijn verwachting niet mogelijk is, omdat deze simpelweg nog niet volwassen genoeg zijn. Op basis van recent uitgewerkte prototypes in platformen zoals WebFlow, Framer en verschillende low-code plugins, geeft hij aan dat er te veel beperkingen zijn in het volledige gebruik van de technieken.

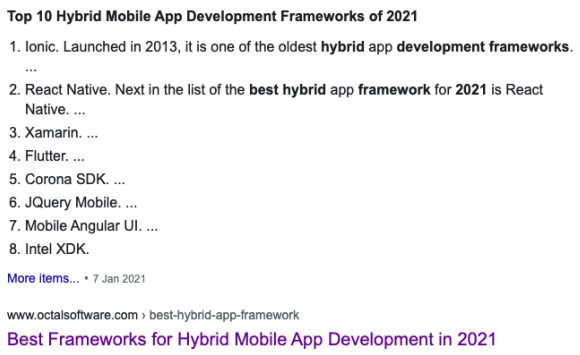
### 3.2.5 Frameworks en ontwikkeltechnieken

**Ophalen van informatie**

Om meer te weten te komen over hybride ontwikkelframeworks en ontwikkeltechnieken, is er gericht gezocht op Google. Hierbij zijn er hybride frameworks gevonden welke volgens de auteurs van de artikelen het beste naar voren komen. Om de verkregen informatie te valideren zijn er in totaal een drietal artikelen doorgenomen en zijn er zowel voor- als nadelen van de genoemde frameworks met elkaar vergeleken. Vervolgens is er gekeken naar de relevantie van een framework binnen CODE14 en is er een lijst met frameworks opgesteld om verder mee te nemen in het onderzoek.

**Beste hybride ontwikkeling frameworks in 2021 volgens (A. Goyal, 2021)**

Om een goed beeld te krijgen van welke ontwikkeltechnieken er hedendaags gebruikt worden, is er gericht gezocht op Google naar *"Best Hybrid Development Frameworks in 2021"*. Een zoektocht in Google heeft geresulteerd in de lijst welke te zien is in Figuur 5. In het artikel van A. Goyal (Goyal, 2021) is een lijst met frameworks opgesteld.




*Figuur 5 - Zoekresultaat Top 10 Hybrid mobile app development frameworks*

**Over Arun Goyal en Octal Software**

Arun Goyal is de Managing Directeur van Octal IT Solutions (Offshore IT Services | IT Software Solutions Company - Octal IT Solution, 2021) een bedrijf welke web, software en mobiele applicaties ontwikkeld voor meer dan 70 landen over heel de wereld. Mede door 14 jaar ervaring in de sector, hun expertise op mobiele applicatieontwikkeling en een indrukwekkend klantbestand met klanten zoals IBM, BMW en Vodafone, kan dit artikel als betrouwenswaardig beschouwd worden.

**Belangrijke punten welke benoemd zijn door A. Goyal (2021)**

| | |
|---|---|
| Voordelen van **React Native** volgens **A. Goyal (2021)**<br>- Hybride applicaties met hoge prestaties<br>- Plugins van externe partijen is mogelijk.<br>- Kostenbesparend t.o.v. andere hybride frameworks. | Nadelen van **React Native** volgens **A. Goyal (2021)**<br>- Amateuristische community voor ontwikkelaars<br>- Heeft een aantal compatibiliteitsproblemen in het eindproduct |
| Voordelen van **Ionic** volgens **A. Goyal (2021)**<br>- Voor gedefinieerde UI elementen<br>- Veel hulp vanuit de community<br>- Uitgebreide documentatie<br>- Codebase voor meerdere platformen | Nadelen van **Ionic** volgens **A. Goyal (2021)**<br>- Te veel afhankelijk van plugins<br>- Het gebruiken van meerdere functionaliteiten heeft impact op de prestaties |
| Voordelen van **Xamarin** volgens **A. Goyal (2021)**<br>- Herbruikbare code<br>- Compleet ecosystem<br>- Naadloze integratie met externe hardware<br>- Prestatie is op een hoog niveau | Nadelen van **Xamarin** volgens **A. Goyal (2021)**<br>- Duurder dan concurrerende frameworks<br>- Relatief kleine community<br>- Gelimiteerd aan specifieke technologieën. |
| Voordelen van **Flutter** volgens **A. Goyal (2021)**<br>- Geweldige cross-platform mogelijkheden<br>- Snelle ontwikkeling en betrouwbare prestaties<br>- Interactief en uitgebreid UI ontwerp en ontwikkeling<br>- Google's support en afhankelijkheid | Nadelen van **Flutter** volgens **A. Goyal (2021)**<br>- Community van ontwikkelaars is gelimiteerd tot Google en Alibaba ontwikkelaars<br>- Apps zijn vaak groter dan wanneer dit native ontwikkeld zou zijn. (Opslag)<br>- Relatief nieuw en heeft tijd nodig om volwassener te worden |
| Voordelen van **CoronaSDK** volgens **A. Goyal (2021)**<br>- Snelle applicatieontwikkeling<br>- Uitgerust met veel functionaliteiten<br>- Levert goed presterende applicaties | Nadelen van **CoronaSDK** volgens **A. Goyal (2021)**<br>- Gelimiteerde support voor extern plugin gebruik<br>- LUA kan saai zijn om mee te werken voor nieuwe ontwikkelaars |
| Voordelen van **jQuery Mobile** volgens **A. Goyal (2021)**<br>- Simpele navigatie<br>- Intuïtieve webpagina's<br>- Volwassen framework<br>- Indrukwekkende plugin library | Nadelen van **jQuery Mobile** volgens **A. Goyal (2021)**<br>- Relatief trage applicaties<br>- Ervaren designer en ontwikkelaar benodigd<br>- Complexe applicatieontwikkeling |

| | |
|---|---|
| | - Eigen ontwerp kan problemen met CSS veroorzaken |
| Voordelen van **Mobile Angular UI** volgens **A. Goyal (2021)**<br>- Gratis in gebruik<br>- HTML5<br>- Ideaal voor lichte applicaties | Nadelen van **Mobile Angular UI** volgens **A. Goyal (2021)**<br>- Gelimiteerd tot weinig functies<br>- Niet geschikt voor complexe applicaties<br>- Afhankelijk van Bootstrap |
| Voordelen van **Intel XDK** volgens **A. Goyal (2021)**<br>- Wordt constant doorontwikkeld<br>- Waardevolle synchronisatie methoden<br>- Ingebouwde tooling voor live previews | Nadelen van **Intel XDK** volgens **A. Goyal (2021)**<br>- Zwaar systeem |
| Voordelen van **Appcelerator** volgens **A. Goyal (2021)**<br>- Op web gebaseerde SDK<br>- Constant groeiende en behulpzame community<br>- Veel API-functies om te gebruiken | Nadelen van **Appcelerator** volgens **A. Goyal (2021)**<br>- Weinig flexibiliteit<br>- Complexe applicaties zijn lastig te ontwikkelen<br>- Kan overwegend duur zijn om in te ontwikkelen |
| Voordelen van **Framework 7** volgens **A. Goyal (2021)**<br>- Levert snelle applicaties<br>- Goede hulp vanuit community<br>- Applicaties zijn volledig licentie vrij | Nadelen van **Framework 7** volgens **A. Goyal (2021)**<br>- Relatief weinig documentatie<br>- Integratie met andere hybride frameworks is niet mogelijk<br>- Werkt alleen op laatste versies van besturingssystemen |

*Tabel 1 - Bevindingen A. Goyal (2021) over hybride ontwikkelframeworks*

**Beste hybride ontwikkeling frameworks in 2021 volgens (Aparna, 2021)**

Om steekproefsgewijs te kijken of deze lijst daadwerkelijk klopt, is er gekeken naar een andere websites die welke naar voren kwam bij de zoekterm *"Top 10 Hybrid Development Frameworks in 2021"*. Op deze wijze kan er gecontroleerd worden of de eerste lijst welke weergegeven werd op Google, daadwerkelijk up-to-date is. Hierbij kwam een artikel van (Aparna, 2021) naar voren. De lijst welke hieruit is gekomen is te zien in Figuur 2.

**Over Aparna en MobileAppDaily.com**

"Aparna is een groeispecialist met praktische kennis in bedrijfsontwikkeling. Ze waardeert marketing als een belangrijke drijfveer voor de verkoop en houdt gelijke tred met het laatste nieuws in de mobiele app-industrie." – (Aparna, 2021)

“MobileAppDaily is een nieuwsportaal waarin mobiele applicaties centraal staan en die de nieuwste doorbraken in de mobiele app-industrie bespreekt. We zorgen voor een snelle levering van elementaire tot zeer belangrijke vorderingen of veranderingen in de wereld van de ontwikkeling van mobiele apps met onze grondige analyse en inzichten.” - (Crunchbase, z.d.)

Door het specifieke onderwerp ‘Mobiele Applicaties’ waar het platform MobileAppDaily zich mee bezig houdt en de expertise van de schrijfster, zullen de kernpunten uit dit artikel meegenomen worden in de opstelling van de lijst met de meest relevante hybride ontwikkelframeworks.

**Belangrijke punten welke benoemd zijn door Aparna (2021)**

| | |
|---|---|
| Voordelen van **Ionic** volgens **Aparna (2021)**<br>- Applicaties zijn eenvoudig in elkaar gezet en gemakkelijk te updaten<br>- Heeft native functionaliteiten<br>- Enorm veel UI componenten<br>- Voor gegenereerde applicatie setups<br>- Extreem populair | Nadelen van **Ionic** volgens **Aparna (2021)**<br>- Geen hot reloading (Ontwikkeltechniek afhankelijk)<br>- Te afhankelijk van plugins<br>- Niet het beste voor zware applicaties |
| Voordelen van **React Native** volgens **Aparna (2021)**<br>- Maximaal code hergebruik<br>- Native rendering van code<br>- NodeJS support<br>- Live reloading<br>- Veel plugins | Nadelen van **React Native** volgens **Aparna (2021)**<br>- Niet het beste voor applicaties met meerdere schermen, UI Transities, Zware speciale effecten en veel interacties<br>- Moet verbeteren in toegang tot hardwarecomponenten<br>- Tekortkomingen in de vorm van navigatie componenten<br>- Te weinig modules op maat |
| Voordelen van **Flutter** volgens **Aparna (2021)**<br>- Top keuze voor cross-platform development<br>- Enorm snel<br>- Hot reloading<br>- Mooie UI designs<br>- Dynamisch schrijven van code<br>- Eigen widgets | Nadelen van **Flutter** volgens **Aparna (2021)**<br>- Erg nieuw, niet volwassen<br>- Applicaties gemaakt met Flutter zijn gemiddeld 40% groter dan wanneer dit in native zou gebeuren<br>- Ontwikkelaars moeten Dart leren voor ze in Flutter aan de slag kunnen<br>- Sommige functies hebben weinig ondersteuning |
| Voordelen van **PhoneGap** volgens **Aparna (2021)** | Nadelen van **PhoneGap** volgens **Aparna (2021)** |

| | |
|---|---|
| - Simpel in gebruik<br>- Single codebase voor iOS, Android en Windows<br>- Extreem flexibel<br>- HTML5, CSS3 en JavaScript<br>- Native functies | - Levert trage applicaties<br>- Populair bij kleinere bedrijven<br>- Biedt geen ondersteuning voor alle native functies<br>- Weinig UI elementen out-of-the-box |
| Voordelen van **Xamarin** volgens **Aparna (2021)**<br>- Native UI functies<br>- Herbruikbare code<br>- Eigen ecosysteem<br>- Gemakkelijke API-integraties<br>- Snel prototypes ontwikkelen | Nadelen van **Xamarin** volgens **Aparna (2021)**<br>- Duur in vergelijking met concurrerende frameworks<br>- Veel overhead code<br>- Platform specifieke limitaties in Android en iOS<br>- Weinig ontwikkelaars weten hoe ze Xamarin moeten gebruiken. (hoge learning curve)<br>- Weinig API-support |
| Voordelen van **NativeScript** volgens **Aparna (2021)**<br>- Snel in gebruik<br>- VueJS support<br>- Native code compilatie | Nadelen van **NativeScript** volgens **Aparna (2021)**<br>- Slechte documentatie<br>- Misleidende omschrijving van de ontwikkelomgeving<br>- Slechte codevoorbeelden |
| Voordelen van **KendoUI** volgens **Aparna (2021)**<br>- Goede UI componenten ter beschikking<br>- High performance i.c.m. goede native runtime | Nadelen van **KendoUI** volgens **Aparna (2021)**<br>- Weinig documentatie<br>- Learning curve<br>- Weinig ondersteuning |
| Voordelen van **Framework 7** volgens **Aparna (2021)**<br>- Eenvoudig te leren<br>- Veel widgets en UI elementen<br>- Ingebouwde hulp functies | Voordelen van **Framework 7** volgens **Aparna (2021)**<br>- Only supports iOS and Android (Afhankelijk van native runtime)<br>- Weinig online community support<br>- Gemiddelde documentatie |
| Voordelen van **Aurelia** volgens **Aparna (2021)**<br>- Simpel testbaar<br>- Makkelijk in ontwikkeling | Nadelen van **Aurelia** volgens **Aparna (2021)**<br>- Learning curve<br>- Gemiddelde documentatie |

| | |
|---|---|
| - Hoge prestaties<br>- React binding | |
| Voordelen van **Onsen UI** volgens **Aparna (2021)**<br>- Makkelijk in gebruik<br>- Snel en efficiënt<br>- Prestaties geoptimaliseerd voor verschillende devices | Nadelen van **Onsen UI** volgens **Aparna (2021)**<br>- Slechte documentatie<br>- Weinig ondersteuning |

*Tabel 2 - Bevindingen Aparna (2021) hybride ontwikkelframeworks*

**Beste hybride ontwikkeling frameworks in 2021 volgens (K. Shah, 2021)**

Voor de laatste lijst met frameworks is de volgende zoekterm gebruikt: “Cross Platform Application Development frameworks 2021”. Het meest relevante artikel is van (Shah, 2021), omdat dit het meest recent is gepubliceerd.

**Over K. Shah en ThirdRockTechKno**

“Krunal Shah heeft meer dan 10 jaar ervaring in de IT-industrie en heeft gewerkt aan verschillende complexe projecten op het gebied van sociale ondernemingen, de auto-industrie en het onderwijs. Krunal is een expert in databaseontwerp, architectuurontwerp, prestatieoptimalisatie, caching en ontwerp, naast andere complexe systemen. Krunal gelooft in hoge standaarden en processen in zijn werk en gedijt op klanttevredenheid.” - (ThirdRockTechKno, z.d.)

Krunal Shah heeft een indrukwekkend aantal artikelen over onderwerpen binnen de software en IT-sector op zijn naam staan en is mede-eigenaar van ThirdRockTechKno (ThirdRockTechKno, z.d.), dit is een bedrijf met meerdere vestigingen in India en Amerika. Hun expertise ligt op het gebied van web- en mobiele ontwikkeltechnieken.

**Belangrijke punten welke benoemd zijn door K. Shah (2021)**
In tegenstelling tot de andere artikelen beschrijft K. Shah geen nadelen van de genoemde ontwikkeltechnieken en frameworks.

| |
|---|
| Voordelen van **Flutter** volgens **K. Shah (2021)**<br>- Ingebouwde grafische engine, daarom geen problemen met verschillende interfaces voor platformen zoals Android en iOS.<br>- Levert snelle cross-platform applicaties<br>- Flutter maakt gebruik van de GPU (Graphical Processing Unit), wat op gebied van prestaties voordelen met zich meebrengt. |
| Voordelen van **React Native** volgens **K. Shah (2021)**<br>- Focus op UI<br>- Open source<br>- Goede support van Facebook<br>- Directe verbinding met native API's |
| Voordelen van **Ionic** volgens **K. Shah (2021)**<br>- UI Elementen direct klaar voor gebruik<br>- Enorm veel plugins beschikbaar<br>- HTML, CSS, JavaScript<br>- Open source<br>- Veel supported platforms (Afhankelijk van Native runtime) |
| Voordelen van **NodeJS** volgens **K. Shah (2021)**<br><br>**NodeJS is de basis voor een groot aantal van de andere benoemde frameworks**<br>- Grote community<br>- Open source<br>- Eén van de snelste ontwikkelframeworks<br>- NodeJS applicaties verkorten de responstijd voor requests |
| Voordelen van **Xamarin** volgens **K. Shah (2021)**<br>- Xamarin is gemakkelijk in gebruik te nemen, ontwikkelaars hoeven alleen kennis te hebben van .NET en C# om het te gebruiken.<br>- Xamarin heeft een grote community met ontwikkelaars |
| Voordelen van **NativeScript** volgens **K. Shah (2021)**<br>- Javascript, HTML en CSS<br>- Mooie UI elementen voor native platformen<br>- Veel plugins, waardoor er geen externe plugins benodigd zijn |
| Voordelen van **CoronaSDK** volgens **K. Shah (2021)**<br>- Veel plugins voor media, analyses, in-app adverteren etc<br>- Lua programmeertaal, waardoor het framework snel is |

| |
|---|
| - Ontwikkelaars kunnen de veranderingen in code real-time zien |
| Voordelen van **Appcelerator** volgens **K. Shah (2021)**<br><br>- CI/CD integraties<br>- Enorm veel tooling voor versnelling van het ontwikkelproces |
| Voordelen van **Sencha Touch** volgens **K. Shah (2021)**<br><br>- Meer dan 50 ingebouwde aanpasbare UI elementen<br>- Cordova integratie, makkelijk om native API's te benaderen |
| Voordelen van **PhoneGap** volgens **K. Shah (2021)**<br><br>- Betalingssystemen te benaderen voor Play- en Appstore<br>- PhoneGap biedt de mogelijkheid om andere frameworks erbij te gebruiken |

*Tabel 3 - Bevindingen K. Shah (2021) hybride ontwikkelframeworks*

#### 3.2.1.1 Bruikbaarheid en toepasbaarheid gevonden frameworks

Niet alle frameworks welke benoemd zijn in de drie artikelen zijn bruikbaar. Neem hierbij CoronaSDK als voorbeeld, het gaat hierbij om een cross-platform framework welke zich richt op de 2D game markt. Om deze reden is er gekozen om kort een blik te werpen op welke frameworks bruikbaar zijn voor de gemiddelde cross-platform applicatie welke CODE14 zou ontwikkelen, dit zijn voornamelijk informatieve en functionele applicaties aansluitend op een dienst of een tastbaar product, bijvoorbeeld de AB Connect applicatie, dit is een applicatie welke chauffeurs informeert over de laatste ontwikkelingen binnen het bedrijf en ze de mogelijkheid biedt om status updates te geven m.b.t. hun opdrachten. Ook wordt hierbij gekeken naar algemene bekendheid, volwassenheid en de toegevoegde waarde van een framework.

| Framework | Toelichting |
|---|---|
| React Native | - In alle drie de artikelen benoemd.<br>- Ondersteuning voor iOS en Android<br>- Grote community<br>- Ondersteuning door grote organisatie (Facebook)<br>- Geen WebView, dus native ervaring en daarmee betere prestaties |
| Flutter | - In alle drie de artikelen benoemd.<br>- Ondersteuning voor iOS en Android en experimenteel met web.<br>- Grote community<br>- Ondersteuning van grote organisatie (Google) |
| Ionic | - In alle drie de artikelen benoemd.<br>- HTML, CSS en Javascript<br>- Grote community<br>- Ervaring met Ionic intern<br>- Ondersteund door organisatie (Ionic)<br>- Grote community<br>- VueJS support |
| Xamarin | - In alle drie de artikelen benoemd<br>- Snel prototypes ontwikkelen<br>- Herbruikbare code<br>- Hoge prestaties |
| NativeScript | - In twee van de drie artikelen benoemd.<br>- HTML, CSS en JavaScript<br>- Compileert naar native code<br>- Veel bruikbare plugins<br>- VueJS Support |
| Framework 7 | - In twee van de drie artikelen benoemd.<br>- VueJS Support<br>- HTML CSS JavaScript<br>- Veel UI elementen<br>- Eenvoudig te leren |
| Appcelerator | - CI/CD Tooling<br>- Versneld ontwikkelproces<br>- Groeiende community |
| NuxtJS + Capacitor | - HTML, CSS en JavaScript.<br>- Zelfde native runtime als Ionic<br>- Veel interne kennis<br>- Statisch renderen van web-pagina's<br>- Android, iOS en websupport. |

*Tabel 4 - Toelichting relevente frameworks*

**Relevante frameworks**
In de artikelen waar de lijsten met frameworks naar voren zijn gekomen, worden al een aantal belangrijke en unieke aspecten benoemd, per framework. In Figuur 2, in het onderste blauwe gedeelte is te zien welke frameworks betrokken zullen worden in het verdere onderzoek. Op basis van eerste indruk van de website, documentatie en focus van deze frameworks kwam naar voren dat deze aan zouden kunnen sluiten op de stack van CODE14. Ieder van deze frameworks zullen meegenomen worden naar de eerste filter fase, waarbij er gemeten wordt aan de criteria van de medewerkers van CODE14.

**Open visie**
Tijdens de zoektocht werden frameworks zoals React Native, Ionic, Flutter en NativeScript vaak genoemd. Immers sluiten deze niet aan op alle benoemde wensen welke benoemd zijn in de enquête en interviews. Om te ondervinden of deze frameworks op andere aspecten een goede aansluiting bieden, worden deze ook gezien als relevante frameworks en daarom meegenomen naar de volgende fase.

**PhoneGap uitgesloten, waarom?**
PhoneGap is tweemaal genoemd in de lijsten met frameworks, immers staat op de website van PhoneGap het volgende**: "**Adobe has discontinued PhoneGap Build and ended investment in PhoneGap and Apache Cordova. This website is no longer being updated." (*Adobe PhoneGap Build*, 2013). Daarnaast is er in onderzoek naar de andere benoemde frameworks, in een artikel van (Ionic, Capacitor, z.d.) gezegd aangegeven dat Capacitor de opvolger is van Cordova en PhoneGap.

## 3.3 Fase 2 - Criteria vanuit CODE14

**Deelvraag: Wat zijn de eisen waar een hybride ontwikkeltechniek aan moet voldoen voor de gemiddelde CODE14 hybride applicatie?**

In Tabel 5 zijn een zestal criterium te vinden welke voort zijn gekomen uit de enquête en de afgenomen interviews. Criteria C1, C2, C3 en C5 zullen meegenomen worden in de eerste afbakening, aangezien dit verplichte criteria is vanuit CODE14. Dit is gedaan volgens de MoSCoW (Must, Should, Could, Would) methode, waarbij de should, could of would criteria als doorslaggevende factor kan fungeren binnen de tweede fase van de afbakening, bij de beoordeling d.m.v. een rubriek.

| ID | Criteria | Toelichting | Invloed | MoSCoW |
|---|---|---|---|---|
| **C1** | Toepasbaar op huidige technologiestack en interne kennis | De toepasbaarheid van een ontwikkeltechniek of framework moet aansluiting hebben op de huidige kennis van de medewerkers van CODE14 en moet een goede aansluiting hebben op de bestaande technologiestack (VueJS of Nuxt)<br><br>HTML, (S)CSS & JavaScript zijn hierbij een pré | **Hoog** | Must |
| **C2** | Toegang tot minimaal de volgende native API-functies: **LocalStorage, Camera, Bluetooth, NFC en Galerij, QR-/Barcode scanner en push notificaties.** | In interviews en gesprekken met medewerkers kwam naar voren dat de door CODE14 gerealiseerde applicaties in het verleden gebruik maakten van de beschreven functionaliteiten. Hierbij wordt aangegeven dat doorontwikkeling van het framework welke toegang biedt tot de functionaliteiten een pré is. | **Gemiddeld** | Must |
| **C3** | Eigen ontwerp UI moet toepasbaar zijn, zonder enige blokkades. | Uit de enquête en interviews is naar voren gekomen dat een hybride ontwikkel framework geen UI componenten aan hoeft te bieden. Mocht dit een gekozen | **Gemiddeld** | Must |

| | | | | |
|---|---|---|---|---|
| | | techniek dit wel aanbieden, moet dit zeker geen blokkade vormen voor eigen ontworpen UI componenten. | | |
| **C4** | Installatie & Development proces. | Een nieuw project moet gemakkelijk op te zetten zijn, ook moet het eenvoudig weer draaiend te krijgen zijn, wanneer het project een tijd stil heeft gelegen. | **Gemiddeld** | Should |
| **C5** | Er kan in het hybride ontwikkel framework zowel Android, iOS als webapplicaties worden gerealiseerd. | Uit gesprekken met medewerkers van CODE14 is voortgekomen dat het een grote pré is als een hybride ontwikkeltechniek zowel Android-, iOS- als webapplicaties ondersteund en kan builden. | **Gemiddeld** | Should (but high pré) |
| **C6** | Mogelijkheid om de overgang te maken naar TypeScript. | In de enquête is naar voren gekomen dat het een pré als een hybride ontwikkeltechniek de mogelijkheid heeft om TypeScript te gebruiken. Daarnaast is er vanuit het Development Board van CODE14 aangegeven dat er gezocht wordt naar een manier om TypeScript organisatie breed te integreren, immers valt dit buiten dit onderzoek. | **Laag** | Could |

*Tabel 5 - Criteria vanuit CODE14*

### 3.3.1 Eerste afbakening

Op basis van de bovenstaande criteria in Tabel 5, zijn de gevonden frameworks gefilterd. In **Bijlage E** worden een viertal criteria *(C1, C2, C3 & C5)* uit Tabel 5 meegenomen. Verder zijn er nog een aantal voor- en nadelen beschreven welke de frameworks met zich mee brengen. In Tabel 6 is te zien welke frameworks wel of niet voldoen aan de benoemde criteria.

#### 3.2.5.1 Overgebleven frameworks

✓ *Voldoet aan de criteria.*
✗ *Voldoet deels aan de criteria of heeft verhinderingen.*
✗ *Voldoet niet aan de criteria.*

| **Framework** | **Criteria C1** Technologiestack | **Criteria C2** Native Functies | **Criteria C3** Customizability | **Criteria C5** Platformen |
|---|---|---|---|---|
| React Native | ✗ | ✓ | ✓ | ✓ |
| **Ionic Vue Capacitor** | ✓ | ✓ | ✓ | ✓ |
| Flutter | ✗ | ✓ | ✓ | ✓ |
| Xamarin | ✗ | ✓ | ✓ | ✓ |
| NativeScript i.c.m. Vue | ✗ | ✓ | ✗ | ✗ |
| **Framework 7 i.c.m. Vue & Capacitor** | ✓ | ✓ | ✓ | ✓ |
| CoronaSDK | ✗ | ✗ | ✗ | ✗ |
| Appcelerator - Titanium | ✗ | ✗ | ✓ | ✓ |
| **NuxtJS** | ✓ | ✓ | ✓ | ✓ |

*Tabel 6 - Afbakening frameworks o.b.v. criteria*

**Native Runtime**

De overgebleven drie combinaties (Ionic, NuxtJS en Framework7) bevatten *Capacitor* als native runtime. Capacitor zorgt voor de mogelijkheid om op web gebaseerde technologieën om te zetten

in uitvoerbare mobiele applicaties in *.apk* of. *.ipa* vorm en heeft een rijkelijk aanbod ondersteunde platformen, zoals; Android, iOS, Web, PWA en Electron applicaties (denk hierbij aan grote applicaties op Desktop en MacOS zoals Slack, Twitch, Visual Studio Code, etc. (ElectronJS, z.d.)). Naast dit grote voordeel biedt deze native runtime ook de mogelijkheid om deze in je eigen front-end stack te integreren. Dit houdt in dat je niet gelimiteerd bent aan het huidige aanbod aan hybride ontwikkel frameworks. Dit is dan ook één van de redenen waarom NuxtJS betrokken is bij het onderzoek.

#### 3.3.1.2 Geschiedenis overgebleven frameworks

**Ionic**

Ionic is in 2013 opgericht door Drify Co. De eerste alpha versie van het framework kwam uit in November 2013, waarop pas in 2016 de tweede versie van het framework werd gepubliceerd, dit had meer opties qua UI (User Interface) elementen, maar er kon nog steeds alleen in AngularJS ontwikkeld worden. In 2019 werd Ionic 4 op de markt gebracht, welke de mogelijkheid aan zijn gebruikers gaf om PWA's (Progressive Web Applications) te ontwikkelingen en hierbij kwamen andere frameworks dan AngularJS ook in beta. (Javatpoint, z.d.)

In de tussentijd heeft Ionic ook een eigen ecosysteem ontwikkeld met daarin native runtime Capacitor en AppFlow als CI/CD tooling.

**Framework7**

De eerste versie van Framework7 kwam in 2019 op de markt. Framework7 heeft in de tussentijd in totaal vier nieuwe versies meegemaakt, waarbij er updates zijn geweest voor de UI en UX-aspecten om weer up-to-date te zijn met Android en iOS en de stijlgidsen die deze besturingssystemen hanteren. Er is naast Cordova ook support voor Capacitor gekomen, omdat Cordova heeft aangegeven niet meer door te ontwikkelen aan hun native runtime.

**NuxtJS**

De eerste publiekelijke release van NuxtJS was 5 jaar geleden, op 7 november 2016, waar ze versie V0.2.0 publiceerde. Sindsdien heeft NuxtJS grote stappen gezet, inmiddels zitten ze op stabiele versie 2.15.3, welke 10-03-2021 uitgebracht werd. In de tussentijd zijn er veel aspecten ingebouwd, zoals het statisch renderen van pagina's. NuxtJS werd gebouwd als alternatief voor VueJS, met de mogelijkheid om statische websites te kunnen renderen (Server Side Rendering), dit had grote impact op de snelheid, aangezien het overgrote deel van de front-end frameworks destijds alleen client-side rendering ondersteunde.

De overgebleven frameworks worden meegenomen in een beoordelingstabel waarin de onderstaande onderwerpen behandeld worden. Er zijn aspecten binnen de beoordelingstabel (Tabel 8) welke zwaarder mee tellen dan anderen, dit kan zijn omdat dit meer van belang is voor CODE14 of beter aansluit op de wensen van CODE14. Omdat in de kern de vier overgebleven frameworks zouden passen binnen CODE14 is een rubric een goede manier om uit te sluiten waarin deze frameworks uitblinken.

Er zullen per onderdeel cijfers worden gegeven aan de frameworks, waarin het laagste haalbare cijfer (**1**) slecht is en het hoogst haalbare cijfer (**10**) uitstekend is.

### 3.4.2 Belangrijkste onderwerpen

De genoemde onderwerpen in tabel 7, zijn naar voren gekomen op basis van verkregen informatie vanuit CODE14 in de enquête, interviews en gesprekken op de werkvloer. Vervolgens is er met Harm-jan Hazelhorst gekeken naar de weging van ieder onderwerp.

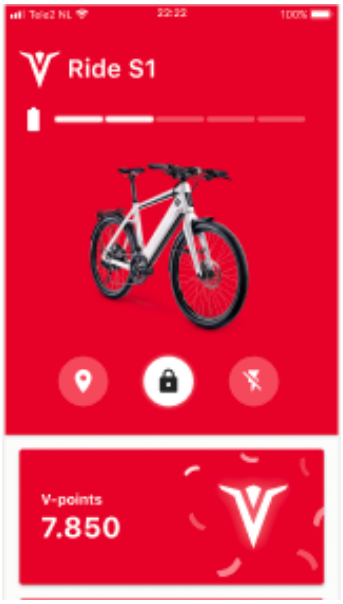
| ID | Onderwerp | Toelichting | Gewicht (%) |
|---|---|---|---|
| **R1** | **Installatie en development proces** | Een framework moet makkelijk op te zetten zijn, hierbij komen een aantal vragen kijken: Hoe is de CLI welke de native runtime of het framework met zich meebrengt? Wat zijn de ontwikkelmethoden, is er mogelijkheid tot live reloading tijdens het ontwikkelen? | **10%** |
| **R2** | **Community** | Het is van belang dat er een community is voor een ontwikkeltechniek, framework of native runtime. Het kan zijn dat er tijdens de ontwikkeling tegen zeer specifieke problemen aangelopen wordt. Als dit het geval is, dan kan de grote van de community van pas komen, vaak betekent dit dat er sneller antwoord is op vragen en is er een vergrote kans dat een ontwikkelaar ooit hetzelfde probleem heeft ervaren. Daarnaast duidt de grote van een community vaak op hoe toegankelijk een framework is, dit is natuurlijk niet hard te maken, maar blijft een aspect om rekening mee te houden. | **10%** |

| | | | |
|---|---|---|---|
| **R3** | **Ondersteunde platformen** | Welke platformen worden ondersteund door een framework? Hoe gaat een framework om met verschillende schermformaten? | **15%** |
| **R4** | **Documentatie** | Zijn de beschikbare functionaliteit goed gedocumenteerd? Hoe leesbaar is de documentatie? | **15%** |
| **R5** | **Prestatie** | Hoe goed presteert de applicatie op verschillende apparaten (oudere en nieuwere)? Hoeveel ruimte neemt een gecompileerde applicatie in beslag op een apparaat? | **10%** |
| **R6** | **Learning Curve, Toepasbaarheid en Toegankelijkheid** | Op basis van de interne kennis van CODE14, hoe makkelijk is het framework te leren? Wat zijn de voornaamste leerpunten voor de gemiddelde programmeur binnen CODE14? En sluit dit aan op de gewenste efficiëntie wat er van een nieuwe ontwikkeltechniek verwacht wordt? Sluiten de gebruikte ontwikkeltechnieken binnen een framework aan op programmeerparadigma's? Hoe toegankelijk is het framework voor junioren? Van welke programmeertalen is voorkennis vereist? | **15%** |
| **R7** | **Deployment** | Deployment is voor ieder platform anders, maar wordt het makkelijker gemaakt bij het framework of native runtime? Hoe wordt er een applicatie voor de verschillende platformen gebouwd en wat is hiervoor nodig? | **10%** |
| **R8** | **Plugins, Native Functies** | Tot welke native functies heeft het framework of native runtime toegang? Wordt er actief doorontwikkeld aan nieuwe technieken? | **15%** |

*Tabel 7 - Te beoordelen onderwerpen*

### 3.4.3 Prototypes

Omdat het ontwikkelen in de praktijk soms net iets anders gaat dan in theorie, is het beoordelen van framework puur o.b.v. literatuuronderzoek onvoldoende relevant. Hierom is de keuze gemaakt om een prototype te ontwikkelen in alle drie de framework combinaties. Dit om ook een goed beeld te krijgen van het gebruiksgemak, de daadwerkelijke learning curve en steekproeven op de beschikbare documentatie.




*Figuur 6 - Dashboard Ontwerp VYBER applicatie*

#### 3.4.3.1 Prototype, wat wordt er getest?

**Geteste platformen:** Android (Samsung Galaxy S20 Exynos Processor), Web, PWA.

Aangezien er op basis van dit onderzoek een hybride applicatie ontwikkeld wordt voor Vyber, zal er gekeken worden of er zonder problemen een scherm nagebouwd kan worden. Vervolgens is het de bedoeling dat er twee native functies aangeroepen worden d.m.v. Capacitor en deze vervolgens getest worden op minimaal twee platformen. De gekozen native functies zijn; *Device Info* en *Camera/Galerij*.

#### 3.4.3.2 Framework7 Vue + Capacitor

**Geteste platformen:** Android (Samsung Galaxy S20 Exynos), Web & PWA.

Voor het Framework7 prototype is er gekozen voor de combinatie met Vue en de native runtime Capacitor. Hierbij is er focus gelegd op de aanpasbaarheid van de UI elementen van Framework7. Het gebruik van *<f7-button>* leverde op basis van styling wel wat problemen op, om het volledig volgens design te krijgen moest er veel overschreven worden door middel van ::v-*deep* selectors, wat niet wenselijk is. Door het gebruik van de normale HTML-element button is er complete controle over de styling en is er ook geen probleem met de cross-platform functionaliteit.

#### 3.4.3.3 NuxtJS + Capacitor

**Geteste platformen:** Android (Samsung Galaxy S20 Exynos), Web, PWA.

Het is mogelijk om met eigen Vue componenten het design op te bouwen in Nuxt en daarmee voldoet het aan de gestelde eis van CODE14. Er worden geen UI elementen meegeleverd.

```
async takePicture() {
    this.cameraActive = !this.cameraActive;
    const image = await Camera.getPhoto( options: {
        quality: 90,
        allowEditing: true,
        resultType: CameraResultType.Uri
    });
    // image.webPath will contain a path that can be set as an image src.
    // You can access the original file using image.path, which can be
    // passed to the Filesystem API to read the raw data of the image,
    // if desired (or pass resultType: CameraResultType.Base64 to getPhoto)
    let imageUrl = image.webPath;
    // Can be set to the src of an image now
    let imageElement = {};
    imageElement.src = imageUrl;
    console.log(imageElement);

    this.images.push(imageElement);
    this.profilePicture = imageElement.src;
},
async getDeviceInfo() {
    this.deviceInfo = await Device.getInfo();
    this.batteryInfo = await Device.getBatteryInfo();
}
```

*Figuur 7 – Broncode voor aanroepen camera functie*

**PWA Functies**
De device info werd wel getoond maar het probleem in deze combinatie is het omzetten naar een Android native project en het aanroepen van de PWA-functies. Dit werkt niet optimaal. Daar waar Framework7 en Ionic beide de camera functie tonen, wordt deze in NuxtJS niet getoond, met dezelfde broncode er wordt hierbij geen foutmelding getoond in de console.

### 3.4.4 Eindbeoordeling

In **Bijlage F** is de toelichting te vinden over de cijfers welke gegeven zijn aan een framework combinatie. Sommige onderwerpen hadden meer de focus op de integratie van de native runtime (in elk geval Capacitor), hier is aparte toelichting voor gegeven indien nodig.

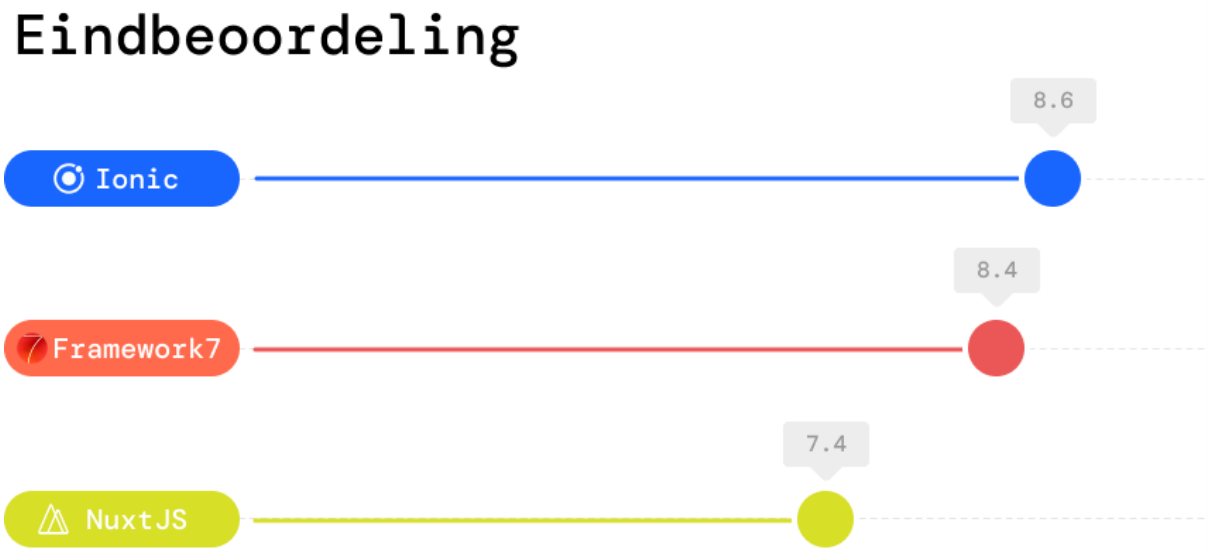
#### 3.4.4.1 Gewicht van criteria

In totaal tellen de acht criteria een gewicht van 100%. Criteria R3, R4, R5 en R6 maken hierbij ieder 15% deel uit van de eindbeoordeling. Dit komt doordat er vanuit de resultaten uit de enquête, gesprekken op de werkvloer en de verkregen antwoorden uit de interviews duidelijk naar voren kwamen dat dit belangrijkere criteria zijn, ten opzichte van de ander genoemde criteria.

| Criteria | Gewicht (%) | Ionic Vue + Capacitor | Framework 7 Vue + Capacitor | NuxtJS + Capacitor |
|---|---|---|---|---|
| **R1 - Installatie en development proces** | 10% | 9 | 9 | 6 |
| **R2 - Community** | 10% | 9 | 8 | 7 |
| **R3 - Ondersteunde platformen** | 15% | 9 | 9 | 9 |
| **R4 - Documentatie** | 15% | 8.5 | 8 | 8 |
| **R5 - Prestatie** | 15% | 8 | 8 | 8 |
| **R6 - Learning Curve, Toepasbaarheid en Toegankelijkheid** | 15% | 8 | 8 | 7 |
| **R7 – Builden en Deployment (CI/CD)** | 10% | 9 | 9 | 6 |

| | | | | |
|---|---|---|---|---|
| **R8 - Plugins, Native Functies** | 10% | 8.5 | 8 | 7 |

*Tabel 8 - Eindbeoordelingstabel voor overgebleven ontwikkelframeworks*




*Figuur 9 - Gemiddeld cijfer per framework (formule: Gewicht. * Cijfer = Bijdrage) bijv. 0,10 (10%) * 9 = 0,90.*

### 3.4.5 Aanvullende criteria

De frameworks Ionic en Framework7 liggen in de eindbeoordeling erg dicht bij elkaar, toch valt de keuze voor Ionic. Er zijn namelijk nog een aantal aspecten waar Ionic op de voorkeur krijgt en wint, welke niet betrokken zijn bij de beoordelingstabel, maar welke voor CODE14 wel belangrijk zijn. Sommige van deze criteria is niet meegenomen in de eindbeoordelingstabel omdat deze niet genoemd zijn in de interviews of medewerkers enquête.

**Open Source maar wel door een organisatie ondersteund**

Framework7 is gemaakt door één ontwikkelaar en heeft een community opgebouwd, welke allemaal een steentje bijdragen aan de ontwikkeling van het framework. Ionic daarentegen is Open Source maar hier wordt ook actief aan doorontwikkeld door het bedrijf zelf. Dit biedt meer zekerheid op snellere releases van de nieuwste features en stabiliteit.

**Grotere community, meer hulp**

Ionic bestaat langer en heeft een grotere community. Er is hierdoor meer informatie en hulp te vinden online. Ionic heeft 43.900 stars op GitHub (GitHub, Ionic, z.d.), waar Framework7 er 16.000 heeft (GitHub, Framework7, z.d.). Dit geeft ook enigszins het vertrouwen van ontwikkelaars weer.

**Ionic Ecosysteem**

De native runtime welke door beide frameworks gebruikt wordt is Capacitor. Deze native runtime is ook door het bedrijf Ionic ontwikkeld, het framework Ionic is als eerste framework overgegaan naar deze native runtime. De samenwerking tussen deze stack zal dus wat beter verlopen dan andere frameworks, buiten Ionic om.

**Interne kennis over Ionic**

Zoals eerder in dit document aangegeven, Ionic Angular werd al eerder bij CODE14 gebruikt. Angular lag voor velen net wat te ver van hun kennisgebieden af en daarom was dit geen efficiënte combinatie om in verder te gaan. Maar, de werking van Ionic, de elementen van Ionic en de mogelijkheden zijn enigszins bekend bij een aantal ontwikkelaars van CODE14. Toen Framework7 genoemd werd, wist niemand hier nog maar iets vanaf intern.

### 3.4.5 Low code i.c.m. Ionic Vue

**Deelvraag: Hoe kan er binnen een aansluitende ontwikkeltechniek gezorgd worden dat er herbruikbare code geschreven wordt?**

In de eerste fase van het onderzoek is er gekeken naar de huidige technologiestack. De ontwerpers van CODE14 maken voornamelijk gebruik van Figma. Figma is een designtool welke wereldwijd gebruikt wordt door designer in verschillende werkvelden. Voor het maken van prototypes kan deze tool ook gebruikt worden. Er is een meeting ingepland met Christian Lemmers (Designer en lid development board) om te kijken naar eventuele aansluitingen op Ionic i.c.m. Vue en Figma.

### 3.4.6 Component Based Development

Door Component Based Development toe te passen wordt door de ontwikkeling van nieuwe (web)-applicatie op termijn steeds minder arbeidsintensief. Door een juiste (door)-ontwikkeling van component op basis van de gebruikte ontwikkeltechnieken en programmeer concepten zou het overnemen en aanpassen van componenten eenvoudiger worden.

#### 3.4.5.1 Figma low-code

Figma low code is een plugin welke een design om kan zetten in Vue componenten. De gegenereerde code zal dan direct implementeerbaar moeten zijn in het project. Samen met Christian Lemmers is er geprobeerd gebruik te maken van deze plugin, maar hier is niks bruikbaars uit gekomen.

#### 3.4.5.2 Overlay

Overlay tech is net als Figma low code een plugin voor Figma, welke ontwerpen om kan zetten in Vue component. Samen met Christian hebben we in een vast tijdframe van een uur geprobeerd

om een bruikbaar Vue component uit te halen, met succes. Binnen een relatief kort tijdsbestek (15 minuten) is er een eerder ontworpen Figma designcomponent omgezet in een bruikbaar Vue component. Er zaten wel wat haken en ogen aan, de aansluiting op de CSS-variabelen vanuit de Nuxt Boilerplate, zijn niet te implementeren in Overlay. Deze moeten handmatig toegevoegd worden aan de gegenereerde component, maar zelfs dan kan er met juiste implementatie veel tijdswinst behaald worden, zonder dat dit ten koste gaat van aanpasbaarheid.

#### 3.4.5.3 Storybook en Nuxt Boilerplate

Om met een framework zoals Ionic Vue nog aansluiting te vinden op de Nuxt Boilerplate en StoryBook component, is er gekeken naar de structuur van een 'gemiddeld' Storybook component en de structuur hiervan. Aangezien Nuxt ook gewoon met Vue componenten werkt is er gekeken of de aansluiting tussen een Ionic Vue component en een NuxtJS component mogelijk is. Dit is gedaan door te kijken wat de verschillen zijn die in dit soort componenten voorkomen.

**Mogelijkheden Storybook**

Er kan een aparte sectie komen voor Ionic Vue componenten, waarin direct de uitwerking inclusief Capacitor gezet kan worden. Hierdoor wordt een ontwikkelaar 'geforceerd' component klare code te schrijven en wordt de geschreven code meer hergebruikt.

**Mogelijkheden aansluiting Nuxt Boilerplate**

Een belangrijk aspect uit de Nuxt Boilerplate is het gebruik van CSS-variabelen in plaats van SASS-variabelen. De reden hiervoor is dat een CSS variabel live geüpdatet kan worden, waar een SASS variabele überhaupt niet aangepast kan worden nadat deze initieel een waarde gekregen heeft. Dit is handig voor features zoals dark modus, waarden veranderen o.b.v. van statements, etc. Daarnaast, kunnen de SASS functies, typografie ook meegenomen worden in een Ionic Vue project, dit scheelt een hoop tijd in het opzetten van een nieuw project en maakt een project schaalbaar, denk hierbij aan verschillen in schermformaten zoals bij desktop, tablet en mobiel.

In Figuur 10 is een voorbeeld van een aantal SASS functies welke ontwikkeld zijn om harde pixel waarden vanuit een ontwerp of gegeneerd vue component om te zetten in een schaalbare eenheid, VW (View Width). Deze eenheid kijkt naar de de initiele breedte van het scherm van het device waar het op draait en geeft op basis hiervan een VW waarde terug. Deze waarde is een percentage van de totale breedte van het scherm, waardoor teksten of andere elementen altijd meeschalen met de breedte van een scherm.

```
@function desktop-vw($px) {
    @return ($px * 100vw) / 1920px;
}

@function tablet-vw($px) {
    @return ($px * 100vw) / 768px;
}

@function phone-vw($px) {
    @return ($px * 100vw) / 375px;
}
```

*Figuur 10 - SASS Functies voor het omzetten van vaste waarden in pixels naar een dynamische eenheid VW (View*

**Een dergelijke SASS functie is als volgt te gebruiken in CSS of SCSS:**

```
.example-class {
        padding: phone-vw(24px);  // Output zal 6,4vw wezen.
}
```

## 3.5 Bevindingen onderzoeksfase

**Deelvraag: Hoe kan er gezorgd worden dat de gemiddelde ontwikkelaar werkzaam bij CODE14, de voorkant van een hybride applicatie kan opbouwen?**

Door zo dicht mogelijk bij de huidige stack te blijven krijgen ontwikkelaars van CODE14 de mogelijkheid hun kennis over de gekozen stack te verbreden en zich comfortabeler te voelen in de producten die ze ontwikkelen. Door constant te wisselen tussen de 'op dat moment' beste ontwikkeltechnieken, blijft de kennis over specifieke frameworks laag, waardoor de gemiddelde CODE14 medewerker minder flexibel inzetbaar is. Een goed voorbeeld hiervan is Laravel. Laravel is een PHP backend framework waarin verschillende CODE14 backend developers zich specialiseren en hierover diepgaande kennis beschikken. Dit komt omdat dit altijd de standaard stack is geweest binnen CODE14, al vanaf 2014 toen alleen Harm-jan Hazelhorst (Mede-eigenaar en developer) nog voor CODE14 ontwikkelde.

### 3.5.1 Voor- en nadelen

Ionic biedt een heel ecosysteem aan, welke de juiste aansluiting biedt op meerdere fronten. Ionic Framework is bedoeld voor de realisatie van een goed uitziende hedendaagse mobiele applicatie met daarbij lichte differentiatie in de ontwerpaspecten als het gaat om native platformen, daarbij ben je niet gebonden aan de UI elementen welke Ionic aanbiedt en dus flexibel op het gebiedt van UI en UX.

#### 3.5.1.1 Voordelen

**Ionic Capacitor doorontwikkeling**

Capacitor is enorm snel met zijn doorontwikkeling, sinds eind Maart 2021 is er een release candidate uitgebracht voor versie 3 van de native runtime. De community groeit enorm snel omdat Capacitor makkelijk te integreren is met grotendeels van de bekendere hedendaagse front-end frameworks. Daarnaast wil het dedicated development team van Capacitor snel native functies van Cordova vervangen voor stabiele, door Ionic ondersteunde native functies.

**AppFlow Ionic CI/CD tooling**

Vanuit verschillende medewerkers is benoemd dat het indienen van applicaties in de Play- en App store soms een moeizaam proces is, vooral het indienen van applicaties bij Apple was altijd spannend, aangezien hier handmatig gecontroleerd wordt of de applicatie aan alle eisen voldoet. Het wachten op goedkeuring is hiernij een onderdeel onderdeel waar winst op te behalen valt, aangezien AppFlow hierover het volgende aangeeft: “Het uitgeven van live applicatie updates, content veranderingen, bug fixes, beta features en meer. Voor of nadat je applicatie in de Play- of Appstore is uitgegeven. Geen regels, geen wachttijd, geen vertragingen.” (Appflow - Ionic - The Cross-Platform App Development Leader, z.d.).

Het prettige hieraan is dat CODE14 hier met pure webapplicaties al aan gewend is. Er is één specifieke aftakking binnen een project op het versiebeheer systeem, waar de hostingpartij op getriggerd wordt voor automatische deployen. Bij AppFlow werkt dit hetzelfde, er wordt ‘geluisterd’ naar een specifieke aftakking van het project op het versiebeheersysteem en mochten hier veranderingen in plaats vinden wordt er automatisch een nieuwe build gemaakt in een beveiligde cloud omgeving, welke uitvoerbare applicaties maakt voor de aangegeven platformen zoals iOS of Android.

**Juiste aansluiting op de huidige kennis**

CODE14 beschikt over veel Vue kennis, daarnaast is er in het verleden gewerkt met Ionic i.c.m. AngularJS. Hierdoor is er al kennis van de mogelijkheden van het Ionic framework en is hierdoor de learning curve minder, omdat kennis makkelijker overgedragen kan worden en er al enige kennis is.

**Herbruikbare code en eventuele toevoegingen voor Storybook**

Omdat Vue een goede aansluiting biedt op de Component Based Development ontwikkelmethodiek en er intern gewerkt wordt aan een componenten bibliotheek d.m.v. Storybook, zou het interessant zijn om ook Vue componenten vanuit hybride applicaties in StoryBook te integreren.

**PWA en Vue plugins**

Ondanks dat Capacitor de brug van web naar native biedt, zijn er meer mogelijkheden om bepaalde functies aan te roepen zoals camera en bluetooth met gebruik van Vue. Hierdoor hebben we de mogelijkheid om een volledig functionerende webapplicatie (incl. native functies waarbij webcam aangeroepen kan worden.) om te zetten in een ‘native’ applicatie.

**Grote community**

Zowel Vue als Ionic hebben allebei een relatief grote community, bijna ieder probleem is al meerdere keren ondervonden en opgelost door een andere ontwikkelaar. Hierdoor kan er tijd worden bespaard op het zoeken naar een oplossing voor een specifiek probleem.

**Efficiënter ontwikkelen**

Door herbruikbare componenten, meer interne kennis over de stack en flexibiliteit op verschillende fronten zoals; ontwerp, ontwikkelstrategie en platformen. Kan er met zekerheid gezegd worden dat hybride ontwikkeling met gebruik van Ionic Vue & Capacitor er efficiënter mobiele applicaties ontwikkeld kunnen worden.

#### 3.5.1.2 Nadelen

**Cordova Deprecated Native Plugins**

Aangezien Capacitor relatief nieuw is, maakt het in sommige gevallen nog gebruik van Cordova packages. Omdat Cordova officieel niet meer doorontwikkeld wordt, zijn sommige van deze packages dus '*deprecated*'. Dit houdt in dat er geen support meer is voor deze plugin. Mocht er door een platform een verandering uitgevoerd worden waardoor de plugin niet meer functioneert, wordt er niet meer gewerkt aan een update om deze weer functioneel te krijgen.

**WebView**

Desondanks dat dit ook het voordeel met zich meebrengt, dat het mogelijk is om één en dezelfde applicatie ook te deployen als web-applicatie, heeft dit ook zijn nadelen. Een WebView neemt vaak meer werkgeheugen in beslag, omdat er een soort wrapper om de applicatie heen gebouwd is. Ook zal een applicatie vaak meer opslag in gebruik nemen dan een native applicatie en mogelijk op oudere smartphones en tablets minder soepel in gebruik zijn.

**VueJS niet de eerste keus voor Ionic**

Ondanks dat er al een tijd officiële ondersteuning is voor VueJS in Ionic, is het niet de eerste keus voor het frontend framework. Ionic was initieel ontwikkeld als UI framework voor AngularJS, waarna de keus is gemaakt om ook ReactJS te ondersteunen om zo direct te concurreren met React Native. Nadat VueJS ongeveer 1,5 jaar in beta is geweest, werd deze uiteindelijk in 2020 pas officieel ondersteund. Mede hierdoor is er voor minder gebruikte functionaliteiten nog geen VueJS specifieke documentatie geschreven.

**Afhankelijkheid van framework**

Omdat het een gegeven is dat Android en iOS niet spoedig zullen stoppen met innoveren en vernieuwen is het benodigd dat Ionic meegroeit met deze vernieuwingen en constant verbetert. Ionic is ontwikkeld door een organisatie en houd stand door de groei van de community. Deze community zal alleen groeien wanneer Ionic onderdeel blijft uitmaken van de beste technieken m.b.t. hybride ontwikkeling d.m.v. web technologieën.

#### 3.6.2.3 Front-end vergaderingen

Er zijn vergaderingen m.b.t. front-end development bijgewoond, waarin *'efficiënter ontwikkelen'* één van de agendapunten is. Hierbij worden positieve en negatieve ervaringen gedeeld welke zich hebben voorgedaan in de afgelopen maand(en). Tijdens de onderzoeksfase zijn er een aantal onderwerpen aangekaart welke invloed hebben op de ontwikkeltijd van de gemiddelde front-end developer. Hierbij is in het kader van efficiënter ontwikkelen het volgende onderwerp aangekaart:

**SASS Functies**

Voor het schrijven van responsive styling zijn media query's benodigd. Deze zorgen ervoor dat een bepaald stuk code alleen uitgevoerd wordt wanneer een scherm een specifiek aangegeven grote heeft. Zo kan er makkelijk smartphone, tablet of desktop specifieke code geschreven worden.

Omdat bij CODE14 dit vaak moet gebeuren en dit veel typewerk betreft, is er bij de vergadering gebrainstormd over een functie welke de vertaalslag van desktop naar mobiel automatisch maakt. Hierbij zijn wij tot de conclusie gekomen dat door de limitaties van SASS, er geen functies geschreven kunnen worden waarin al mediaquery's staan. Omdat een SASS functie al ingeladen wordt, voordat er viewport informatie opgehaald kan worden.

**Live Templates in IDE (Integrated Development Environment)**
Live templates geven ontwikkelaars de mogelijkheid om doormiddel van voor gedefinieerde termen, stukken code te genereren wat vaak herhaaldelijk getypt dient te worden. Denk hierbij aan media query's in (S)CSS, basistemplates van een Vue component, etc. Hierdoor kan er veel tijd bespaard worden door het herhaaldelijk typen van veel gebruikte functies, te laten genereren op de benodigde momenten.

### 3.6.2 Efficiënter ontwikkelen

#### 3.6.2.1 Native applicaties (Android & iOS)

VYBER heeft in 2019 een tweetal applicaties laten ontwikkeling, één voor het Android platform en één voor het iOS platform. Deze applicaties moesten aansluiten op de bestaande Vyber architectuur. Toen in 2019 de ontwikkeling voor deze applicaties begon, werden de uren geregistreerd in het urenregistratieplatform Toggl. Hier zijn door de betrokken medewerkers destijds 662 uur geregistreerd op de ontwikkeling van de native applicaties. Eind 2019 is de keuze gemaakt om over te gaan naar urenregistratieplatform Simplicate, hier is dan nog eens 171 uur geregistreerd op de ontwikkeling van de Vyber native applicaties. Aangezien er bij CODE14 nog nooit één zelfde project in zowel native ontwikkeltechnieken, als in hybride ontwikkeltechnieken is ontwikkeld. Zal er wanneer de Vyber hybride applicatie productie klaar is, een duidelijke vergelijking te zien zijn o.b.v. de bestede uren.

**Tijdsindeling**

Na navraag bij een collega welke betrokken is geweest bij het ontwikkelproces van de native applicaties, heeft deze aangegeven dat er weinig tot geen code reviews zijn gedaan tijdens het ontwikkeltraject. Aangezien dit wel gaat gebeuren bij de ontwikkeling van de hybride applicaties, zullen deze uren bij de vergelijking afgetrokken worden van het totale aantal uren besteed bij de ontwikkeling van de hybride applicaties.

## 3.7 Samenvatting deelvragen

### 3.7.1 Wat zijn de mogelijkheden van hybride ontwikkeling t.o.v. native ontwikkeling?

De bekendere hybride frameworks bieden een groot scala aan native API-functies aan. Hiermee wordt iedere native functie welke ontwikkeld is in een mobiele applicatie, welke gerealiseerd is door CODE14, afgevangen. Toch zijn er een aantal aspecten waar rekening mee gehouden moet worden tijdens de overweging tussen native en hybride ontwikkeling, zoals prestaties en vloeiende weergave, VR en AR ondersteuning en volledige ondersteuning van platform specifieke externe plugins zoals betalingsmethoden (Google pay, Apple pay), etc.

### 3.7.2 Wat zijn de eisen waar een hybride ontwikkeltechniek aan moet voldoen voor de gemiddelde CODE14 hybride applicatie?

In Tabel 5 zijn criteria opgesteld o.b.v. de antwoorden uit de enquête en de interviews. Een hybride framework moet makkelijk toepasbaar zijn op de huidige kennis en stack en moet geen limitaties hebben op het gebied van custom UI. De benoemde vereiste native functies (Camera, Bluetooth, Local Storage, NFC, QR-/Barcode scanner en Push Notificaties) zijn o.b.v. voorheen ontwikkelde mobiele of native applicaties. Verder is het een pré als het installatie en ontwikkelproces eenvoudig te doen zijn, waarbij er geen grote learning curve is. De mogelijkheid moet er zijn om in de toekomst gebruik te maken van TypeScript i.p.v. JavaScript.

### 3.7.3 Welke ontwikkeltechnieken worden het meest gebruikt binnen CODE14, als het gaat om front-end ontwikkeling?

Er wordt veel gebruik gemaakt van VueJS gerelateerde ontwikkeltechnieken. Daarbij wordt Storybook gebruikt als componenten bibliotheek en de Nuxt Boilerplate als starters template voor een nieuw project. Figma wordt gebruik door de product designers om ontwerpen te maken.

### 3.7.4 Wat zijn de voor- en nadelen voor CODE14 welke hybride ontwikkeling met zich meebrengt?

#### 3.7.4.1 Voordelen

Urenbesparing, daling van complexiteit t.o.v. native ontwikkeling en flexibeler inzetbaar personeel, goede aansluiting op de gemiddelde projecten en business cases van CODE14 en CODE14 invest.

#### 3.7.4.2 Nadelen

Geen toegang tot alle native functies, afhankelijkheid van doorontwikkeling van hybride framework en (mogelijk) mindere prestaties.

### 3.7.5 Hoe kan er gezorgd worden dat de gemiddelde ontwikkelaar werkzaam bij CODE14, de voorkant van een hybride applicatie kan opbouwen?

Zo dicht mogelijk bij de huidige technologiestack blijven. Framework specifieke kennis opbouwen en aansporen tot kennisdelen. Inzichtelijk houden waar een framework naar toe gaat en wat de concurrentie doet.

### 3.7.6 Hoe kan er binnen een aansluitende ontwikkeltechniek gezorgd worden dat er herbruikbare code geschreven wordt?

Door Component Based Development kunnen herbruikbare componenten ontwikkeld worden. Deze componenten dienen geschreven te worden conform de CODE14 codestyle, zodat de aansluiting met kleuren, padding en marges en typografie direct overgenomen kunnen worden in andere projecten, zonder dat de ontwikkelaar hier nog naar om hoeft te kijken.

# 4. Realisatiefase

In dit hoofdstuk wordt er meer verteld over het ontwikkeltraject van de VYBER-hybride applicatie. Hoe het front-end project is opgezet en ingericht conform de bevindingen van de onderzoeksfase.

## 4.1 Te ontwikkelen hybride applicatie

Voor de VYBER-hybride applicatie wordt een schatting gemaakt de **328 uur** o.b.v. het huidige ontwerp, waarop deze gebaseerd is, is te vinden in **Bijlage H**. Verder, zijn in **Bijlage G** de opgestelde requirements van de applicatie te vinden.

| Project | Ontwikkeltijd in uren | Verschil in procenten |
|---|---|---|
| Vyber Native (iOS & Android) | 833 uur | **100%** |
| Vyber Hybrid (Inschatting) | 328 uur | **-61%** |

*Tabel 9 - Tijdsbesparing voor ontwikkeling van dezelfde applicatie in verschillende ontwikkeltechnieken*

In Tabel 9 is te zien dat in het geval van de Vyber Native applicaties, er een ontwikkeltijdsbesparing zou kunnen zijn van 61% o.b.v. de gemaakte inschatting voor de ontwikkeltijd.

## 4.2 Software ontwikkel methoden

Wekelijks wordt de voortgang van het project bijgehouden, dit wordt gedaan in een briefing conform de scrum methode. Er is gezamenlijk met Christian Lemmers een schatting gemaakt van de te ontwikkelen componenten.

### 4.2.1 Deeltaken

Bij CODE14 wordt een ontwerp opgedeeld in deeltaken. Deze worden verwerkt als issues in GitLab, het versiebeheersysteem. Hierin is per project een Storyboard te vinden waarin deze taken staan, wie er aan een taak gekoppeld is en wat de status hiervan is. Op basis van deze taken wordt er een aftakking in het project gemaakt, waarin de taak uitgevoerd kan worden. Wanneer de ontwikkelaar denkt dat deze afgerond is, kan hij/zij de taak koppelen aan iemand die deze moet reviewen.

## 4.3 Project initialisatie

### 4.3.1 Ionic Vue + Capacitor

Het opzetten van het Ionic Vue project i.c.m. Capacitor is zorgvuldig gedaan. Hierbij zijn er een aantal stappen ondernomen om dezelfde styling aan te houden als de Nuxt Boilerplate en een passende aansluiting te vinden voor kwaliteitsborging.

#### 4.3.1.1 Initialisatie

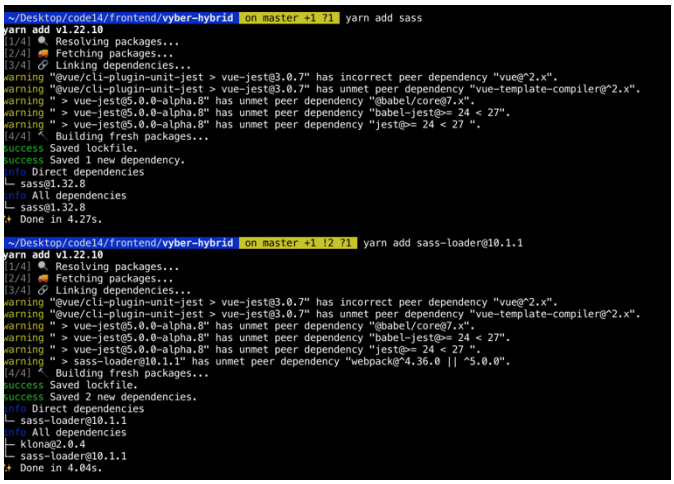
Om een nieuw Ionic project te realiseren, zal eerst de Ionic CLI geïnstalleerd moeten worden. Dit kan gedaan worden met de Node Package Manager. Doormiddel van de Ionic CLI wordt de mogelijkheid gecreëerd om het start commando aan te roepen.

**ionic start**

Hiermee wordt een reeks vragen gesteld, zoals welk front-end framework er gebruikt moet worden. Welke ionic template er gebruikt moet worden, etc. Wanneer het Ionic project opgezet is kan er in de map gegaan worden.

**YARN i.p.v. NPM**

CODE14's Development Board heeft in 2020 de keuze gemaakt voor alle nieuwe front-end projecten YARN te gebruiken. De voornaamste reden is dat deze package manager aanzienlijk sneller is dan NPM en duidelijker zou zijn met eventuele foutmeldingen.



*Figuur 11 - Initialisatieproces Ionic project*

Aangezien Ionic out-of-the-box NPM gebruikt, is de *package-lock.json* verwijderd uit de root van het project en is er vervolgens een **yarn install** aangeroepen om de gebruikte packages via deze manager te installeren.

**package.json**

Er zijn een aantal scripts aangepast in package.json. CODE14 medewerkers zijn gewend om in hun terminal **yarn dev** te gebruiken om een development server op te starten. Hierachter is het **ionic serve** commando geplakt, zodat deze wordt uitgevoerd op het moment dat yarn dev aangeroepen wordt. Het commando **ionic serve** is een Ionic CLI commando om een ionic development server op te starten op poort 8100. Hiervoor is globale of lokale installatie van de Ionic CLI dus benodigd.

**Toevoegen van SASS bestanden uit Nuxt Boilerplate**

In de Nuxt Boilerplate is in het verleden een structuur voor SASS bestanden uitgewerkt welke aansluiting biedt op de gemaakte StoryBook componenten. Om binnen een Ionic Vue project deze zelfde aansluiting te vinden moet er gebruik worden gemaakt van dezelfde structuur. Hiervoor moeten de volgende punten gebeuren:

- Implementeren van de SCSS-bestanden voor *functions*, *mixins*, *variables en animations*.

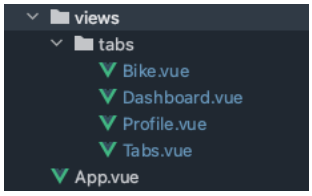
- Het installeren van de benodigde SASS packages: *sass*, *sass-loader*.
- Het aanpassen van de variables.css file in *main.scss* voor ondersteuning van SASS functies.
- Het aanpassen van de CSS-variabelen van Ionic, zodat de kleuren gelijklopen met die van de boilerplate.

**Het uitschrijven van een README-bestand**

Om het eenvoudiger te maken voor een ontwikkelaar om het project op te zetten is er een README.md file opgezet. Dit bestand is in markdown geschreven, waarin de basiscommando's staan om het project op te zetten en uit te voeren.

**Logische mappenstructuur**

De mappenstructuur van Ionic Vue komt niet geheel overeen met die van veelgebruikte frameworks van CODE14. In NuxtJS worden de routes automatisch aangemaakt o.b.v. de mappenstructuur van de aangemaakte pagina's. Om het overzichtelijk te houden worden verschillende pagina's dan ook in corresponderende mapnamen aangemaakt. Door het gebruik van Ionic Vue wordt de ontwikkelaar hier geheel vrij ingelaten, omdat we in de Vue Router zelf bepalen welke view er achter een route staat.



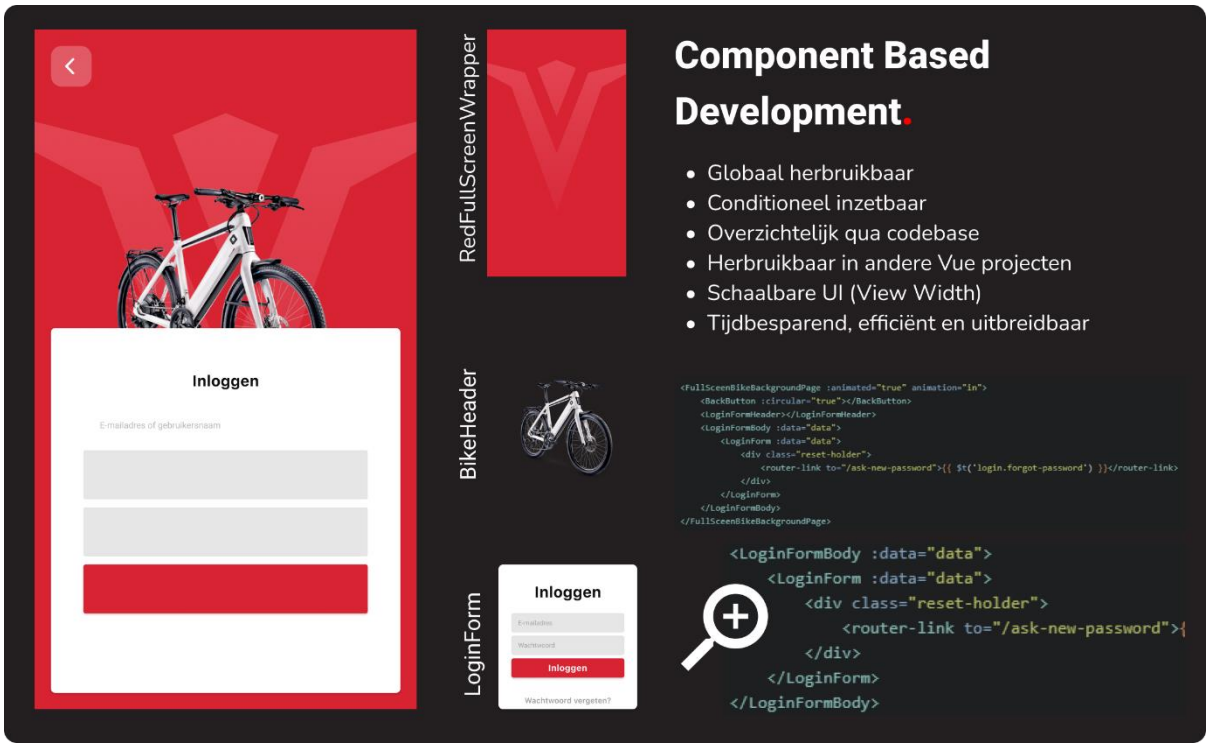

*Figuur 12 - Mappenstructuur*

## 4.4 Ontwikkeling

### 4.4.1 Ontwikkelstrategie

Als eerst is het visuele ontwerp ontwikkeld, waarna er over is gegaan naar de gebruikersfuncties en de e-bikefuncties. Om op een efficiënte wijze te werk te gaan, is er voor de start van de ontwikkeling gekeken naar het ontwerp. Zo kan er van tevoren een aanpak bedacht worden om zo efficiënt mogelijk met componenten te werken, daar waar een element niet vaker terugkomt, is het ook niet nodig om deze dynamisch op te bouwen. Wanneer een globaal te gebruiken element zoals een input, button, laad animatie o.i.d. wordt gerealiseerd, dient de styling gedaan te worden in het *_project.scss* bestand.

### 4.4.1.1 Component Based Development




*Figuur 13 - Component Based Development in VYBER applicatie*

In Figuur 9 is te zien hoe het principe van component based development is toegepast binnen de VYBER Hybride applicatie. Door gebruik te maken van de mogelijkheden welke VueJS biedt, is het mogelijk om gehele user interfaces om te zetten in verschillende herbruikbare en uitbreidbare componenten.

Vue Slots is hierbij van toegevoegde waarde, omdat je componenten in componenten kan integreren, zodat er een logische en overzichtelijke opbouw van een scherm of pagina ontstaat. In Figuur 9 is een voorbeeld te zien van het inlog scherm van de Vyber applicatie. Omdat in het ontwerp de verschillende componenten vaker terugkomen zijn ze opgedeeld in logische componenten waar nog specifieke aanpassingen aan toegevoegd kunnen worden.

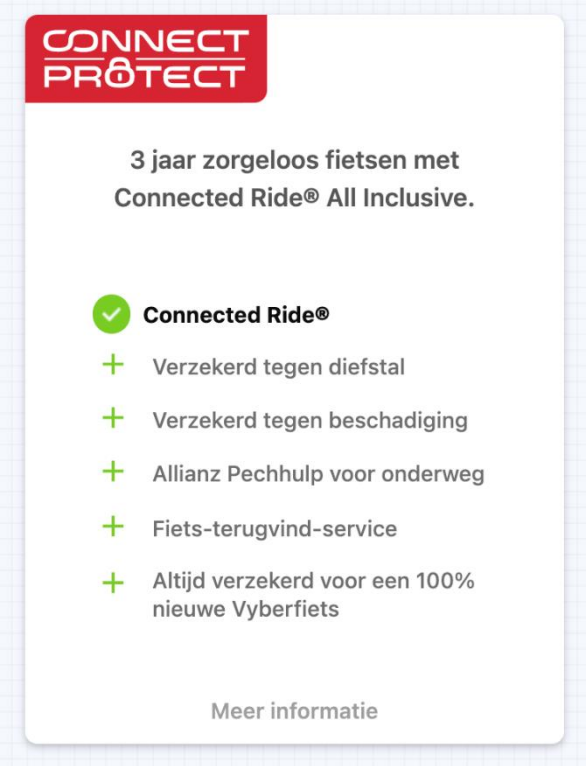
### 4.4.1.2 Schaalbare ontwerp

Het grootse deel van de front-end projecten binnen CODE14 zijn gebaseerd op VueJS (gerelateerde) framework(s). Hierdoor is de mogelijkheid ontstaan om goed opgebouwde componenten te herbruiken binnen andere projecten. Immers is het zo dat de Vyber Hybride applicatie puur ontworpen is voor mobiel, maar door de opbouw van de componenten en het efficiënte gebruik van verschillende SASS functies is de applicatie op verschillende schermformaten goed bruikbaar.

#### 4.4.1.3 Toepassen Overlay Low-code

Overlay is gebruikt om een aantal Vue componenten te genereren voor de VYBER Ionic Vue applicatie. Om een beeld te krijgen of deze code bruikbaar is, zijn de gegenereerde componenten samen doorlopen met Christian Lemmers. Hier is de conclusie getrokken dat de gegenereerde HTML-structuur van het component bruikbaar is. De gegeneerde CSS is niet volgens de CODE14 styleguide en dit moest deels herschreven worden.

Een ander struikelpunt is dat het ontwerp voor de VYBER-applicatie in Figma niet gemaakt is conform de richtlijnen van Overlay. Hiermee wordt gedoeld op de naamgeving van elementen en het groeperen van elementen tot één component.




*Figuur 14 - Gegenereerd Overlay component*

#### 4.4.1.4 Componenten in Storybook

Om componenten in Storybook te plaatsen, moet het CODE14 Storybook project lokaal geïnstalleerd worden. Het volledige Vue component kan gekopieerd worden vanuit het originele project en geplakt worden in het Storybook project. Vervolgens dient een collega de broncode en compleetheid van het component te controleren. Bij goedkeuring kan het component dan bij de lijst van bruikbare componenten toegevoegd worden door deze samen te voegen met de hoofd branch van het Storybook project. Afhankelijk van de bijkomst van externe plugins dient er duidelijk documentatie geschreven te worden over de implementatie van het specifieke component.

## 4.5 CI/CD, Pipelines en kwaliteit waarborgen

Bij CODE14 wordt er gebruik gemaakt van pipelines in het versiebeheersysteem. Deze pipelines worden aangeroepen wanneer er nieuwe bestanden, code of wijzigingen naar het versie beheersysteem worden geüpload. Tijdens het inrichten van het Gitlab project is er een zogeheten '.gitlab-ci.yml' bestand opgesteld. Hierin wordt gespecificeerd wat er moet gebeuren wanneer er wat nieuws op het versiebeheer systeem wordt gezet.

```yaml
cache: &global_cache
  key: ${CI_COMMIT_REF_SLUG}
  paths: [node_modules, .npm/]

yarn:
  image: node:14-alpine
  stage: build
  except: [schedules]
  interruptible: true
  artifacts:
    expire_in: 1 day
    paths: [node_modules/]
  script:
    - cp .env-example .env
    - yarn install --frozen-lockfile
    - yarn build

qa-checker:
  image: registery.code14.nl/code14/innovation/qa:latest
  stage: test
  needs: [yarn]
  except: [schedules]
  interruptible: true
  cache:
    <<: *global_cache
    policy: pull
  script:
    - qacheck-ci
```

*Figuur 15 - gitlab-ci.yml bestand voor VYBER Hybrid project*

**Cache:** Dit zorgt ervoor dat wanneer er geen wijzigingen in packages zijn aangebracht, ze ook niet opnieuw geïnstalleerd worden. Dit versnelt het pipeline proces enorm.

**Yarn:** NPM gebruikt de laatste docker versie 14-alpine als image. Dit wordt uitgevoerd in de build stage van het pipeline-proces. Dit proces installeert alle packages en cached hierbij het package.json bestand, om te controleren of er veranderingen zijn t.o.v. de laatste keer dat de pipelines gedraaid zijn. Daarna kopieert deze het environment voorbeeld bestand in een bruikbaar environment bestand en draait een build.

**QA-checker:** QA-checker controleert de code styling van CODE14. Dit betreft het variabele gebruik, mismatch van parameters, codestructuur in verschillende programmeertalen, etc. Deze wordt ook lokaal gebruikt zodat mogelijk problemen van tevoren al bekend zijn. Mocht een ontwikkelaar dit lokaal niet gecontroleerd hebben, dan zal de build in de pipelines de fouten weergeven. Dit resulteert in besparing van tijd. De fouten die hierdoor boven water komen, hoeft de reviewer van de merge request al niet meer te controleren.

**Ionic Appflow:** De organisatie Ionic, heeft een platform ontwikkeld welke applicatie distributie kan automatiseren, genaamd AppFlow. Op basis van een trigger vanuit het versiebeheer systeem zal dit platform automatisch de applicaties builden, publiceren en updaten. De integratie van dit platform is optioneel en biedt verschillende pakketten met verschillende prijzen. Afhankelijk van het verwachte aantal eindgebruikers, consequentie in doorontwikkeling en doorlooptijd is integratie van dit CI/CD platform sterk aan te raden.

## 4.6 Back-end

Er is voor de native applicaties Laravel back-end applicatie ontwikkeld, waar verschillende API-endpoints om de front-end applicaties van gegevens te voorzien, aan te laten passen, toevoegen of verwijderen.

### 4.6.1 Docker

**Wat is docker?**

“Docker is een container-technologie waarbij er bij gebruik van een container geen rekening gehouden hoeft te worden met hardware- of configuratie specificaties” - (Combell, 2020). Wanneer een project op zowel MacOS, als Windows foutloos moet kunnen draaien, is het handig dat het project op één specifiek platform draait, zodat er geen platform specifieke configuraties aangepast hoeven te worden.

De API van VYBER is ingericht om in zo’n docker container te kunnen draaien, net zoals alle andere back-end projecten van CODE14. Dit gebeurt om mismatches van package versies te voorkomen en daarmee als ontwikkelaar makkelijker tussen projecten te kunnen schakelen.

#### 4.6.1.1 Opzetten van Docker

**Traefik**

Om gebruik te maken van deze docker containers, zal docker ingericht moeten worden volgens de standaarden van CODE14. Hierbij wordt Traefik (Traefik, 2016) gebruikt. Wanneer je een specifieke URL probeert te benaderen, bijvoorbeeld ‘vyber-api.dev.test’, zorgt Traefik ervoor dat deze verwijst naar de juiste docker container.

**Aliassen**

Om eenvoudig een Docker container te kunnen opstarten of afsluiten. Kunnen er in de .zshrc aliassen worden aangemaakt, welke de ontwikkelaar lange of ingewikkelde terminal commando’s eenvoudig uit laat voeren. Neem als voorbeeld de `‘docker-compose exec vyber-api php artisan migrate -seed’`. Dit commando zorgt ervoor dat de migratietabellen van de database gedraaid worden, zodat de database structuur opgezet wordt en vervolgens gevuld wordt met data van een database seeder. Hiervoor kan een alias gemaakt worden, bijvoorbeeld `‘artisan migrate -seed’`. Dit is minder typewerk, overzichtelijker en is mogelijk omdat alle laravel back-end projecten al voor een Docker container ingericht zijn.

**Altijd actief**

Een groot voordeel is dat met de huidige instellingen, de Docker containers altijd actief zijn wanneer de laptop of pc van de ontwikkelaar aan staat. Tijdens ontwikkeling van een front-end aansluitend op een API. De back-end van een project hoeft dus maar één keer opgestart te worden.

## 4.7 Data handling & State management

**HTTP Client**

Voor het afhandelen van HTTP Requests is er gekozen om **Axios** (GitHub, Axios, z.d.) te gebruiken. De keuze is hiervoor gevallen omdat deze http-client veel gebruikt wordt binnen projecten van CODE14. Dit is een promise-based http-client, wat wil zeggen dat er een asynchrone actie wordt uitgevoerd waarin de belofte wordt gedaan dat deze zal slagen of falen. Hiermee kan er data worden verzonden, opgevraagd of verwijderd uit de API. In het gedeelte waar zeker is dat de actie geslaagd of gefaald is, kunnen vervolg opdrachten aangeroepen worden, waardoor de eindgebruiker altijd een melding kan krijgen over de staat waarin de applicatie zich bevindt.

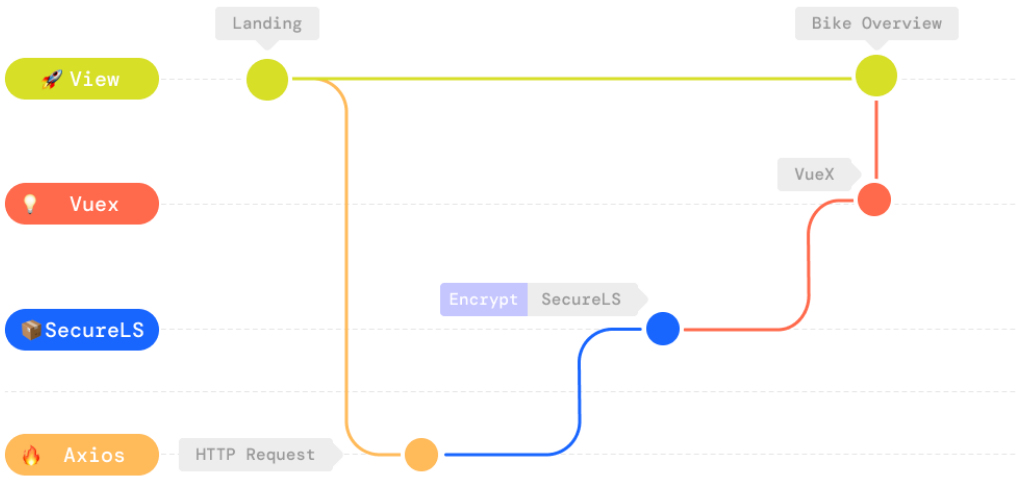
**Lokale opslag**

Om de time-to-interactive zo laag mogelijk te houden, wordt er data lokaal in de applicatie opgeslagen. Denk hierbij aan gegevens van de authentiseerde gebruiker en de fietsen waar deze een relatie mee heeft. Hiervoor wordt SecureLS (GitHub, SecureLS, z.d.) gebruikt. Deze encrypt de aangeleverde data en slaat deze vervolgens op in de local storage.

**State Management**

VueX is gebruikt om de state van de applicatie bij te houden. Deze voorziet de views van de benodigde data om weer te geven aan de gebruiker. Hierbij wordt er gebruik gemaakt van interfaces (models) van de veelal gebruikte objecten zoals een user of een bike.

In Figuur 16 is gevisualiseerd welke stappen er ondernomen worden, wanneer een gebruiker doormiddel van een login request zich navigeert naar de 'Bike Overview' pagina, waar vervolgens de gekoppelde fiets(en) getoond worden.




*Figuur 16 - Data Handling & State management*

### 4.7.1 Uitvoeren van een request

In het onderstaande stuk broncode is te zien hoe een 'login' verzoek verstuurd wordt naar de API. Om deze functie aan te roepen wordt er een `'userData'` object en een `'messages'` object verwacht. In de userData wordt de gebruikersnaam (email) en een wachtwoord verpakt, welke opgestuurd worden doormiddel van een Axios post request. De API url wordt globaal in het `'.env'` bestand geplaatst, zodat bij wijzigingen deze door de gehele applicatie wordt toegepast.

De `.then()` en `.catch()` methoden worden aangeroepen afhankelijk van de reactie van het verzoek. Bij een OK status wordt de code in de `.then()` aangeroepen, bij een fout status (4**) wordt de catch uitgevoerd.

De status van het proces wordt bijgehouden in een boolean genaamd loggedIn, deze wordt alleen op true gezet wanneer het loginproces geslaagd is. De `setUser` functie regelt het opslaan van de profielgegevens van een gebruiker in de secure local storage. Vervolgens worden de fietsen van de gebruikers, de access_token voor authenticatie bij vervolg verzoeken en de gebruiker zelf opgeslagen in de Vuex Store, welke globaal toegankelijk is door de gehele applicatie, wat gebruikt wordt voor de state management.

```
async login(userData: object, messages: []) {
        let loggedIn = false;
        await axios.post(process.env.VUE_APP_API_URL + 'api/auth/login', userData).then((res) => {
                loggedIn = true;
                this.setUser(res.data.user).then(() => {
                        BikeService.setUserBikes(res.data.bikes);
                        this.setAccessToken(res.data.access_token).then();
                        store.state.access_token = res.data.access_token;
                        store.state.user = res.data.user;
                        store.state.bikes = res.data.bikes;
                        ToastService.presentToast(messages.success, 2500, 'success');
                });
        }).catch(e => {
                loggedIn = false;
                ErrorHandler.handleListOfErros(e.response.data.errors);
                ErrorHandler.handleListOfErrors(e.response.data.errors);
        });
},
```

### 4.7.2 Scan-bike flow

Als een gebruiker een koppeling wilt maken met een fiets, wordt op basis van bepaalde properties gekeken naar welk scherm de gebruiker genavigeerd dient te worden. Het kan zijn dat er bij een gescande fiets een gratis abonnementsvorm zit, waardoor de gebruiker vervolgens een subscription request dient te doen. Hierdoor wordt deze geleidt naar de `FreeConnectedRide` pagina.

Mocht dit niet het geval zijn, maar heeft de fiets al wel een abonnement, bijvoorbeeld bij het overkopen van een tweedehands VYBER e-bike. Dan wordt er gekeken of er een upsell mogelijk is (wanneer er een basis abonnement en geen all-inclusive abonnement actief is.). Mocht dit het geval zijn, wordt deze naar de `ConnectedRideUpsell` pagina genavigeerd, anders wordt deze naar de `BikeOverview` pagina genavigeerd en zal de gebruiker daar zijn nieuw gescande fiets treffen.

Mocht alles van het bovenstaande niet het geval zijn, dan wordt de gebruiker doorgeleid naar de `NoConnectedRide` pagina doorgeleid, waar alleen de mogelijkheid is om het upsell proces in te gaan, of te navigeren naar de profielpagina van de geautoriseerde gebruiker.

Hieronder is te zien hoe dit in code is uitgewerkt binnen de `'AddBikeToAccount'` pagina.

```
BikeService.attachRequest(data).then((foundBike) => {
        this.buttonPressed = false;
        if (foundBike.model.default_subscription_type_id) {
                this.$router.push({name: 'FreeConnectedRide', addToHistory: false}).then();
        } else {
                if (foundBike.currentSubscription) {
                        if (foundBike.currentSubscription.upsellEnabled) {
                                this.$router.push({ name: 'ConnectedRideUpsell' });
                        } else {
                                this.$router.push({ name: 'BikeOverview' });
                        }
                } else {
                        this.$router.push({ name: 'NoConnectedRide', addToHistory: false}).then();
                }
        }
});
```

## 4.8 Native Functies

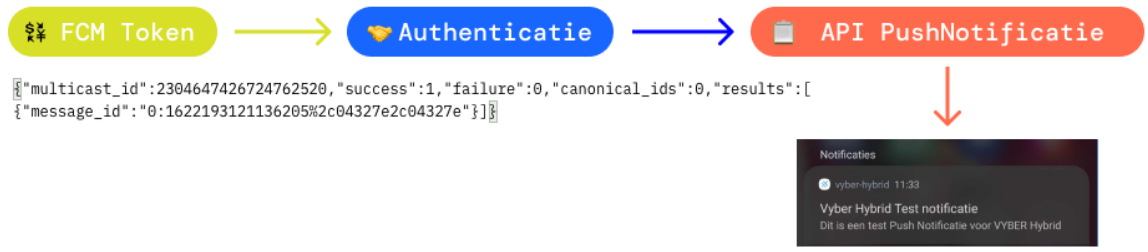
In de VYBER-applicatie worden een aantal native functies aangeroepen, namelijk; Camera, QR-Scanner, Local Storage, DeviceInfo, Netwerkinformatie en Push Notificaties. Omdat de applicatie met Ionic Vue en Capacitor is opgezet zijn er voor sommige functies meerdere oplossingen mogelijk. Zo is er voor de QR-Scanner voor een Vue package gekozen, zodat deze ook eventueel op web kan functioneren. Capacitor biedt voor een aantal van zijn native functies ook de PWA-variant (Progressive Web App) aan, welke op dezelfde wijze wordt aangeroepen, zoals de camera functie.

**Local Storage**

Ionic biedt een eigen variant voor secure local storage aan, maar omdat Axios een promise-based http-client is. Is er voor de SecureLS (Secure Local Storage) gekozen, deze werkt ook d.m.v. promises, waarmee er verzekerd wordt dat de aangevraagde informatie aangeleverd wordt, of er een fout wordt weergegeven wanneer de aanvraag niet slaagt.

**Push Notificaties**

Push Notificaties worden geregeld door de Push Notificaties integratie van Capacitor (Ionic Capacitor, z.d.) i.c.m. de Capacitor community FCM plugin (Capacitor Community FCM, z.d.). Voor de authenticatie van een gebruiker is het benodigd dat er een zogeheten FCM-token (Firebase Cloud Messaging token) meegestuurd wordt. Dit token zorgt ervoor dat wanneer de API een push notificatie gestuurd moet worden, dat Firebase weet naar welk toestel deze notificatie gestuurd moet worden.




*Figuur 17 - FCM Token to Push Notification*

## 4.9 Bevindingen realisatiefase

Tijdens de ontwikkeling van de VYBER-hybride applicatie d.m.v. het framework Ionic i.c.m. Vue en Capacitor en de low-code plugin Overlay zijn er een aantal bevindingen gedaan, deze zullen in dit subhoofdstuk worden besproken.

### 4.9.1 Positieve bevindingen

#### 4.9.1.1 Gebruik van Vue plugins

Tijdens de ontwikkeling zijn er keuzes gemaakt voor de te gebruiken externe plugins. Veel van de plugins welke Capacitor biedt werken zowel op een gecompileerde applicatie (Android of iOS) als op een web-applicatie. Immers is tijdens de onderzoeksfase ontdekt dat Ionic Capacitor gebruik maakt van een WebView, wat zou betekenen dat op de achtergrond de code omhulst wordt door een soort wrapper en daardoor dus niet vertaald wordt naar de ontwikkeltechnieken platform specifiek zoals Java (Android) en Swift (iOS). Mede hierdoor is de keuze gemaakt om voor functionaliteiten zoals de QR-Scanner, een Vue plugin te gebruiken.

### 4.9.2 Negatieve bevindingen

#### 4.9.1.2 Uitgebreide globale styling

Er is een bestand genaamd *_project.scss* aangemaakt, welke globaal ingeladen wordt om standaard componenten vanuit daar te stylen, denk hierbij aan knoppen, titels, hergebruikte elementen, etc. Immers is het zo dat, hoe groter het project wordt, hoe onoverzichtelijker dit bestand wordt. Omdat dit bestand ook in de Nuxt Boilerplate gebruikt wordt, is dit in het VYBER-project niet gewijzigd, maar wel in de front-end meeting van CODE14 aangekaart.

**De oplossing**

Omdat er tijdens de ontwikkeling van de Nuxt Boilerplate specifiek gekozen is om zo min mogelijk aparte SASS bestanden te maken, om het overzicht te bewaken. Is de keuze gemaakt om voor specifieke globale componenten wel SASS bestanden te maken, alleen wanneer dit nodig lijkt. De gedachte achter het *_project.scss* bestand is uit verband getrokken in lopende projecten, waar een refactor voor ingepland staat om het overzicht weer te herpakken.

#### 4.9.1.3 Achterstallige plugins en packages

Voor de implementatie van de QR-Scanner is er lang gezocht naar een geschikte package. Vanuit Capacitor en hun community zijn er een aantal opties, maar deze functioneerde niet op alle platformen. Ook bij het zoeken van Vue plugins waren er beperkingen omdat er Vue 3 in de applicatie wordt gebruikt. Als een package gemaakt voor een oudere Vue versie (Vue 2.*) niet altijd goed op aansloot op de implementatie methode van Vue 3.

# 5. Discussie

In dit hoofdstuk worden er suggesties gedaan en wordt de algehele voortgang van de applicatie besproken.

## 5.1 Efficiënter werken

### 5.1.1 Projectopzet

Wanneer een nieuwe ontwikkelaar een project op moet zetten. Moet het eenvoudig zijn om een project werkend te krijgen. Na gesprekken met collega's gevoerd te hebben, kwam naar voren dat het niet altijd eenvoudig is om als nieuwe ontwikkelaar een bestaand project op te zetten. Er moet meer aandacht besteed worden aan het uitschrijven van duidelijk documentatie voor de opzet, maar ook voor de eventuele benodigde gegevens, zoals sleutels van externe plugins zoals Google Maps.

### 5.1.2 Low-/no code

Door het onderzoek en het gebruik van verschillende low-code tools/plugins, is goed te zien welke kant de algemene 'applicatieontwikkeling' opgaat. Er is een beeld geschept over de huidige situatie m.b.t. low-code en zelfs no-code technieken. Met de groei van deze technieken is ook te zien dat een bedrijf zoals CODE14 niet stil kan blijven staan. De teams binnen CODE14 zijn vaak opzoek naar frisse ideeën, welke efficiënter, klantvriendelijker of minder complex zijn. Wanneer er nu gebruik wordt gemaakt van low- of no code technieken, zal dit de volledige overstap in de toekomst vereenvoudigen wanneer deze technieken volwassen genoeg zijn. Daarmee kan er gesteld worden dat Overlay een goede eerste stap is.

Er is na het aanwakkeren van het onderwerp no-/low code een pilot gestart voor een webapplicatie waarbij het no-code platform Webflow wordt gebruikt. Hierbij wordt er gekeken hoeveel tijd er bespaart kan worden voor het ontwikkelen van een kleine MVP (Minimal Viable Product) webapplicatie, welke daarnaast ook in code ontwikkeld gaat worden. Ik denk dat het belangrijk is dat een team constant probeert te vernieuwen en meegaat met de trends. Maar het gevaar is wel dat door no-code tooling de inzetbaarheid van medewerkers wel ingedamd, aangezien dit voornamelijk design tooling is en grotendeels van de medewerkers bij CODE14 software ontwikkelaars zijn.

## 5.2 Afronding applicatie

### 5.3.1 Openstaande punten

Visueel is de gehele applicatie afgerond, ook functioneel is er al heel wat afgerond, er zijn nog een aantal features die ontwikkeld of afgerond dienen te worden, namelijk:

**Niet afgeronde functionaliteiten**

- API endpoint voor anti-diefstal en killmodus aanroepen
- API endpoint voor gebruiker aan fiets toevoegen aanroepen
- API endpoint voor gebruikers van fiets aanroepen
- Dashboard gegevens vanuit de API opvragen en tonen
- Gebruikers beheer van een e-bike
- Voorwaarden, privacy en garantiebepaling data vanuit de API ophalen en tonen
- Het tonen van push notificaties op iOS (niet getest)
- Het openen van een deeplink op iOS (niet getest)

Op basis van persoonlijke ervaring tijdens de ontwikkeling van deze applicatie, is voor het ontwikkelen of afronden van deze functionaliteiten +- 60 uur geschat. Deze schatting is echter wel exclusief testen, mocht dit een accurate schatting zijn, zou de totale ontwikkeltijd van de hybride applicatie op 305 uur uitkomen.

**Deeplinks vanuit e-mails**

Deeplinks zijn momenteel zo ingesteld dat wanneer een smartphone de VYBER-hybride applicatie geïnstalleerd heeft, er bij een link vanuit een andere applicatie, welke linkt naar *https://vyber.com*, gevraagd wordt of deze geopend dient te worden in de VYBER-hybride applicatie of in Google Chrome browser. Het zou handig zijn dat hier een apart domein voor komt, zodat conflicten tussen URL's van de website en hybride applicatie vermeden kunnen worden.

**Push Notificaties**

Om Push Notificaties op iOS en Android afgerond te krijgen voor productie, dienen er een aantal stappen verricht te worden tijdens het deployen naar productie. Tijdens de implementatie van Push Notificaties is er getest op Android. De lokale Android build beschikt hierdoor al over alle benodigde toevoegingen. Tijdens app registratie van iOS dienen er een aantal stappen uitgevoerd te worden, om de gebruiker bij eerste keer openen van een verzoekprompt te voorzien voor het ontvangen van push notificaties. In het bijleverde README bestand binnen het hybride project, wordt verwezen naar de benodigde stappen om dit functioneel te krijgen.

**Conditional Rendering**

De applicatie bevat een aantal edge cases welke d.m.v. conditional rendering afgevangen dienen te worden. Bijvoorbeeld, wanneer er meerdere gebruikers op één fiets geregistreerd staan, moet de eigenaar van die fiets de gebruikers kunnen beheren. De normale gebruikers mogen wel de functionaliteiten van de fiets gebruiken maar niet gebruikers toevoegen, aanpassen of verwijderen.

**Testen**

Voordat het product in productie gezet kan worden, ter vervanging van de huidige applicaties, dienen alle functionaliteiten afgerond en doorlopen te zijn. Hiervoor kunnen 'jest' tests geschreven worden in het front-end project. Immers wordt dit onderwerp binnen het Development Board veelal besproken, de uitkomst hiervan zal geïmplementeerd worden binnen alle front-end stacks het is nog niet bekend met welke tooling dit dus gedaan wordt.

Het is voor de afronding van de hybride applicatie wel van toepassing dat de verschillende edge cases goed nagelopen worden voor livegang. Vooral in de flow van het scannen van een fiets, kunnen er doormiddel van de verkregen data vanuit de API veel verschillende scenario's afspelen. Om de eindgebruiker van een soepele ervaring te voorzien, dient dit op verschillende devices (oud en nieuw) goed getest te worden.

**Error Handling**

De Vyber API geeft foutmeldingen terug in verschillende scenario's, bij het foutief aanleveren van data aan een endpoint worden er foutmeldingen in een array teruggestuurd naar het front-end. De meldingen welke hieruit voortkomen zijn niet altijd duidelijk. Bijvoorbeeld bij een login request, wordt de melding 'auth failed' getoond wanneer een eindgebruiker een fout e-mailadres of wachtwoord heeft ingevoerd.

**Bugs**

Er is een bug bekend m.b.t. Vue en Vue Router i.c.m. Ionic Tabs, waar de profielpagina over de dashboard pagina heen geladen wordt in scenario's waarbij beide pagina's al een keer geladen zijn. Deze bug is aangegeven als issue binnen het project en aangegeven bij Harm-Jan Hazelhorst. Dit is een bekend probleem binnen de Ionic Vue community en hiervoor is een issue (GitHub, z.d.) op hun GitHub repository.

### 5.3.2 Updates

In de drie artikelen welke gebruikt zijn in de eerste fase van het onderzoek, werd meermaals benoemd dat een nadeel van Ionic het gebrek aan een 'native' live-reload functionaliteit mist. Dit is iets wat Cordova destijds wel geïmplementeerd had, maar niet altijd even stabiel werkte. Capacitor heeft tijdens de afstudeerperiode aangekondigd dat er een geschikte release candidate

is, welke beschikt over precies deze functionaliteit. Aangezien de VYBER-applicatie nog in ontwikkeling is, is mijn advies om Capacitor bij officiële release direct te updaten, zo kan er doormiddel van hot reloading direct ontwikkeld worden op een Android of iOS device.

### 5.3.3 Deployment

Bij CODE14 is er ervaring met het deployen van mobiele applicaties naar de App- en Play stores. Maar mochten er intenties zijn om een web variant live te zetten, is dit ook mogelijk. Het is van belang dat hierbij de environment credentials op productie gezet worden.

**CI/CD tooling**
Ionic App Flow zou gebruikt kunnen worden wanneer de VYBER-hybride applicatie in productie gezet kan worden. Maar voor de code kwaliteit hoeft dit al niet meer te gebeuren, aangezien dit wordt gedaan doormiddel van de bestaande pipelines in het versiebeheer systeem en door de controle van merge requests. Wel kan het handig zijn om te controleren om de platform specifieke vereisten.

**Benodigde aanpassingen voor webapplicatie**
Om de VYBER-hybride applicatie in productie te zetten als web-applicatie zijn er een aantal wijzigingen die gedaan moeten worden in de API. Zoals het accepteren van authenticatie requests van het 'mac' en 'windows' besturingssysteem, deze worden momenteel niet geaccepteerd bij een login request. Ook moet het FCM-token *nullable* gemaakt worden, aangezien push notificaties niet mogelijk zijn op een web-applicatie.

### 5.3.4 Verbeterpunten

**Case styling**
Tijdens de ontwikkeling van de koppeling tussen de API en de hybride applicatie kwam naar voren dat er in de ontwikkeling van verschillende endpoints in de API niet consistent gewerkt wordt met de case styling. Bij het inloggen wordt er namelijk tussen woorden een `snake_case` gebruikt (bijv. 'first_name'), terwijl bij de registratie `camelCasing` wordt gebruikt (bijv. 'firstName'). Het is handig om hier consistent mee om te gaan en in wijzigingen door te voeren bij nieuwere versies van de backend.

**Overgang naar TypeScript**
Ook uit de enquête kwam voort dat er meer kennis opgedaan moet worden over TypeScript en het gebruik hiervan. De overname van types in modellen vanuit de back-end is dan veel minder foutgevoelig. Ook in de realisatie van de VYBER-hybride applicatie is dit principe in de services en helpers toegepast om altijd te kunnen aangeven wat er wordt verwacht. Immers moet er intern wel enigheid geschept worden over de implementatie hiervan en zal dit ook gelijkgestemd moeten zijn met de te koppelen back-end applicaties, hierbij is het dus van belang dat een team zoals het development board kijkt naar de impact die een volledige overgang van Javascript naar Typescript kan hebben.

# 6. Advies

In dit hoofdstuk zullen de onderwerpen welke van toepassing zijn op het beantwoorden van de hoofdvraag worden toegelicht.

## 6.1 Conclusie

***Hoe kan CODE14 op efficiëntere wijze toekomstgerichte hybride applicaties ontwikkelen en onderhouden?***

### 6.1.1 Framework

Uit het errichte onderzoek kwam het framework **Ionic i.c.m. VueJS** en **Capacitor** als beste naar voren. Na de opgedane kennis en praktijkervaring in de onderzoeks- en realisatiefase kan er worden geconcludeerd dat er veel raakvlakken zijn met de gebruikte ontwikkeltechnieken bij CODE14. De ontwikkelde Vue componenten kunnen opnieuw gebruikt worden binnen Vue, Nuxt en Gridsome projecten. Hiermee worden alle huidige lopende front-end projecten van CODE14 afgevangen. Omdat ieder front-end project van CODE14 gebruik maakt van HTML, SASS en JavaScript/TypeScript biedt het naast de variaties tussen JavaScript en TypeScript genoeg mogelijkheden om zo efficiënt mogelijk te ontwikkelen en onderhoud te plegen.

#### 6.1.1.1 Toekomstgericht

Ionic biedt een heel ecosysteem aan, welke goed op elkaar aansluiten. Het Ionic framework voor de UI elementen, om zo dicht mogelijk bij een native gebruikservaring te komen. Capacitor als native runtime voor de native functies en bridge, AppFlow voor de Continues Integration en Deployment. Daarnaast bestaat het framework Ionic al sinds 2013, wat niet heel lang lijkt, maar op de markt waar hun zich in bevinden, behoort dit framework tot één van de beste in zijn soort. Er zou bij een business case, mits goed onderhouden, in de basis dit framework gebruikt kunnen worden. Toch is de bruikbaarheid over een *x* aantal jaren afhankelijk van de doorontwikkeling, groei van de community en de interne technologiestack van CODE14.

**Meertaligheid**

Ik adviseer CODE14 om voor de statische content binnen een applicatie altijd een i18n module of alternatief voor meertaligheid toe te voegen. CODE14 denkt meer in trajecten dan in projecten, waarbij het handig is om voorbereid te zijn wanneer een klant uit wil breiden naar het buitenland.

### 6.1.3 Efficiëntie

#### 6.1.3.1 Flexibiliteit

Omdat CODE14 in het verleden niet heel consistent gebruik heeft gemaakt van één ontwikkeltechniek voor het realiseren van mobiele applicaties, kan er gesteld worden dat met overgang naar een hybride framework, er qua efficiëntie al een grote stap gezet wordt. Omdat er intern enorm veel ervaring is met VueJS en de learning curve die Ionic hieraan toevoegt, niet heel hoog is. Hierdoor zal de flexibele inzetbaarheid van medewerkers vergroten. Daar waar CODE14 in het verleden maar 4 of 5 ontwikkelaars had met kennis van Java en Swift, zal het bedrijf volgens resultaten uit de enquête nu bijna iedere huidige ontwikkelaar in moeten kunnen zetten op de ontwikkeling van een hybride applicatie d.m.v. Ionic, Vue en Capacitor.

#### 6.1.3.2 Tijdsbesparing

Het onderzoek heeft aangetoond dat er qua ontwikkeltijd veel winst te behalen valt. De totale ontwikkeltijd voor de native applicatie bedraagt 833 uur, waar de ontwikkeltijd voor de hybride applicatie op 328 uur is geschat. Omdat de applicatie nog niet geheel afgerond is (zie openstaande punten in hoofdstuk 5), kan de daadwerkelijke ontwikkeltijd nog niet volledig aangetoond worden. Maar o.b.v. de gemaakte progressie en openstaande punten kan dit een vrij accurate schatting zijn. De huidige ontwikkeltijd bedraagt namelijk 245 uur op 11-06-2021 en de openstaande punten op dit moment zijn in totaal geschat op

#### 6.1.3.2 Live-templates en code standaarden.

Na het bijwonen van een vergadering over efficiënter werken binnen front-end projecten, heb ik het advies gegeven om (meer) gebruik te maken van live-templates. Deze zorgen ervoor dat doormiddel van het typen van sleutelwoorden, er code gegeneerd kan worden, waar een ontwikkelaars anders meer tijd voor nodig had om volledig uit te typen.

Ook denk ik dat er meer eenheid, structuur en duidelijkheid geschept moet worden m.b.t. de algehele codestyling binnen CODE14. Er is apart nagedacht over codestyling van back-end en front-end, waarbij er rekening gehouden wordt met gebruikte kwaliteit tooling zoals de QA-checker. Maar tijdens de realisatiefase heb ik geconcludeerd dat er geen rekening is gehouden met de aansluiting welke de front-end maakt op de back-end.

#### 6.1.3.3 No-/Low-code

Mijn advies op het gebied van no-/low code is om vaker te oriënteren in het kader van tooling. Na de vondst van Overlay is er binnen het Development Board een pilot gestart met het platform Webflow (Webflow, z.d.). Helaas biedt dit platform niet mogelijkheid om code te exporteren of om mobiele applicaties te genereren, maar als het gaat om kleine webapplicaties waar snel een MVP (Minimal Viable Product) voor gerealiseerd moet worden, zie ik hier zeker potentie in.

Tijdens de realisatiefase is voor sommige componenten gebruik gemaakt van low-code plugin 'Overlay'. Het is hierbij van toepassing dat de product designers, welke werken met Figma, meer aandacht schenken aan de opbouw en naamgeving van de ontworpen componenten. Dit komt omdat Overlay de structuur en naamgeving meeneemt de gegenereerde componenten. Afhankelijk van de complexiteit en de te gebruiken HTML-attributen van een component is een gegenereerd Vue component vanuit VueJS wel of niet bruikbaar. Ik adviseer de Product Designers van CODE14 om meer te werken volgens de richtlijnen van Overlay om te zien of optimaal ontworpen UI componenten om te zetten zijn in een goed bruikbare Vue componenten.

#### 6.1.3.4 Aansluiting op bestaande concepten

Om efficiënter te werken is het van belang dat bestaande en lopende projecten up-to-date blijven, zowel met packages als met de gebruikte ontwikkeltechnieken. Om deze aansluiting te kunnen vinden zal met interne doorontwikkeling van onder andere de Nuxt Boilerplate, de ge-update werkwijze van codestyling direct toegepast moeten worden op bestaande en nieuwe projecten. Hierdoor hoeft een ontwikkelaar niet constant te schakelen tussen verschillende versies en kan hij/zij altijd dezelfde werkwijze behouden.

#### 6.1.3.5 Component Based Development

In het kader van component based development kan het van toegevoegde waarde zijn om specifieke eisen te stellen aan componenten welke hergebruikt kunnen worden. Zoals de naamgeving van de meegeleverde properties, het volledig uitwerken van een responsive design conform de standaarden van de Nuxt Boilerplate.

### 6.1.4 Onderhoud

Wanneer er vaker gekozen wordt voor één en dezelfde ontwikkeltechniek, zal de interne kennis over de ontwikkeltechniek stijgen. Waardoor er meer bewustzijn zal ontstaan over ontwikkelingen binnen de ontwikkeltechniek. Hierdoor kan er actie ondernomen worden, mochten er updates, bugs of verhinderingen bekend zijn binnen een ontwikkeltechniek. Uiteindelijk zal door de groeiende kennis, de complexiteit van de ontwikkelde applicatie dalen en zal er hogere kans zijn dat een applicatie na mate deze veroudert, gewoon blijft functioneren.

# 7. Reflectie

**Onderzoeksfase**

Tijdens mijn afstudeerperiode heb ik voor het eerst een onderzoeksrapport opgesteld. Ik heb een plan opgesteld waarin ik een duidelijke scope heb gedefinieerd en mijzelf hieraan heb gehouden. Op basis van het onderwerp heb ik verschillende stappen ondernomen om aan de benodigde informatie te komen, zoals het afnemen van interviews en een enquête. Ik heb literatuuronderzoek gedaan waarbij ik meerdere bronnen heb gebruikt om gevonden informatie te kunnen valideren. Ik ben zelf tevreden over hoe ik het onderzoek heb uitgevoerd.

Achteraf waren er dingen die ik anders had kunnen doen. Zoals, in de derde fase van het onderzoek heb ik een aantal belangrijke onderwerpen (criteria) opgeschreven d.m.v. de resultaten uit de enquête, interviews en gesprekken en eigen kennis over het onderwerp. Vervolgens heb ik nog een aantal criteria beschreven waarop Ionic het wint van Framework 7, terwijl sommige van deze criteria ook meegenomen hadden kunnen worden in Tabel 9 met de genoemde criteria. De impact van de keuze om dit wel of niet te doen is niet groot, aangezien dit niks zou veranderen aan de uitkomst, maar dit waren wel criteria die belangrijk genoeg waren om een impactvollere rol te spelen in de uitkomst.

**Realisatiefase**

De kwaliteit van de ontwikkelde hybride applicatie is dusdanig dat CODE14 besloten heeft dat zij deze, mits afgerond en getest, in productie willen nemen. Dit zal ter vervanging zijn voor de huidig bestaande native applicaties. Deze hybride applicatie is op het gebied van onderhoudsvriendelijkheid en efficiënte doorontwikkeling van hogere kwaliteit dan de al bestaande native applicaties. Mede hierdoor ben ik ervan overtuigd dat ik een mooi eindresultaat heb behaald, het geeft een trots gevoel om te weten dat mijn applicatie doorontwikkeld wordt om vervolgens in gebruik genomen te worden.

Ik heb tijdens de realisatiefase veel geleerd over het ontwikkelen van hybride applicaties, zo had ik nog nooit Push Notificaties ontwikkeld. Er werd van mij verwacht zelf beslissingen te maken over de structuur en vaak als mijn broncode gecontroleerd werd, kreeg ik positieve feedback over hoe de broncode opgebouwd en gestructureerd was.

Ook heb ik veel nagedacht over interactie met de gebruiker. Bepaalde transities en animaties welke de applicatie iets meer ‘premium’ doen aanvoelen, ook hier was de bedrijfsbegeleider erg over te spreken en vond dat dit ook een onderwerp was wat meer meegenomen moet in de ontwikkeling van front-end applicaties.

**Communicatie**

Tijdens de gehele afstudeerperiode heb ik initiatief genomen voor de communicatie tussen mij en de begeleiders, ik heb de stakeholders op de hoogte gehouden van de voortgang van het project. Ik heb zelf het initiatief genomen om een presentatie te geven voor het development board over mijn bevindingen en conclusies na afronding van de onderzoeksfase.

**Herzien en herschrijven**

Tijdens de realisatiefase heb ik de focus gelegd op het ontwikkelen van een applicatie, waarin makkelijk doorontwikkeld kan worden. Omdat ik tot op heden alleen heb gewerkt aan deze applicatie, heb ik zelf veel keuzes moeten maken. De opbouw van de applicatie met gebruik van de gekozen ontwikkelmethode 'Component Based Development' is uitgedacht op basis van het ontwerp van de VYBER-applicatie. Tijdens de ontwikkeling heb ik componenten zoals de modal (de popover na het indrukken van call-to-action knop) niet altijd even logisch opgebouwd.

Omdat ik dit component vaker moest gebruiken binnen de applicatie en ik een efficiëntere implementatiemethode zag, heb ik hier zonder feedback of kritiek op te hebben- ontvangen, de keuze gemaakt om de aanpak te wijzigen en de implementatie van het modal te herschrijven. Achteraf was dit een goede keus, omdat er nu meer flexibiliteit in de implementatie en mogelijkheden zit. Immers zie ik wel in, dat afhankelijk van het project, het beschikbare budget en de bijkomstigheid van een deadline dit in de praktijk niet altijd een optie kan zijn.

**Leerervaring**

Bij de start van mijn afstudeerperiode was ik onzeker over mijn kunnen. De opdracht stond me heel erg aan maar ik wist niet zo goed waar ik moest beginnen. Ik had geen ervaring met het uitvoeren van een onderzoek en het schrijven van een onderzoeks- of adviesrapport. Tijdens mijn afstudeerperiode heb ik een duidelijk plan opgesteld en ben mede hierdoor zekerder in mijn schoenen gaan staan. Tijdens de gehele afstudeerperiode heb ik beter leren omgaan met druk, ondanks dat dit voornamelijk druk vanuit mijzelf was omdat ik veel van mijzelf verwachtte. Uiteindelijk ben ik tevreden met de opgeleverde beroepsproducten en kan ik terugkijken op een leuke en vooral leerzame afstudeerperiode.

# 8. Bronnenlijst

*Adobe PhoneGap Build*. (2013, 1 januari). Adobe PhoneGap. https://build.phonegap.com/

Aparna. (2021, 12 maart). *The 10 Best Hybrid App Frameworks in 2021*. MobileAppAaily. https://www.mobileappdaily.com/best-hybrid-app-frameworks

Appcelerator. (z.d.). *Quick Start - Documentation & Guides - 2.0 - Appcelerator Wiki*. Wiki Appcelerator. Geraadpleegd op 24 maart 2021, van https://wiki.appcelerator.org/display/guides2/Quick+Start

*Appflow - Ionic - The Cross-Platform App Development Leader*. (z.d.). Ionic AppFlow. Geraadpleegd op 7 april 2021, van https://ionic.io/appflow

Bach, H. M. (2020, 4 september). *Differences Between Vue 2 And Vue 3 - JavaScript in Plain English*. Medium. https://javascript.plainenglish.io/differences-between-vue-2-and-vue-3-ee627e2c83a8

*Betekenis MVP*. (z.d.). Betekenis Definitie, MVP. Geraadpleegd op 12 juni 2021, van https://www.betekenis-definitie.nl/MVP

*Build Truly Native Mobile Apps with Vue | NativeScript*. (z.d.). NativeScript.Org. Geraadpleegd op 23 maart 2021, van https://nativescript.org/vue/

Capacitor Community FCM. (z.d.). *GitHub, Capacitor Community FCM*. GitHub, Capacitor Community FCM. Geraadpleegd op 28 mei 2021, van https://github.com/capacitor-community/fcm

*CapicatorJS integrating in NuxtJS*. (2020, 28 augustus). CapacitorJS. https://capacitorjs.com/blog/integrating-capacitorjs-plugins-with-nuxtjs

Centraal Bureau voor de Statistiek. (2019, 4 februari). *Meeste Nederlanders beschermen gegevens op smartphone*. https://www.cbs.nl/nl-nl/nieuws/2019/06/meeste-nederlanders-beschermen-gegevens-op-smartphone

Combell. (2020, 6 augustus). *Wat is Docker en waarom wil je ermee werken? » Combell*. Combell blog. https://www.combell.com/nl/blog/wat-is-docker/

Corona Labs. (z.d.). *Corona: Free Cross-Platform 2D Game Engine*. Geraadpleegd op 23 maart 2021, van https://coronalabs.com/

Crunchbase. (z.d.). *MobileAppDaily - Crunchbase Company Profile & Funding*. Geraadpleegd op 13 maart 2021, van https://www.crunchbase.com/organization/mobileappdaily

ElectronJS. (z.d.). *Electron | Build cross-platform desktop apps with JavaScript, HTML, and CSS.* Geraadpleegd op 29 maart 2021, van https://www.electronjs.org/

Encyclo. (z.d.-a). *User Experience*. Geraadpleegd op 17 februari 2021, van https://www.encyclo.nl/begrip/User_experience

Encyclo. (z.d.-b). *User interface - 8 definities - Encyclo*. Geraadpleegd op 17 februari 2021, van https://www.encyclo.nl/begrip/User_interface

Figma. (z.d.). *Figma: the collaborative interface design tool.* Geraadpleegd op 18 mei 2021, van https://www.figma.com/

Flutter. (z.d.). *Material Components widgets*. Geraadpleegd op 22 maart 2021, van https://flutter.dev/docs/development/ui/widgets/material

Flutter. (2021, 18 maart). *Supported platforms*. https://flutter.dev/docs/development/tools/sdk/release-notes/supported-platforms

Framework 7. (z.d.). *Framework7 Vue*. Framework 7, Vue Documentation. Geraadpleegd op 24 maart 2021, van https://framework7.io/vue/

GitHub. (z.d.). *bug: ionic vue main outlet re-rendering tabs when leaving and going back to tabs · Issue #22519 · ionic-team/ionic-framework*. GitHub, Ionic. Geraadpleegd op 11 juni 2021, van https://github.com/ionic-team/ionic-framework/issues/22519

GitHub, Axios. (z.d.). *Axios/axios*. Geraadpleegd op 27 mei 2021, van https://github.com/axios/axios

GitHub, Capacitor. (z.d.). *ionic-team/capacitor*. Geraadpleegd op 29 maart 2021, van https://github.com/ionic-team/capacitor

GitHub, Framework7. (z.d.). *GitHub Framework7*. GitHub - Framework7. Geraadpleegd op 29 maart 2021, van https://github.com/framework7io/framework7

GitHub, Ionic. (z.d.). *ionic-team/ionic-framework*. Geraadpleegd op 29 maart 2021, van https://github.com/ionic-team/ionic-framework

GitHub, NuxtJS. (z.d.). *GitHub NuxtJS*. NuxtJS GitHub. Geraadpleegd op 29 maart 2021, van https://github.com/nuxt/nuxt.js

GitHub, SecureLS. (z.d.). *softvar/secure-ls*. Geraadpleegd op 27 mei 2021, van https://github.com/softvar/secure-ls

Goyal, A. (2021, 7 januari). *Best Frameworks for Hybrid Mobile App Development in 2021*. Octal IT Solution. https://www.octalsoftware.com/blog/best-hybrid-app-framework

IMU Redactie. (z.d.). *Wat is een Unique Selling Point? (USP) | Uitleg & Voorbeelden*. IMU BV. Geraadpleegd op 9 februari 2021, van https://imu.nl/internet-marketing-kennisbank/begrippen/usp-unique-selling-point/

Ionic. (2021, 9 maart). *Ionic Framework - Ionic Documentation*. Ionic Docs. https://ionicframework.com/docs

Ionic, Capacitor. (z.d.). *Capacitor v3 documentation*. Capacitor. Geraadpleegd op 24 maart 2021, van https://capacitorjs.com/docs/v3

Ionic Capacitor. (z.d.). *Push Notifications*. Ionic Capacitor, Push Notifications. Geraadpleegd op 28 mei 2021, van https://capacitorjs.com/docs/apis/push-notifications

Javatpoint. (z.d.). *Ionic History - javatpoint*. Www.Javatpoint.Com. Geraadpleegd op 29 maart 2021, van https://www.javatpoint.com/ionic-history

jcesarmobile. (2019, 16 juli). *Assets in folder with underscore not loaded on Android · Issue #1750 · ionic-team/capacitor*. GitHub, Issue with NuxtJS Capacitor. https://github.com/ionic-team/capacitor/issues/1750#issuecomment-511785165

Lynch, M. (z.d.). *Ionic Article: Cordova vs Capacitor*. Ionic Framework. Geraadpleegd op 29 maart 2021, van https://ionic.io/resources/articles/capacitor-vs-cordova-modern-hybrid-app-development

Microsoft, Xamarin. (z.d.). *Xamarin documentation - Xamarin*. Microsoft Docs. Geraadpleegd op 24 maart 2021, van https://docs.microsoft.com/nl-nl/xamarin/

Mishra, P., Sachdeva, S., Kumar, A., & Kumar, A. (2019). Hybrid Application Development and Implementation. *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, 102–107. https://ieeexplore-ieee-org.saxion.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=8991279&isnumber=8991147

Mozilla Developers. (2019, 18 maart). *API - Woordenlijst | MDN*. https://developer.mozilla.org/nl/docs/Glossary/API

NuxtJS. (z.d.). *NuxtJS TypeScript support*. NuxtJS TypeScript Support. Geraadpleegd op 12 maart 2021, van https://typescript.nuxtjs.org/

*Nuxt.js - The Intuitive Vue Framework.* (z.d.). NuxtJS. Geraadpleegd op 18 mei 2021, van https://nuxtjs.org/

*Offshore IT Services | IT Software Solutions Company - Octal IT Solution*. (2021). Octal IT Solution LLP. https://www.octalsoftware.com/

Overlay tech. (z.d.). *Overlay | Design production ready code components*. Overlay. Geraadpleegd op 21 mei 2021, van https://overlay-tech.com/

Pastori, D. (2020, 17 augustus). *Packaging a NuxtJS app for iOS and Android with CapacitorJS*. Server Side Up. https://serversideup.net/packaging-a-nuxtjs-app-for-ios-and-android-with-capacitorjs/

React Native. (2021, 19 maart). *Introduction · React Native*. React Native Documentation. https://reactnative.dev/docs/getting-started

Sau, S. (2019, 8 augustus). *Java Programming, Basic Java programming - Google Play*. Java Programming, Basic Java programming - Google Play. https://play.google.com/books/reader?id=z86nDwAAQBAJ&hl=nl&pg=GBS.PA17

Shah, K. (2021, 8 maart). *Best 10 Cross-Platform App Frameworks to Consider in 2021*. Third Rock Techkno. https://www.thirdrocktechkno.com/blog/best-10-cross-platform-app-frameworks-to-consider-in-2021/

Solar2D. (z.d.). *Solar2D*. Solar2D - 2D Game Engine. Geraadpleegd op 23 maart 2021, van https://solar2d.com/

Storybook. (z.d.). *Storybook: UI component explorer for frontend developers*. Geraadpleegd op 18 mei 2021, van https://storybook.js.org/

ThirdRockTechKno. (z.d.). *About Us*. Third Rock Techkno. Geraadpleegd op 14 maart 2021, van https://www.thirdrocktechkno.com/about-us/

Traefik. (2016). *Traefik*. https://traefik.io/

*TypeScript Support Vue.js*. (z.d.). VueJS TypeScript Support. Geraadpleegd op 12 maart 2021, van https://vuejs.org/v2/guide/typescript.html

Vyber Cycles. (z.d.). *Vyber Cycles | MÃ©Ã©r kracht. MÃ©Ã©r actieradius*. Vyber. Geraadpleegd op 21 mei 2021, van https://www.vyber.com/

You, E. (2014). *Vue.js*. VueJS. https://vuejs.org/

# 9. Bijlagen

## 9.1 Bijlage A – Interview Leon Kosters

**Geïnterviewde:** Leon Kosters (L)

**Interviewer:** Jordy ten Den **(J)**

**Datum:** 02-03-2021, 16:15

**Locatie:** Slack Conference Call, Online

**J: Wat is je naam en wat is je functie?**

L: Leon Kosters, Product ontwerper bij CODE14 en front-end developer.

**J: Hoelang werk je al bij CODE14?**

L: Te lang, sinds 2015, dus 6 jaar.

**J: Je weet waar mijn onderzoek over gaat binnen CODE14, het efficiënter maken van het onderhouden en ontwikkelen van mobiele applicaties, met de nadruk op mobiele applicaties. Maar een aantal van de vragen die ik je straks ga stellen gaan ook over het ontwikkelen van applicaties in het algemeen binnen CODE14, dus applicatieontwikkeling. Een beetje een generieke vraag, maar in welke ontwikkeltechniek denk jij dat CODE14 zich specialiseert?**

L: In Laravel en Vue.

**J: Laravel en Vue, Webontwikkeltechnieken dus?**

L: Ja

**J: Dat is alleen Javascript en PHP?**

L: Ja, PHP Laravel Framework en Vue Javascript Framework.

**J: Juist, en zijn er nog meer specifieke programmeertalen waar veel kennis over is binnen CODE14?**

L: Nee, niet véél kennis. Ik denk dat er kennis is over C# (C Sharp), denk ik. Een aantal projecten ook met C# vooral Webo die 3D dingen doet, met 3D tekenprogramma's en die data doorsturen naar een webapplicatie.

J: **Weet jij toevallig hoeveel mensen daaraan werken, aan dat soort applicaties?**

L: 2 á 3, Rick en Kjell, misschien Rico, dat is het wel. Een half team, 1 project. Een gedeelte van één klant.

**J: Je benoemde net Vue en Laravel en dat zijn ontwikkeltechnieken die al lang worden gebruikt binnen CODE14, of dat relatief nieuw?**

L: Laravel vanaf het begin, zover ik weet. Dat is 2014, sinds 2015 sowieso sinds ik er ben. Front-end werd eerst ook binnen Laravel gemaakt, qua user interface met Laravel Blade zit je dan. Qua templating, daar gebruik je dan HTML, CSS en Javascript en Vue wordt denk ik ongeveer 4 à 3 jaar geleden gebruikt, voor het eerst. En dat is nu wel de standaard voor als je Javascript schrijft, en ook de enige standaard eigenlijk. Alle front-end applicaties zijn inmiddels in Nuxt, in ieder geval als ze nieuw zijn. Je hebt ook wat oudere projecten wat nog steeds in Laravel Blade is, of waar jQuery in zit of het is Gridsome of het is een echte Vue app, Vue CLI.

**J: Nuxt is gebaseerd op Vue?**

L: Ja.

**J: In hoeverre denk jij dat CODE14 kennis heeft van een framework zoals Vue of Nuxt?**

L: Ik denk Vue wel diepgaande kennis, ik denk wel elke webapplicatie die je maakt en dan de front-end kant, client side kant denk ik dat er echt wel een sowieso stuk of 5 zijn die echt heel veel kennis hebben, echt diepgaande kennis van Vue. Denk een Rick, Rico, Kjell, Robert die hebben wel denk ik het meeste kennis daarin, echt heel veel. Maar ik denk dat iedereen basiskennis heeft van Vue.

**J: Iedereen binnen CODE14? Stel er komt een nieuwe medewerker bij CODE14, dan is het eigenlijk één van de eerste dingen waarmee die persoon aan het werk wordt gezet, afhankelijk natuurlijk van zijn functie of rol binnen CODE14?**

L: Ja, als developer is het denk ik Laravel en Vue, is het een front-end developer dan alleen Vue, is het alleen backend dan Laravel. Mensen worden ook aangenomen, tenminste front-end developers worden aangenomen op Vue kennis, in ieder geval dat is wel een pré.

**J: Oké, duidelijk, Vue dus voornamelijk. Wat doet CODE14 volgens jou om zo efficiënt mogelijk te ontwikkelen aan (mobiele) applicaties?**

L: Zo efficiënt mogelijk, leesbare code denk ik dat de basis is voor CODE14 in het algemeen, zowel API als mobiele applicaties. Herbruikbare code. Specifiek mobiele applicaties denk ik web-technieken gebruiken, omdat je dan in één codebase direct voor desktop, Android en iOS kan ontwikkelen.

**J: Heb je een voorbeeld voor dat laatste wat je noemde? Vooral web-technieken gebruiken?**

L: Ja, Spotted! Pro is ontwikkeld in Ionic, dat is dan nog met Angular omdat Vue toch niet ondersteund werd. Wildvank is in Ionic geschreven, tijdsregistratie appje. Pigeon Racer uiteraard.

*(Onderbreking door Siri op de telefoon van Leon)*.

L: We hebben natuurlijk ook wel gewoon native apps geschreven, maar goed. Als je dan gaat vergelijken, welke vorm het minste tijd kost om te ontwikkelen, dan kost native gewoon veel langer. Ook andere kennis vereist hiervoor, er zijn ongeveer twee mensen die Java en twee mensen die Swift kunnen schrijven, drie misschien? Van de 36?

**J: Oké, dus dat is wel een reden om daarvan af te wijken? Tenminste, dat zeg jij dan?**

L: Het lijkt me zeker niet handig, nee. Dan ben je qua planning en beschikbaarheid, stel een klant wilt opschalen, zo beperkt dat je niet snel kan schakelen, maar ook voor op de langer termijn. Ik denk dat je in het begin geen snelheid kan maken, maar later ook niet, gezien je alles twee à drie keer moet doen. Omdat je een web app wil of eigenlijk desktop, Android en iOS, dus dan ben je gewoon drie keer code aan het schrijven.

**J: Oké, ik hoorde je net praten over Ionic. Je zei dat daar in het verleden zijn daar applicaties in gemaakt, doormiddel van Angular. Toevallig ben ik zelf bekend met dat framework, maar heb jij ervaring in het ontwikkelen in Ionic? Met wat voor taal dan ook?**

L: Ja, ik heb mobiele applicaties gemaakt in Ionic met Angular.

**J: Waarom is de keuze destijds gevallen op Ionic? In plaats van een andere ontwikkeltechniek?**

L: Omdat die kennis breder wordt gedragen als je het op basis van web technologie schrijft. Dan werd het gedragen door het hele team, en dan is het ook het hele team die eraan kan ontwikkelen. Anders was dat binnen een team maar één persoon, dus dan gaat je doorlooptijd gewoon heel erg omhoog. En onderhoudbaarheid, dus bijvoorbeeld AB Connect is dan ook een Ionic app, voor AB Texel. Dat is een Intranet, die willen ze continu door ontwikkelen, af en toe doe je daar 2 maand niks voor en daarna willen ze een nieuwe functionaliteit, bijvoorbeeld verlof aanvragen. Dan is het gewoon niet ideaal dat je weer opeens in Swift moet duiken, terwijl je 2 maanden lang PHP en Javascript hebt geschreven en voor een klein aantal uurtjes, dat is gewoon niet interessant je bent waarschijnlijk gewoon weer heel veel uur kwijt aan leertijd en je moet het dubbel of drie keer schrijven, want je doet het voor web want sommige chauffeurs die gebruiken een desktopcomputer want ze hebben geen smartphone. Je moet het voor iOS schrijven, dus swift. En je moet het voor Android schrijven dus dan is de keuze Ionic. Hierdoor kan je gewoon web technologie gebruiken, iedereen kent dat. En je hebt maar één codebase.

**J: Ja, duidelijk. Wat ik toevallig ook zelf weet over Ionic, Ionic is voornamelijk een framework die de native aspecten van de UI ondersteund voor mobiele applicaties, daar de achterliggende kern om het te bouwen of uitvoerbaar te maken op een mobiele applicatie is bijvoorbeeld Cordova of Capacitor, welke keuzes hebben jullie daarin gemaakt?**

L: Capacitor is vrij nieuw, dus die hebben we nog niet gebruikt. Maar wel interessant voor de toekomst, is nieuwer dan Cordova dus die doet het op een andere manier welke misschien beter past bij wat tegenwoordig gewoon modern is. Cordova gebruikt, volgens mij heb je ook

PhoneGap, maar dat is helemaal oud. Maar in feite zijn alle apps die hybride zijn, Ionic met onderwater Cordova binnen CODE14.

**J: Juist, maar dat heeft dus voornamelijk te maken met de tijd waarin de applicatie ontwikkeld is en niet specifiek omdat Cordova beter zou zijn dan Capacitor. Meer omdat Capacitor toen gewoon nog niet uit was?**

L: Ja, deze was nog in BETA volgens mij.

**J: Ben jij tijdens het ontwikkelen in Ionic of in hybride applicaties ansicht tegen problemen aangelopen, waarvan je denkt dat je daar tegen native ontwikkelen niet aan zal lopen, beperkingen misschien?**

L: Ik denk het nadeel is tegelijkertijd dat ook met web-technologieën werkt, dus de native API's wat bijvoorbeeld een Apple daarin aanbiedt is waarschijnlijk wat moeilijker te implementeren dan als je puur een iOS zal maken, dus als je de camera of bluetooth functionaliteit wilt gebruiken. Die API's zijn waarschijnlijk makkelijker te benaderen als je puur Swift schrijft, dan als je het in Cordova gaat doen. Kan een nadeel zijn, het voordeel is daarentegen wel dat je maar één keer die native functionaliteit hoeft te schrijven. Als je het native doet moet je het alsnog twee keer doen, dan zou het eventueel wel makkelijker zijn in Native, maar je moet het wel twee keer schrijven. Maar dat is denk ik het nadeel van dat je via web-technologieën, native telefoon functies aan wil roepen, dat is waarschijnlijk moeilijker dan dat je het direct via native wilt doen.

**J: De applicaties die je net benoemde zoals Spotted! Pro. Zijn dat applicaties die echt heel veel gebruik maken van de native API's? Hoe zou jij de algemene hybride opdracht voor CODE14 omschrijven, zie je dit als 'native heavy' of niet?**

L: Ik denk het niet eigenlijk, 80% van de tijd zal interface uitwerken zijn. Met web technologieën met de kennis die je hebt is het gewoon makkelijker. Bijvoorbeeld in Java of Swift is het best wel lastig, vooral één of twee jaar geleden was het sowieso een stuk lastiger met iOS om interfaces te maken. Ik heb ook wel ervaring met iOS apps maken en daar is interface gewoon heel lastig en complex. Met gewoon HTML en CSS is het veel makkelijker en dat is denk ik al 80% van de tijd, misschien daarnaast veel functies die veel met de CODE14 API, aangezien je dat met webapplicaties ook altijd doet is dat met Ionic veel meer standaard, veel meer wat je dagelijks doet, je dagelijkse bezigheid. Stel je zou dat in Swift doen, je bent in totaal met de gehele functionaliteit drie weken bezig en daarna doe je het weer drie maanden niet, dat is gewoon niet interessant. Je hebt hiervoor niet echt een specialisatie binnen het bedrijf voor puur Swift of Java, dat denk ik. Stel het zou native zijn, dan is het heel vaak het camera gebruiken en dat is gewoon goed te doen in hybride. De moeilijkere zijn misschien GPS of bluetooth, maar deze zijn ook al in Ionic Cordova te doen.

**J: Ja, de camera is relatief toegankelijk in Native API calls.**

L: Ja, dat denk ik wel. Als je de tijd vergelijkt, als je het en in Swift en in Java moet doen en je hebt er geen ervaring mee of het is al een tijd geleden. Als je die tijd bijhoudt en je zou dezelfde functionaliteiten ontwikkelen doormiddel van Cordova, dan zou die ontwikkeltijd misschien wel even lang zijn. Ik denk dat het weinig verschil maakt, maar je kan wel veel sneller stappen maken met interface uitwerken in Ionic en het is een pluspunt dat iedereen HTML, CSS-kennis heeft binnen CODE14.

**J: Over HTML, CSS en Javascript kennis gesproken. Hoe denk jij dat in dit soort talen efficiëntie slagen te behalen vallen binnen CODE14?**

L: Ik denk heel veel, vooral als je met Javascript werkt zoals in Vue. Dat je in componenten gaat werken en deze ergens centraal op kan slaan of gaat documenteren dat die kan hergebruiken. Stel je een component ontwikkelen voor een webapplicatie, zou je deze ook kunnen gebruiken voor een mobiele applicatie.

**J: In hoeverre denk je dat je hybride ontwikkeltechnieken zouden gaan 'clashen' met de UI die deze ontwikkeltechnieken zelf aanleveren? Zie je daar je problemen in?**

L: Ik denk niet zoveel, er zijn wel wat standaard interface elementen voor iOS en Android maar ik denk dat 90% gewoon interface is voor de content. Eigenlijk heb je waarschijnlijk een navigatiebalk die je wil gebruiken van de Apple guidelines en de Android guidelines omdat gebruikers die ook gewend zijn.

**J: Houden jullie daar rekening mee, als product ontwerpers met het uitwerken van een mobiele applicatie, de native styleguide van iOS en Android?**

L: Ja houden we wel rekening mee, soms ga je er een beetje tussen inzitten, omdat je voor één standaard dan kiest. Maar iOS en Android komen ook steeds meer bij elkaar te liggen. Als je een navigatie hebt bij Android zat dat eerst boven in het scherm met tabs of met een zogeheten hamburgermenu, dat is nu ook een navigatie die onder in je scherm zit en dat is bij iOS ook al zo. Op die manier kan je zelf een HTML en CSS een navigatie maken die ook aan de onderkant zit. Dan je hem zowel als mobiele applicatie gebruiker als PWA (Progressive Web Application). Daarin maakt het niet uit of het een web-applicatie is of een mobiele applicatie.

**J: Hoe zie jij de toekomst voor het ontwikkelen van mobiele applicaties binnen CODE14? En dan praat ik over een tijdsbestek van over 5 jaar? Stel je werkt over 5 jaar nog CODE14 en je wilt dan een mobiele applicatie ontwikkelen, hoe zou je dat proces van A tot Z zien?**

L: Stel je zou korte termijn, 1 à 2 jaar nemen. Binnen CODE14 zou het hybride zijn omdat we werken met web-technologieën. Doormiddel van frameworks die dat goed ondersteunen. Ik denk dat als je naar 5 jaar kijkt, dat de front-end en het design tool in elkaar zit, dus dat je doormiddel van een low code design tool de voorkant maakt, dat je daar ook wel kan invullen als iets een button is je klikt erop dat je een API-call, dus dan roep ik deze endpoint aan en ik stuur deze data

mee. Ik denk dat dat binnen 5 jaar gewoon low code is, of dat nou Apple is die dat aan gaat bieden of een externe partij die dat aan gaat bieden voor iOS of Android, of dat het een externe partij is die dat voor hybride gaat doen, dat ligt er maar net aan wie dat gaat doen. Wat het meest interessant is, denk ik. Als je kijkt naar Web Flow design, dat is low of no code, die doet dit al voor websites, zo zie ik dit over vijf jaar ook voor mobiele applicaties.

**J: Ja, want WebFlow biedt dit momenteel niet aan voor mobiele applicaties? Dat is echt puur website gerelateerd? Ben je bekent met No-/Low code platformen die dit aanbieden op dit moment?**

L: Ik ken wel een aantal tools, vooral Expo. Maar ik heb dat niet veel gebruikt, ook omdat ik het idee heb dat die tools er nog niet helemaal klaar voor zijn. Dat ze nog niet hebben ontdekt hoe ze het makkelijk kunnen houden maar toch schaalbaar en geen limitatie hebben.

**J: Te gelimiteerd momenteel?**

L: Ja, te gelimiteerd in de functionaliteit wat je er mee kunt. Dat denk ik voor nu, ik denk over twee jaar dat zo'n tool veel verder is en dat ze steeds meer pakken van wat mogelijk is.

**J: Je bent dan ook van mening, dat het onderzoek wat ik moet gaan doen dat ik meer moet richten op korte termijn dan op lange termijn? Voor efficiëntie slagen maar ook voor het kiezen van een ontwikkeltechniek voor de komende 'x' aantal jaren of maanden misschien?**

L: Ja, ik denk dat er een standaard moet komen voor de korte termijn. Dat deze meer prioriteit is om te onderzoeken omdat deze standaard er nog niet is, maar er wel veel winst mee kan halen. Eerst korte termijn en daarna kunnen wij kijken op lange termijn, maar met techniek is het zo dat het zo snel kan gaan dat je nu wel een onderzoek kan doen van wat gaan we de komende vijf jaar of over vijf jaar doen, maar dat is waarschijnlijk niet meer relevant. Je kan kijken welke tools er nu al zijn en wat deze kunnen, maar onderzoek naar hoe je dit kunt implementeren niet heel interessant is omdat het vijf jaar waarschijnlijk totaal anders is.

**J: Denk jij dat met de huidige staat, ik weet niet hoe diep je onderliggende kennis gaat over low-code platformen, dat een combinatie mogelijk zou zijn met hybride en low-code platformen?**

L: Ja, ik denk dat dat wel low code is. Dat je visueel programmeert maar dat je altijd de code kan openen om die te verrijken en de applicatie functioneel kan verrijken met echt code schrijven. Als je nu design tool Framer hebt, dat is gewoon een interface en prototype design tool. Elk component onderwater is gewoon React code en ik kan gewoon van design naar het tabje code en daar zie ik precies wat de onderliggende code is en daar kan ik mee interacteren. Dat is dan React, dan is de vraag of dat ooit in Vue komt. Misschien moeten we dan toch naar React gaan, omdat die tools zo goed worden dat je daar een hele efficiëntie slag kan maken.

**J: Ja want React Native is ook een hybride ontwikkeltechniek, heb je daar enige ervaring mee in programmeren?**

L: Nee, framer is puur React. Dus geen React Native, maar ik heb hiermee geen ervaring.

**J: Oké, dan heb ik nog één hoofdvraag. Waar moet een toekomstige hybride ontwikkeltechniek voor CODE14 aan voldoen, volgens jou? En dan doel ik meer op de native API-calls die mogelijk zijn.**

L: Voor korte termijn?

**J: Je noemde net al 1 á 2 jaar, dus laten we dat als maatstaaf nemen?**

L: Ik heb het idee dat je bij Cordova/Capacitor al het meeste wat je nodig hebt al ondersteund wordt, bijvoorbeeld bluetooth, fotocamera, fotogalerij en verschillende sensoren. En eigenlijk zijn sensoren al dingen die we eigenlijk nooit hoeven te gebruiken, we maken meer mobiele applicaties omdat dit gebruiksvriendelijker is dan altijd alleen maar een webapplicaties in een browser die alleen mobiel responsive is. Het is niet dat wij nou echt apps ontwikkelen die echt native moeten zijn voor bijvoorbeeld performance, misschien dat er native functies worden gebruikt maar geen functies die niet door hybride frameworks niet ondersteund worden. Ik kan eigenlijk geen functies noemen die niet ondersteund wordt, misschien Augmented Reality?

**J: Waar denk jij dan dat het verschil tussen een native- en hybride applicatie ligt?**

L: Als je een native applicatie met Swift maakt dat je een kleinere applicatie zou hebben, dus bijvoorbeeld 10MBs, terwijl deze zelfde in Ionic wel 100MB kan zijn? Denk niet dat dit heel veel uitmaakt, maar dat is een verschil. Performance kan een verschil zijn, ligt er maar net aan wat je doet, als je veel de GPU (Graphical Processing Unit) van een iPhone zou gebruiken, zoals bij fotobewerking en je wil dat real time laten zien dan zal dat waarschijnlijk wel invloed hebben. Maar als je een lijst van 10 objecten wilt tonen, zoals in Spotted! Pro je huisdieren laat zien, dan zal dat geen verschil gaan maken.

**J: Denk je dat de performance die de hedendaagse smartphones bieden een probleem vormt voor een modellen uit 2016?**

L: Ik denk zolang je je Javascript code goed schrijft, dat het heel veel verschil gaat maken. Misschien dat het 100 milliseconden scheelt, ik denk niet dat dat het argument is om dubbel zoveel uren in een native applicatie te stoppen.

**J: Dat waren eigenlijk alle vragen die ik op had geschreven, heb je zelf nog op- of aanmerkingen waarvan je denkt dat ik die echt mee moet nemen in het onderzoek? Mogen ook framework suggesties zijn.**

L: Suggestie op basis van CODE14 en de kennis die intern is, gewoon Ionic met Vue, dus web technologie met de focus op Vue. Omdat je gewoon continu webapplicaties aan het maken bent intern en dat zou op basis van mijn kennis de meest logische keuze zijn.

**J: Jullie werken al een behoorlijke tijd met Vue, hebben jullie dan ook al een efficiëntere wijze om code te hergebruiken?**

L: Nou ja, op basis van Vue componenten, er zijn natuurlijk al wel componenten die we kunnen herbruiken, of die makkelijk te herbruiken zijn natuurlijk een tweede. Ze zijn wel te kopiëren en plakken en dat is iets wat natuurlijk niet zo makkelijk gaat in Java of Swift.

**J: Top, meer heb ik voor nu niet, bedankt voor je tijd en de antwoorden!**

## 9.2 Bijlage B – Interview met Mathijs Pluimers

**Geïnterviewde:** Mathijs Pluimers, Mede-eigenaar CODE14 **(M)**

**Bijkomstige:** Harm-Jan Hazelhorst **(HJ)**

**Interviewer:** Jordy ten Den **(J)**

**Datum:** 04-03-2021, 13:45

**Hoofdvraag voor dit interview:** Wat zijn de voor- en nadelen voor CODE14 welke hybride ontwikkeling met zich meebrengt op business niveau?

**J: Ik had aan jou de vraag, wat zijn de voor- en nadelen die hybride ontwikkeling met zich meebrengt op business niveau?**

M: Hoe bedoel je dat?

**J: In het verleden heeft CODE14 native applicaties ontwikkeld en nu zoekt CODE14 naar een efficiëntere manier om mobiele applicaties te ontwikkelen of in ieder geval cross platform applicaties te ontwikkelen. Met mijn onderzoek moet ik dieper ingaan op hybride ontwikkeltechnieken, wat voor voor- en nadelen zitten eraan verbonden in verhouding met de manier waarop vroeger mobiele applicaties werden ontwikkeld?**

M: Kijk, als je kijkt naar de voordelen van hybride ten opzichte van native, als je het goed doet hoef je het in principe maar één keer te bouwen. Voor zowel iOS als Android, dat is het grote voordeel, tijd. Kosten besparen, tijd is geld.

**J: Zijn er daarnaast nog meer voor- of nadelen aan te verbinden? Als we kijken naar projecten, kunnen we daardoor misschien minder projecten aannemen, of minder complexe projecten omdat die meer vereisen?**

M: Dat is altijd de scheidingslijn tussen native gaan en/of hybride doen. Niemand weet precies tot hoe hoog je kan met hybride, zodat je geen problemen ervaart. Vaak waar je in native wel verder kan, in hoeverre kun je dat ook in hybride. Is er een performance issue, functionaliteiten die je kan bouwen, dat zijn allemaal dingen waarbij hybride en native verschil maken. Het verkoopt gewoon makkelijker, kijk als je 15 duizend euro of 30 duizend euro moet verkopen voor dezelfde applicatie, ja, tel uit je winst.

**J: Maar in principe zou je met native dan toch kijken naar pure omzet, dat zou dan toch interessanter zijn?**

M: Nee

**J: Waarom niet?**

M: Omdat mensen dat vaak niet willen betalen.

**J: Is dat in het verleden al wel eens voorgekomen? Dat daarom een opdracht niet door zou gaan?**

M: Wel vaker ja.

**J: Oké en waar liggen potentiele faalkosten?**

M: Van hybride?

**J: Ja.**

M: Rework, functionaliteiten die toch niet in hybride kunnen, allemaal workarounds die moeten zorgen dat hybride wel functioneert.

**J: In hoeverre denk je dat een hybride ontwikkeltechniek op de toekomstgericht moet zijn? Naar wat voor termijn kijken we dan?**

M: In principe moet je gewoon een platform kiezen die redelijk sustainable is, dus die wel vijf jaar mee moet kunnen.

**J: 5 jaar?**

M: Ja

**J: Denk je niet dat dat te lang is voor de technieken en de snelheid waarmee technieken vernieuwd worden?**

M: Nee, want als ik kijk naar Roaders, die is vier jaar geleden ontwikkeld? (Vraag richting Harm-Jan)

HJ: Roaders is van 2015.

M: 6 jaar geleden ontwikkeld en dat werkt erg goed.

**J: Maar dat is native?**

M: Ja, maargoed, die eisen heb je natuurlijk ook aan hybride.

HJ: Dat is bij Hybride inderdaad niet haalbaar.

**J: Nee, inderdaad. Dat is niet per se een conclusie die ik nu al kan trekken, maar ik denk dat 5 jaar een te lang termijn is. Als je bijvoorbeeld kijkt naar andere oplossingen, nieuwe innovatie op gebied van het ontwikkelen van mobiele- of cross platform applicaties, dan kan het heel goed zijn dat over twee jaar Low-/No code platformen al een stuk verder zijn. Dan zou je kunnen kijken of dat beter implementeerbaar is binnen de technologiestack van CODE14, en in een interview wat ik met Leon heb afgenomen heeft hij gesuggereerd dat de focus moet liggen op 2 jaar.**

M: Ja en daar ben ik het niet mee eens.

**J: Oké, maar zou je dat kunnen onderbouwen?**

M: Twee is een veel te korte termijn.

HJ: Je kunt geen enkele businesscase in twee jaar wegpoetsen. Een businesscase is minimaal 3 tot 5 jaar, en dat is al aan de korte kant.

M: Dus daarom zeg ik vijf jaar.

**J: Maar dan zou de doorontwikkeling van zo'n framework dus ook heel veel impact hebben op de business. En wat zijn dan bepaalde criteria die ik dan mee moet nemen aan de business kant, zijn er beloftes die zo'n framework moet doen?**

M: Zeker, maar daar kun je haast geen onderzoek naar doen. Ze kunnen wel beloftes doen, maar ze komen het toch niet na. Je weet niet waar een hybride platform heen gaat, dat is de ellende. Waarbij we vijf jaar geleden Ionic mega ruk vonden, echt slecht. Is dat nu wel aanzienlijk verbeterd. Het heeft ook met meerdere dingen te maken, hybride applicaties vreet ook meer van je RAM-geheugen van je device, die worden ook steeds sneller, dus, het is een combinatie van software en hardware ontwikkeling en het is lastig om daar een uitspraak over te doen.

**J: Oké, ik heb vanochtend deze vraag ook gesteld aan Harm-Jan, maar ik denk dat het interessant is om deze ook aan jou te stellen. Hoeveel mensen denk je dat er intern diepgaande kennis moeten hebben van de ontwikkeltechnieken die gebruikt worden in zo'n framework?**

M: Dat is afhankelijk van hoeveel mensen er in de operatie staan.

**J: Hoe bedoel je dat?**

M: Nou, als wij met 100 man zijn, waarvan er 30 programmeren. Of we zijn met 100 man, waarvan er 80 programmeren. Ja, dan kun je dus geen vast aantal houden. Hoevaak komt het voor? Hoevaak doe je dingen in zo'n hybride framework? Wij vinden dat iedereen verstand van Laravel moet hebben, want daar werken we vaak mee. Dus die maatstaaf ligt veel hoger dan een hybride framework, waar we niet zo veel mee werken.

**J: Ja, maar daar ligt toch ook al een duidelijke scheidingslijn tussen een front-end en een back-end developer en een full-stack developer bijvoorbeeld? Er wordt bij CODE14 verwacht dat een full-stack developer zowel Laravel als Vue kennis heeft, tenminste dat is een pre.**

M: Ja, want anders ben je geen full-stack.

**J: Ja, voor CODE14. Maar wordt er tijdens het sollicitatieproces van een frontend developer ook al verwacht dat hij Laravel kennis heeft?**

M: Nee

**J: En bij een backend developer wordt er niet verwacht dat hij Vue kennis heeft?**

M: Ook niet

**J: Denk je dat naast uren, dus geld, nog andere winst te behalen valt als je volledig over zou gaan naar een hybride ontwikkeltechniek voor mobiele projecten?**

M: Onder houdbaarheid, maar dat is ook terug te herleiden naar uren. Complexiteit wordt wel minder, dat is wel een belangrijke.

**J: En daardoor kunnen meer mensen werken aan een hybride ontwikkeltechniek?**

M: Juist.

**J: Om even terug te komen op die 5 jaar, ik had het hierbij voornamelijk over de keuzes in ontwikkeltechniek die bijvoorbeeld na twee jaar moet herzien.**

M: Ja, daar ben ik zeker voorstander van, maar de gekozen ontwikkeltechniek nu, moet minimaal 5 jaar goed kunnen draaien.

**J: Oké, dat is wat ik voor nu aan jou wou vragen! Bedankt voor je antwoorden en tijd.**

## 9.3 Bijlage C – Interview met Christian Lemmers

**Geïnterviewde:** Christian Lemmers (C)

**Interviewer:** Jordy ten Den **(J)**

**Datum en tijdsstip:** 08-03-2021 09:20

**Locatie:** CODE14 Headquarters, Walstraat 17, 7462BA Rijssen.

**J: Wat is je naam?**

C: Christian Lemmers

**J: Hoe oud ben je en wat is je functie?**

C: 28 jaar oud en ik ben product designer bij CODE14.

**J: En hoe lang werk je al bij CODE14?**

C: Bijna 2 jaar.

**J: Je hebt ook al een beetje meegekregen waar mijn onderzoek over gaat. Het efficiënter onderhouden en ontwikkelen van mobiele applicaties. Een aantal vragen die ik je ga stellen gaan ook over normale- of webapplicaties. Maar in welke ontwikkeltechniek denk jij dat CODE14 zich specialiseert?**

C: Op dit moment met name Vue. Dat kunnen dus ook de Nuxt en Gridsome projecten zijn, Vue is wel de hoofdtaal. Voorheen was het best wel Laravel, dat is nog steeds wel met API's. Maar met name mobiele apps zit je wel aan de Vue kant, Ionic, die kant op.

**J: Zijn er nog specifieke programmeertalen waar binnen CODE14 veel kennis van is?**

C: PHP, Javascript, komt op hetzelfde een beetje neer.

**J: Dus de kern van die frameworks?**

C: Ja.

**J: Je benoemde Vue en Laravel en dat zijn frameworks die al lang binnen CODE14 gebruikt worden, Wat ik van Leon begreep Laravel al sinds 2014, Vue? Weet jij hoelang dat al gebruikt wordt binnen CODE14?**

C: Ik weet dat we twee jaar terug alles nog in Laravel deden, maar wel Vue binnen Laravel gebruikten, maar daar begon het een beetje. Ik zou zeggen sinds twee jaar.

**J: Wat doet CODE14 volgens jou om zo efficiënt mogelijk te ontwikkelen aan mobiele- of webapplicaties?**

C: CODE14 is wel steeds meer aan het hameren op herbruikbaarheid van componenten. Wat je dus in Vue hebt, dat je veel opbouwt in componenten. We zijn steeds meer aan het kijken hoe we deze componenten over kunnen dragen naar nieuwe projecten.

**J: Daar hou jij als product ontwerper ook rekening mee in je designs?**

C: Ja

**J: Heb je zelf ervaring met het ontwikkelen van mobiele applicaties?**

C: Deels, ik doe zelf ook frontend werk. Ik heb wel eens aan wat mobiele applicaties gewerkt ja.

**J: Was dat dan native of hybride?**

C: Dat was wel echt hybrid, o.b.v. Ionic. Ionic samen met Angular. Bijv. de Spotted app en AB Track.

**J: Is dat lang geleden?**

C: Spotted laatst nog, AB Track een jaar geleden. Het is niet heel recentelijk dat ik een project heb moeten opzetten.

**J: Ben je tijdens de ontwikkeling van Spotted nog tegen problemen van het framework aangelopen?**

C: Niet zozeer het framework. Ionic Vue is nu volgens mij beschikbaar, niet meer in beta. Maar alle apps die tot nu toe met Ionic zijn gemaakt zijn met Angular. Dus in die zin was dat wel een omschakeling en lastig om daar een app in te bouwen.

**J: Ja, oké. Dus de beperking ligt hem voornamelijk bij de kennis die jij zelf hebt over de te gebruiken ontwikkeltechniek binnen Ionic?**

C: Ja, in dit geval wel ja.

**J: Heb je al eens geknutseld met Ionic Vue, nu het uit beta is?**

C: Nee, nog niet echt.

**J: Oké. Welke specifieke aspecten aan een ontwikkeltechniek of framework zijn volgens jou belangrijk voor CODE14?**

C: Het moet aansluiten op wat we al doen. Het moet zo dicht mogelijk bij de kern frameworks blijven. De schakeling tussen hybride- en webapplicaties moet heel makkelijk zijn.

**J: Vind jij het dan ook belangrijk dat de UI componenten die bijvoorbeeld Ionic levert, vind je dat handig om te hebben of een vereiste?**

C: Absoluut niet, wat we in Spotted! Pro hebben gedaan is dat we de componenten links hebben laten liggen en dat we vanaf scratch zelf componenten zijn gaan bouwen. En je gebruikt puur de native functionaliteiten daarin.

**J: Heb je dan ook wel eens geprobeerd zelf een 'back-button' te maken, echt de functionaliteiten die Ionic voor jou al op lost?**

C: In principe wel, je bouwt gewoon een nieuw div'je en knalt je eigen styling er overheen en vervolgens maak je een clickfunctie en maak je hier een back button van.

**J: Ja, er zit natuurlijk wel enigszins wat Javascript achter. Er zit best wat logica achter, het moet aan sluiten op ieder platform etc. Zou je niet zeggen dat het een efficiëntere wijze is om hiervoor wel Ionic te gebruiken?**

C: In het geval van efficiëntie zou ik zeggen ja, pak niks geen eigen styling en pak alleen componenten van Ionic zelf. Maar wat wij leveren aan klanten is een ervaring, in dit geval op een mobiele app met een eigen huisstijl. Een klant neemt geen genoegen met een applicatie die eruitziet als 10 andere apps.

**J: Hou je dan ook rekening met de verschillende design aspecten van de platformen, de native style guides?**

C: Niet heel veel, wel een beetje dat je op iPhone onder in het balkje hebt en op Android een echte backbutton hebt. Maar ik laat me er niet door beperken door er constant bij stil te staan.

**J: Oké, ben je bekend met Low-/No code platformen?**

C: Deels, al wel eens naar gekeken. Nog niet echt daadwerkelijk gebruikt. Een keer een prototype gemaakt, laatst voor invest hebben we AirTable gepakt, dat is ook een no-code platform. Ik heb een keer op WebFlow gekeken, je hebt voor Figma ook wel plugins die Vue componenten maken.

**J: Die plugins wat je zegt voor Figma, Figma wordt veel gebruikt om voor CODE14 ontwerpen te maken en Vue wordt veel gebruikt. Heb je al eens gekeken naar die plugins dan en wat voor code er komt uitrollen?**

C: Er zijn dus twee plugins die je zou kunnen gebruiken die inderdaad Vue code uitspugen. Daar moet ik zelf ook onderzoek naar doen, het klinkt in ieder geval interessant en ik denk dat het qua efficiëntie ook een stuk scheelt als je straks gewoon niet meer een stuk frontend op hoeft te pakken voor de simpele dingen zoals een design omzetten naar een Vue component.

**J: Ja, goed! Waar moet volgens jou een toekomstgerichte ontwikkeltechniek voor mobiele applicaties aan voldoen?**

C: Ja, nou het moet aansluiten op wat we nu doen. Met name Vue. En niet te beperkt in de ontwikkeltechniek. Nouja, Ionic, je werkt om de componenten heen omdat je ze gewoon niet fijn vindt. Eigenlijk zou je het liefst willen dat 90% van het platform gewoon fijn is om te gebruiken, dan zou Nuxt met Capacitor zou misschien heel chill zijn, je hebt geen beperkingen alleen extra functionaliteiten die je altijd nodig bent.

**J: Zoals? (Welke functies?)**

C: Nouja, je native functies welke je dan beschikbaar hebt. Zonder die standaard componenten, die je eigenlijk niet nodig bent. Hoe je het ook wendt of keert, binnen CODE14 maken we gewoon maatwerk software. Dus in die zin, als we maatwerk software maken zou ik niet zo snel grijpen naar een component al wel bestaat om dat vervolgens helemaal te gaan ombouwen, wat uiteindelijk misschien wel meer tijd kost, dan het van scratch opbouwen.

**J: Oké, duidelijk. Wat denk jij dat de voordelen zouden zijn voor het kiezen voor Nuxt met Capacitor i.p.v. Ionic met Capacitor.**

C: Ik denk dat je het ontwikkelproces een stuk simpeler maakt, in de zin van in het design werken wij dingen uit die niet standaard in Ionic zitten, die zien er anders uit. Dus wat je nu bijvoorbeeld krijgt is dat je Ionic pakt, en denkt “er zit ongeveer zo’n component in Ionic.”, dan pak je die en ga je die vervolgens ombouwen om het design te matchen. Je hebt veel meer vrijheid door niet werken met die componenten. Wil niet zeggen dat die componenten slecht zijn.

**J: Nee, oké. Maar stel dat je wel voor Ionic Vue kiest als in de zin van een backup, stel dat je bezig bent met een project waar 400 uur voor wordt geschat en we zitten met elkaar op 350 uur en we komen in tijdsnood, er zijn een aantal UI dingen niet af en bijvoorbeeld een drag-down-to-refresh. Echt van die aspecten waar veel meer javascript dan styling achter zit, binnen Ionic is dit binnen 5 minuten gefixt, in Nuxt zal je dit helemaal zelf moeten uitwerken.**

C: Ik denk inderdaad dat je het in de eerste 2 á 3 projecten misschien merkt dat het langer gaat duren, maar aan de andere kant zijn we binnen CODE14 aan het focussen op hergebruik en het werken met packages. Stel dat je het één keer voor een app hebt gebouwd, kijk je af hoe het in Ionic is gedaan en kopieer je het naar je Nuxt applicatie, werk je het daar netjes in uit en maar je er een package van. Volgende project heb je die package nodig, implementeer je dat in je package.

**J: Broncode van Ionic krijg je niet te zien hé.**

C: Nou, ik denk dat je daar nog wel achter kan komen.

**J: Ja, maar dan zit daar ook weer tijd in hé.**

C: Ja maar dat zeg ik, initieel ga je het merken aangezien het meer tijd zal kosten. Maar op langer termijn heb je straks een minder zware app die je sneller in elkaar kan zetten.

**J: Ja precies, daar doelde ik op met de vraag. Wat zijn de aspecten waarvoor je dan niet voor Ionic zou kiezen, het zou logisch. Zijn als zo’n framework zwaarder is. Maar geen idee of het zwaarder is dan Nuxt.**

C: Als ik zie hoe een Ionic applicatie functioneert is dat soms wel erg traag.

**J: Ja maar het compileren, kijk Ionic is eigenlijk niets meer dan een bootstrap hé.**

C: Ja, maar daar ben ik ook geen fan van, puur omdat je zoveel overhead hebt die je niet eens gebruikt. Of als je het wel gebruikt dan moet je er weer omheen schrijven om het zo te krijgen als je zelf wilt.

**J: In Ionic 5 met Vue is dat niet per se het geval, aangezien je in je pagina's een component moet implementeren en voordien worden ze niet in jouw broncode bijgeschreven.**

C: Dat is in ieder geval een ding dat wel scheelt. Maar nog steeds als je het naar je eigen smaak wilt hebben, moet je er overheen schrijven. Ik weet ook niet precies wat nou het beste is, ik weet alleen dat we hier nog niet hebben gewerkt met Nuxt en Capacitor en misschien kom je er dan wel achter dat het helemaal niet lekker werkt. Het is puur aannames van wat ik weet wat we wel met Ionic hebben gedaan.

**J: Gezien de focus op efficiëntie ligt en er is in het verleden met Ionic Angular gewerkt, met Ionic React niet, zie jij React als een optie?**

C: Ja, React zie ik als een 1 op 1 concurrent van Vue.

**J: Ja, oké. Maar intern? Binnen CODE14? Zou dit een kanshebber zijn?**

C: Ik denk nooit dat je ineens kan switchen binnen CODE14 naar React. Ik weet dat Robert als enige echt React kennis heeft en voor de rest denk ik niet dat je daar iemand voor kan overtuigen.

**J: Ik heb Robert al gesproken en die vond het zelf ook niet echt een goed idee. Stel dat ik prototypes zou maken, een Ionic Vue prototype en een Nuxt Capacitor prototype. En ik ga die applicaties uit eindelijk builden en ik zie dat hoe veel opslag zo'n applicatie in bezit neemt en dat scheelt niet veel. Dan zie ik dat niet echt als reden om niet voor Ionic Vue te kiezen. Zijn er specifieke dingen die in Nuxt zijn en niet in Vue waarom je dan toch naar Nuxt zou gaan en is dat dan de reden?**

C: Nee, ook niet zozeer. Onderwater komt het allemaal neer op Vue, over het algemeen maakt het niet zo veel uit. Eigenlijk de enige reden waarom je voor Nuxt met Capacitor zou kiezen is puur de flexibiliteit en gewoon dat je jezelf niet vastzet in een framework, ja Nuxt dan maar niet een Ionic Framework.

**J: Hoe goed is er destijds nagedacht over de keuze voor Nuxt?**

C: Nuxt is wel flink over nagedacht omdat in het begin werkten we met Laravel en Vue. Dus de Vue kennis was er. Toen zijn we overgegaan naar Gridsome omdat je daar heel goed statische websites mee kan bouwen en toen merkten we dat je met Gridsome tegen kleine beperkingen aanliep die je met bijvoorbeeld een Nuxt niet hebt. Gridsome is altijd statisch, Nuxt heb je de keuze. Dus met Nuxt heb je veel meer flexibiliteit en daar is wel over nagedacht om dan de Nuxt te pakken.

**J: Duidelijk, oké, dat was het wel, bedankt voor de antwoorden!**

## 9.4 Bijlage D – Toelichting van enquête vragen en antwoorden.

Persoonsgegevens

1) ***Wat is je voor- en achternaam?***

**Toelichting**
Gezien het geen anonieme enquête betreft en er een duidelijk beeld nodig is van de respondenten, is er om een voor- en achternaam gevraagd.

2) ***Datum van invullen.***

**Toelichting**
Aantonen in welk tijdsbestek deze enquête is ingevuld.

3) ***Wat is je functie binnen CODE14? (Meerkeuze vraag)***

*Opties*

- MBO
- HBO
- Universitair
- *Anders, namelijk:*

**Toelichting**
Deze vraag is gesteld om het opleidingsniveau van de respondenten in kaart te brengen.

4) ***Hoelang werk je al bij CODE14? (Meerkeuze vraag)***

*Opties*

- 1-6 maand(en)
- 6 tot 12 maanden
- 1 tot 2 jaar
- Langer dan 2 jaar

**Toelichting**
Gezien CODE14 bijna 7 jaar bestaat, is het van toepassing om te weten hoelang iemand onderdeel uitmaakt van het ontwikkelteam en hierom mogelijk onderdeel heeft uitgemaakt van de keuzes die in het verleden gemaakt zijn in de technologiestack.

5) ***Hoelang programmeer je al? (Meerkeuze vraag)***

*Opties*

- Korter dan 1 jaar
- 1 tot 5 jaar

- Langer dan 5 jaar

**Toelichting**
Deze vraag brengt in kaart hoeveel ervaring een respondent heeft in het daadwerkelijke programmeren. Daarnaast kan dit enige invloed uitoefenen op de antwoorden welke worden gegeven op de latere vragen in de enquête.

***6) Heb je wel eens een mobiele applicatie ontwikkeld? (Meerkeuze vraag)***

*Opties*

- Ja
- Nee
- Niet noemenswaardig

**Toelichting**
De antwoorden van deze vraag zijn van toepassing om te valideren of de respondent daadwerkelijk kennis heeft over het onderwerp waar komende vragen over zullen gaan.

### Kennisgebieden

***7) Met welke programmeertalen heb je ervaring? (Selectievakjes)***

*Opties*

- JavaScript
- Java
- C
- HTML
- CSS
- Python
- Cobol
- PHP
- C++
- C#
- Visual Basic .NET
- Swift
- TypeScript
- Kotlin

**Toelichting**

Deze vraag toont aan van welke programmeertalen de respondent kennis heeft. Niet iedere programmeertaal in deze lijst is relevant voor het vraagstuk, aangezien niet iedere bovengenoemde programmeertaal gebruikt wordt in een hybride ontwikkeltechniek, maar dit kan wel aantonen hoe breed de kennis van een respondent op gebied van programmeertalen is.

8) ***Heb je ervaring met één of meerdere van de volgende front-end frameworks? (Selectievakjes)***

***Opties***

- VueJS
- ReactJS
- AngularJS

**Toelichting**
Deze drie frameworks worden zijn het meest terug te vinden in hedendaagse hybride ontwikkeltechnieken. Gezien de voorkennis welke is opgedaan over CODE14, doormiddel van gesprekken met medewerkers en eigenaren. En gezien hun verleden met hybride ontwikkeltechnieken maar ook web-ontwikkelingstechnieken is de kans groot dat een respondent met minimaal één van deze frameworks in aanraking is geweest.

9) ***Hoe denk jij dat CODE14 efficiënter kan ontwikkelen als het gaat om het ontwikkelen van hybride applicaties? (Open vraag)***

**Toelichting**
Gezien dit een open vraag is, wordt de ruimte aan de respondent gegeven om zich te uiten over het verleden van CODE14 met hybride ontwikkeling, dit is van belang om te kunnen zien of de respondent tevreden is over de keuzes die er in het verleden zijn gemaakt.

***Ben je tevreden met de huidige front-end technologiestack van CODE14?***

*Opties*

- Ja
- Nee
- *Anders, namelijk:*

**Toelichting**
Hierdoor Er is hierbij bewust de keuze gemaakt om de vraag semi-gesloten te stellen, toch wil ik de respondent de mogelijkheid bieden om zich toe te lichten over eventuele twijfel tussen de antwoordmogelijkheden.

Even terug in de tijd

De vragen van het onderdeel ‘even terug in de tijd’ zijn alleen gesteld aan mensen welke ervaring hebben met het ontwikkelen van mobiele applicaties.

***10) Hoelang is het geleden dat je voor het laatst aan een mobiele applicatie hebt gewerkt? (Meerkeuze vraag)***

*Opties*

- 1 maand geleden
- 2 tot 4 maanden geleden
- 6 maanden geleden
- Langer dan 6 maanden geleden

**Toelichting**
Aangezien de ontwikkeltechnieken meestal niet stil blijven staan, is het van toepassing dat de mening welke de respondent in het verleden heeft gevormd over de destijds gebruikte ontwikkeltechniek, nog van toepassing is.

***11) Was de realisatie van deze mobiele applicatie in opdracht van CODE14? (Meerkeuze vraag)***

*Opties*

- *Ja*
- *Nee*

**Toelichting**
Er is in de context analyse naar voren gekomen welke ontwikkeltechnieken CODE14 heeft gebruikt voor het ontwikkelen van mobiele applicaties. Daarmee kan er indien nodig onderzoek gedaan worden naar de staat van die specifieke ontwikkeltechniek, op dat moment.

**12) Welke ontwikkel techniek is er destijds gebruikt voor de ontwikkeling van deze mobiele applicatie(s)? (Selectievakjes)**

*Opties*

- Native Ontwikkeling (iOS)
- Native Ontwikkeling (Android)
- Web-ontwikkelingstechnieken (Hybride)

**Toelichting**
Dit toont aan of er kennis is over hybride- en native ontwikkeling.

13) ***Met welke hybride ontwikkel framework heb jij de meeste ervaring? (Meerkeuze vraag)***

*Opties*

- Ionic
- React Native
- Flutter
- Native Script
- Ik heb geen ervaring met hybride frameworks
- *Anders, namelijk:*

**Toelichting**
Een respondent kan ervaring hebben met meer frameworks, immers willen we met het antwoord op deze vraag alleen naar voren halen met welke hij of zij de meeste ervaring heeft.

14) ***Veel van de hybride frameworks bieden verschillende UI (User Interface) componenten aan, is dit een vereiste voor een toekomstig hybride ontwikkelframework? (Meerkeuze vraag)***

*Opties*

- Ja, dit is een vereiste
- Nee, dit is geen vereiste

**Toelichting**
Uit het voorafgaande interview met Leon Kosters & Christian Lemmers werden frameworks zoals VueJS en NuxtJS veel genoemd. Gezien deze frameworks uit zichzelf geen UI elementen aanbieden, kan dit uitsluiten hoe belangrijk de medewerkers van CODE14 dit aspect vinden bij een hybride ontwikkelframework.

15) ***Is het volgens jou van belang dat een toekomstig hybride framework aansluiting heeft op de kennisgebieden van de medewerkers?***

*Opties*

- Ja
- Nee
- *Anders, namelijk:*

**Toelichting**
Deze vraag kan enigszins aantonen hoe de medewerkers denken over eventuele veranderingen in de technologiestack, het leerproces wat dit met zich meebrengt en hoe ze denken over de huidige technieken welke ze gebruiken.

16) ***Welke ontwikkel technieken denk jij dat aansluiten op de technologiestack en kennis binnen CODE14 als het gaat om hybride applicatieontwikkeling? (Open vraag, optioneel)***

**Toelichting**
Hier krijgt de respondent de mogelijkheid om kennis te uiten over hybride ontwikkeltechnieken en kan er een goed beeld gevormd worden in welke richting er gedacht wordt door de ontwikkelaars. Omdat deze vraag enigszins voorkennis vereist over hybride ontwikkeltechnieken is deze optioneel.

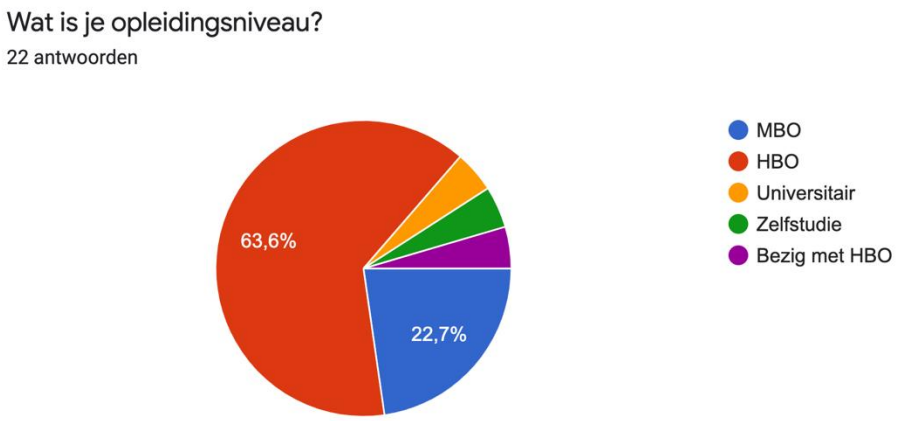
17) ***Waar denk jij dat efficiëntieslagen te behalen vallen binnen het ontwikkelen van hybride applicaties? (Open vraag, optioneel)***

**Toelichting**
Hierin krijgen ook de respondenten zonder voorkennis over hybride ontwikkeltechnieken de mogelijkheid om eventueel in te spelen op onderliggende technieken welke veelal gebruikt worden in hybride applicatieontwikkeling, zoals: HTML, (S)CSS, Javascript.

### 9.4.1 Uitkomsten medewerkers enquête

De enquête is in totaal 22 keer ingevuld, hiermee is het door grotendeels van de ontwikkelaars van CODE14 ingevuld.

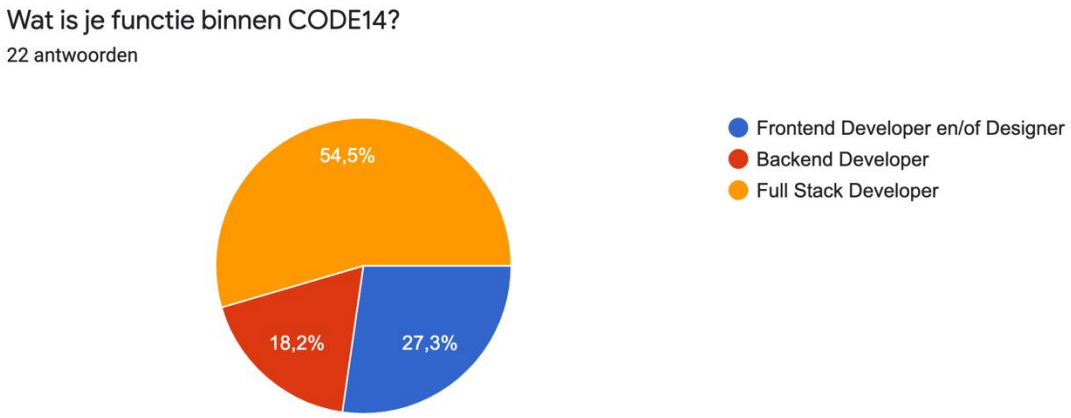
**Wanneer is deze enquête ingevuld?**
Tussen Dinsdag 09-03-2021 en Vrijdag 12-03-2021




*Figuur 18 - Diagram met antwoorden op het Formulier. Titel van de vraag: Wat is je opleidingsniveau?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

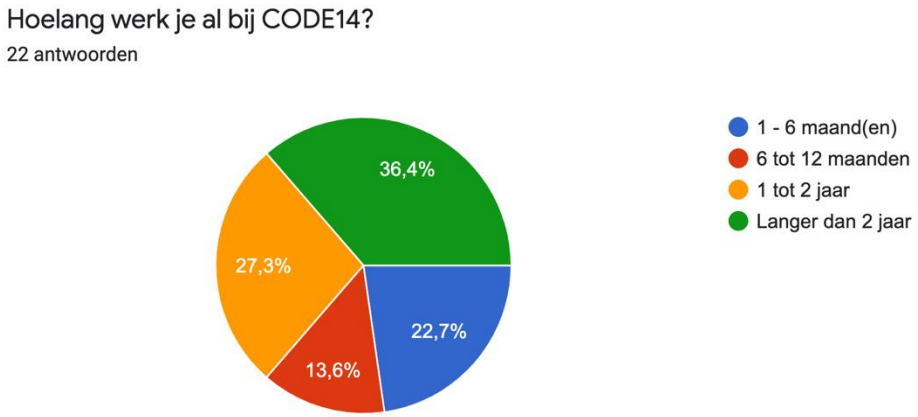
Grotendeels van de medewerkers bij CODE14 zijn HBO geschoold of zijn hiermee bezig, daarnaast zijn er ook een aantal afkomstig van het MBO. Er is één ontwikkelaar universitair afgestudeerd en een enkeling heeft doormiddel van zelfstudie het programmeer vak aangeleerd.




*Figuur 19 - Diagram met antwoorden op het Formulier. Titel van de vraag: Wat is je functie binnen CODE14?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

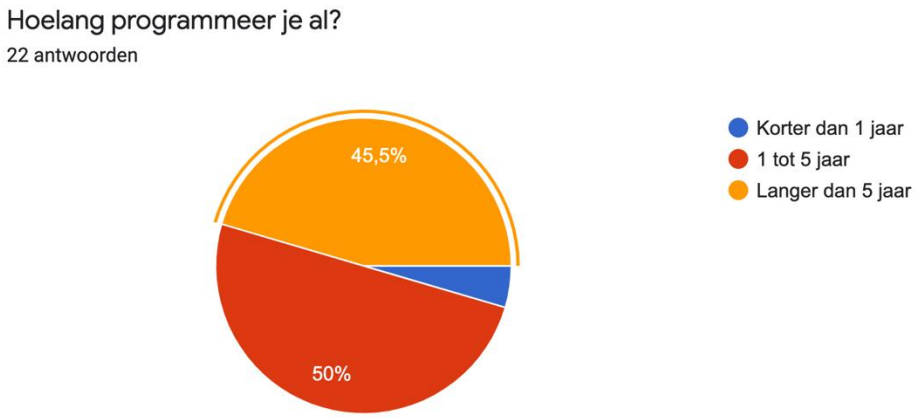
Meer dan de helft van de ontwikkelaars binnen CODE14 fungeert als Full-Stack Developer (12 medewerkers). Daarnaast zijn er een aantal welke alleen backend taken uitvoert (4 medewerkers) en zijn er in totaal 6 medewerkers welke alleen front-end en/of design taken uitvoeren.




*Figuur 20 - Diagram met antwoorden op het Formulier. Titel van de vraag: Hoelang werk je al bij CODE14?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

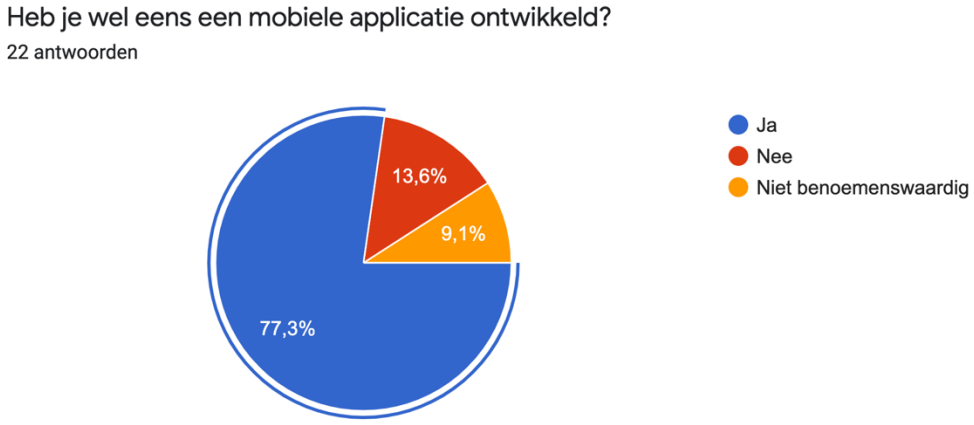
In totaal zijn er 8 medewerkers welke langer dan twee jaar werkzaam zijn bij CODE14, dit wordt gevolgd door 6 medewerkers welke 1 tot 2 jaar werk verrichten bij CODE14. 5 medewerkers zijn relatief nieuw binnen het bedrijf en zijn 1 tot 6 maand(en) werkzaam binnen het bedrijf en een drietal is tussen de 6 maanden en een jaar werkzaam binnen het bedrijf.




*Figuur 21 - Diagram met antwoorden op het Formulier. Titel van de vraag: Hoelang programmeer je al?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

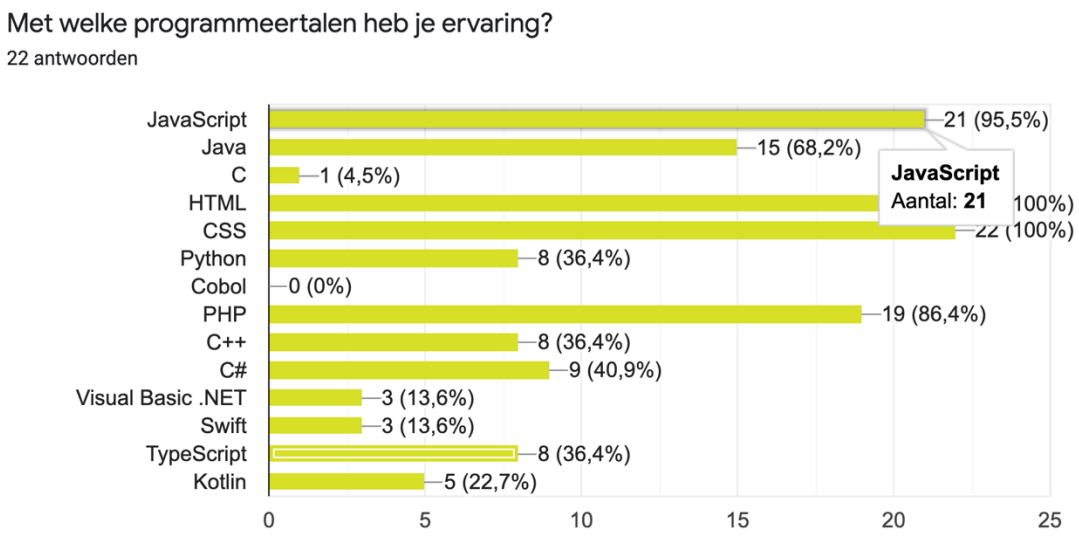
Het gaat enigszins gelijk op als het aankomt op de tijd dat de respondenten al programmeren. 11 van de respondenten programmeert al langer dan 5 jaar, gevolgd door een tiental welke tussen de 1 en 5 jaar programmeerervaring heeft en één welke relatief nieuw is in het vakgebied.




*Figuur 22 - Diagram met antwoorden op het Formulier. Titel van de vraag: Heb je wel eens een mobiele applicatie ontwikkeld? Aantal antwoorden: 22 antwoorden.*

**Toelichting**

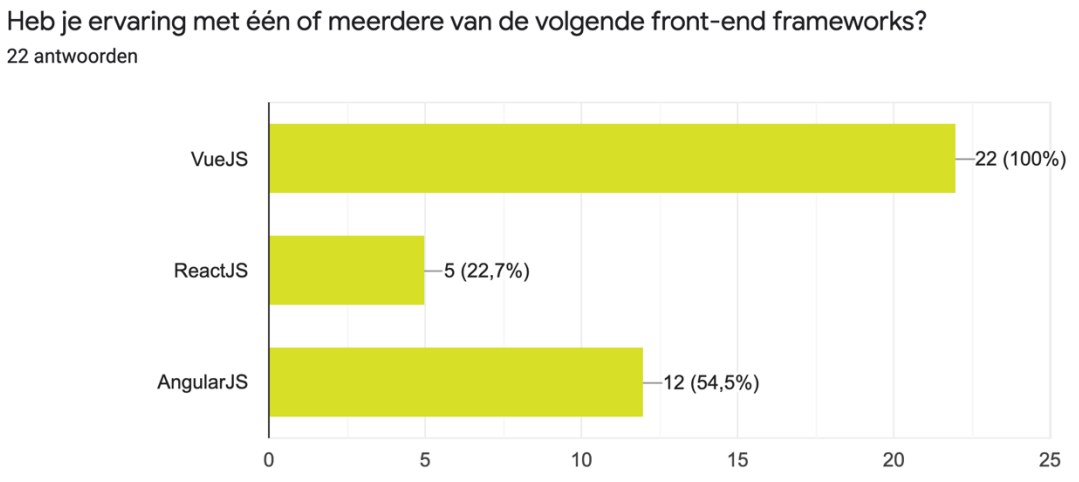
Van de 22 respondenten hebben 17 al eens een mobiele applicaties ontwikkeld, daarnaast zijn er twee welke hun bijdrage niet noemenswaardig vinden aan de ontwikkelde applicatie en zijn er 3 welke nog nooit een mobiele applicatie hebben ontwikkeld.




*Figuur 23 - Diagram met antwoorden op het Formulier. Titel van de vraag: Met welke programmeertalen heb je ervaring?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

In figuur 23 is duidelijk te zien waar de meeste ervaring qua programmeertalen ligt. Bijna alle respondenten hebben ervaring met JavaScript, op één na. Daarnaast heeft iedere respondent ervaring met HTML en CSS, gevolgd door 19 met PHP-ervaring, wat natuurlijk aansluit op de technologiestack, gezien Laravel het meest gekozen backend framework is voor CODE14, dit is dan ook een framework welke zich baseert op PHP.




*Figuur 24 - Diagram met antwoorden op het Formulier. Titel van de vraag: Heb je ervaring met één of meerdere van de volgende front-end frameworks?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

Iedere respondent heeft ervaring met VueJS. De verwachting na de afname van de interviews was dat dit het framework zou zijn waarin de meeste respondenten ervaring zouden hebben. De opvolger in dit diagram is AngularJS, wat ook een verwachte uitkomst is. Gezien er minimaal 3 projecten van CODE14 draaien op Ionic i.c.m. Angular. Er is relatief weinig kennis over ReactJS, met maar 5 respondenten welke aangeven hier ervaring mee te hebben.

**Hoe denk jij dat CODE14 efficiënter kan ontwikkelen als het gaat om het ontwikkelen van hybride applicaties?**

| |
|---|
| Ik denk dat we meer moeten focussen op native runtimes zoals Capacitor. Bij CODE14 hebben we heel veel JS kennis in huis door het ontwikkelen van web-applicaties. Door dus de native API's te gebruiken van een Capacitor kunnen we ook de huidige kennis gebruiken voor mobiele applicaties. |
| Het gebruiken van Vue bij het bouwen van apps. |
| Zolang we bij onze bekende stack kunnen blijven. Met Nuxt/Vue bijvoorbeeld. Het zou enorm helpen als we daar hybride apps van kunnen bouwen.<br><br>Een goede cursus in TypeScript zou ook niet verkeerd zijn. Daar vang je een hele hoop bugs mee af.<br><br>Als laatste: testen schrijven voor je frontend. |
| Als er gewerkt gaat worden met één framework waarmee het mogelijk is om cross-platform applicaties te ontwikkelen lijkt het mij verstandig dat er gekeken wordt naar herbruikbare code. Hiermee bedoel ik dat het verstandig is dat bepaalde modules / componenten opgezet worden zodat deze makkelijk herbruikt kunnen worden in hetzelfde / een ander project. Op het moment dat er redelijk snel een nieuwe applicatie ontwikkeld kan worden met herbruikbare code ben je efficient bezig. |

*Figuur 25 - Eerste vier antwoorden. Hoe denk jij dat CODE14 efficiënter kan ontwikkelen als het gaat om het ontwikkelen van hybride applicaties?*

| |
|---|
| ionic vue gebruiken inplaats van ionic angular |
| Meer onderzoek naar bestaande frameworks, als hierop (app dev) gefocust dient te worden moeten we niet bang zijn hier een niet taal / framework te gaan gebruiken / leren. |
| Geen idee, ik ben maar een simpele afstudeerder |
| Vue / Capacitor |
| Heb hier zelf geen ervaring mee |
| Templating |
| Ik hou mij niet zo bezig met hybride apps. Maar wat maakt het niet efficient? Moet er misschien meer gestandaardiseerd worden net zoals we bij onze web stack standaard laravel+vue doen en daarbinnen ook standaard manieren hebben om dingen te doen? |

*Figuur 26 - Antwoord 5 tot 10. Hoe denk jij dat CODE14 efficiënter kan ontwikkelen als het gaat om het ontwikkelen van hybride applicaties*

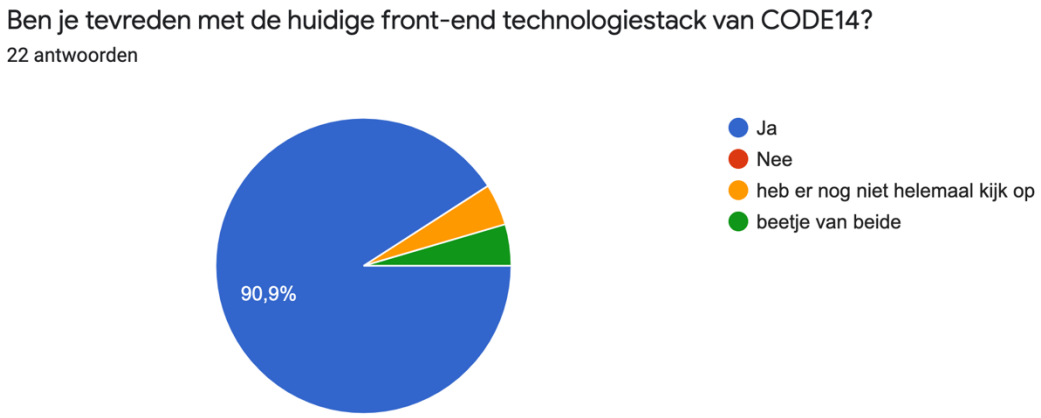
| |
|---|
| CODE14 kan (nog) meer gebruik maken van herbruikbare componenten. Daarnaast gebruiken we binnen CODE14 vaak Ionic i.s.m. Angular echter gebruiken we voor alle andere applicaties Vue. Een overstap naar Ionic met Vue (of zelfs Capacitor met Vue) zou al een hele verbetering in efficiëntie zijn. |
| stick to one language |
| Low code tools, van design naar front-end code. Werken met standaard componenten (eigen of goede libraries (ionic vue). Werken met standaard packages (front-end en back-end). Goede OTAP inrichting met automatische deploys (ook naar app stores). Nuxt gebruiken als standaard framework, omdat de standaard nu is voor webapps. Capacitor gebruiken voor de apps te builden. |
| Dichter bij de bekende stack blijven. Meer mobiele applicaties maken zodat de kennis ook vergroot wordt. Geen junioren laten werken met een stack die over 't algemeen niet heel erg goed bekend is binnen CODE14. |
| VueJS gebruiken in Ionic als dit ooit stabiel is, PWAs adviseren |

*Figuur 27 - Antwoord 11 tot 16. Hoe denk jij dat CODE14 efficiënter kan ontwikkelen als het gaat om het ontwikkelen van hybride applicaties*

**Toelichting**

In bovenstaande Figuren (25, 26 en 27) is te zien dat van de 16 antwoorden, **VueJS** in totaal 6 keer aangeraden door de medewerkers, **Ionic** is in totaal 3 keer benoemd, waarbij éénmaal i.c.m. VueJS. **NuxtJS** is in totaal 2 keer benoemd en **Capacitor** is in totaal 5 keer genoemd. **AngularJS** is tweemaal benoemd maar niet in positieve zin, deze werd voornamelijk benoemd omdat dit framework momenteel gebruikt wordt i.c.m. Ionic en dit niet aansluit op de huidige stack.

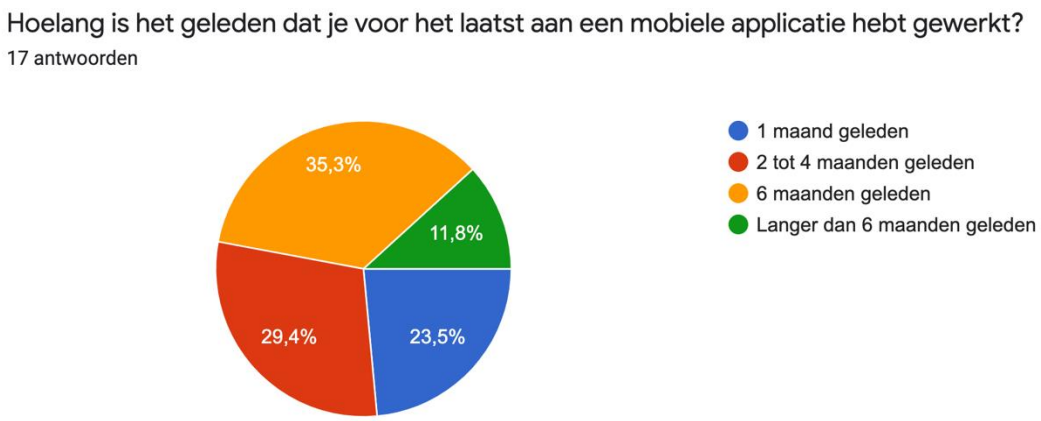
Verder wordt er benoemd dat er meer kennis benodigd over **TypeScript**, hierdoor zouden er aan de voorkant al veel problemen kunnen worden afgevangen, gezien over het algemeen nu overal nog Javascript voor gebruikt wordt. De geïntegreerde frameworks in de technologiestack bieden al wel de mogelijkheid om **TypeScript** (NuxtJS, Vue) te gebruiken, (NuxtJS, z.d.), (TypeScript Support Vue.js, z.d.). Daarnaast wordt er ook aangegeven dat **low code tooling** van pas zou kunnen komen en dat er meer bij één programmeertaal gebleven zou moeten worden. **Herbruikbare code** is een herhaaldelijk onderwerp, ook wordt dit meermaals aangegeven in de afgenomen interviews en mits toepasbaar, kan dit zeker een stap naar efficiëntere workflow zijn.




*Figuur 28 - Diagram met antwoorden op het Formulier. Titel van de vraag: Ben je tevreden met de huidige front-end technologiestack van CODE14?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

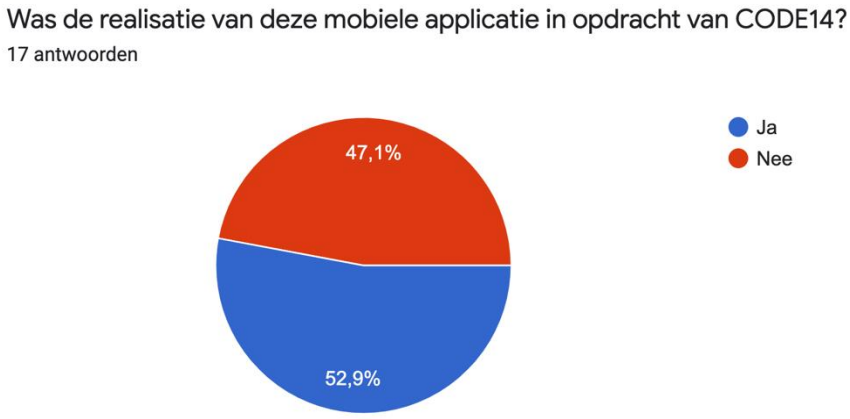
Het overgrote deel van de respondenten is tevreden met de huidige front-end technologiestack van CODE14, alle nieuwe projecten worden nu opgezet in NuxtJS, wat gebaseerd is op VueJS.




*Figuur 29 - Diagram met antwoorden op het Formulier. Titel van de vraag: Hoelang is het geleden dat je voor het laatst aan een mobiele applicatie hebt gewerkt?. Aantal antwoorden: 17 antwoorden.*

**Toelichting**

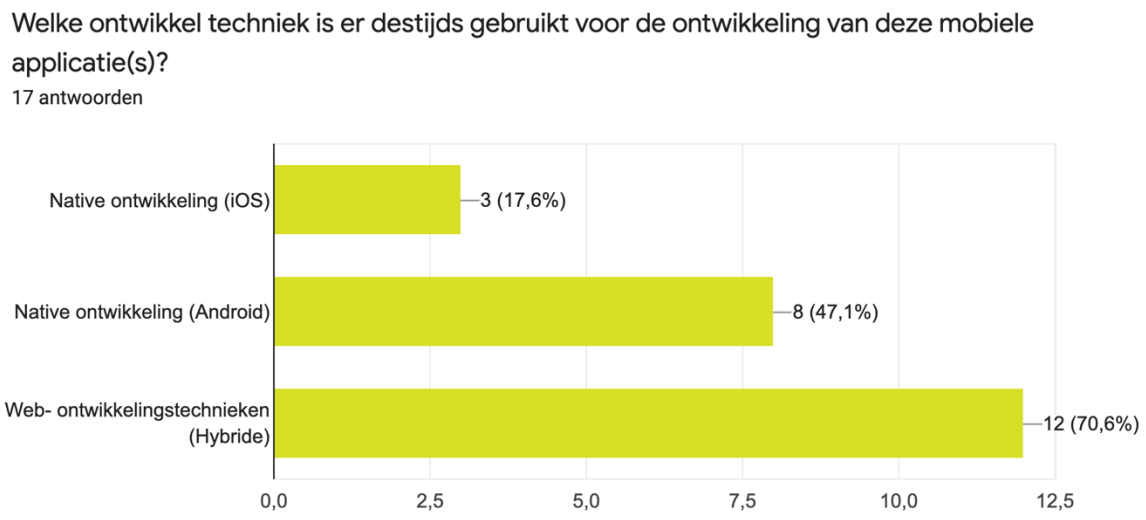
In deze antwoorden is wat meer verdeeldheid te zien, met mobiele applicaties kan hierin ook native ontwikkeling bedoeld worden. 4 mensen hebben in de afgelopen maand nog gewerkt aan een mobiele applicatie, 5 mensen tussen de 2 en 4 maand geleden, 6 mensen 6 maanden geleden en 2 mensen langer dan 6 maanden geleden.




*Figuur 30 - Diagram met antwoorden op het Formulier. Titel van de vraag: Was de realisatie van deze mobiele applicatie in opdracht van CODE14?. Aantal antwoorden: 17 antwoorden.*

**Toelichting**

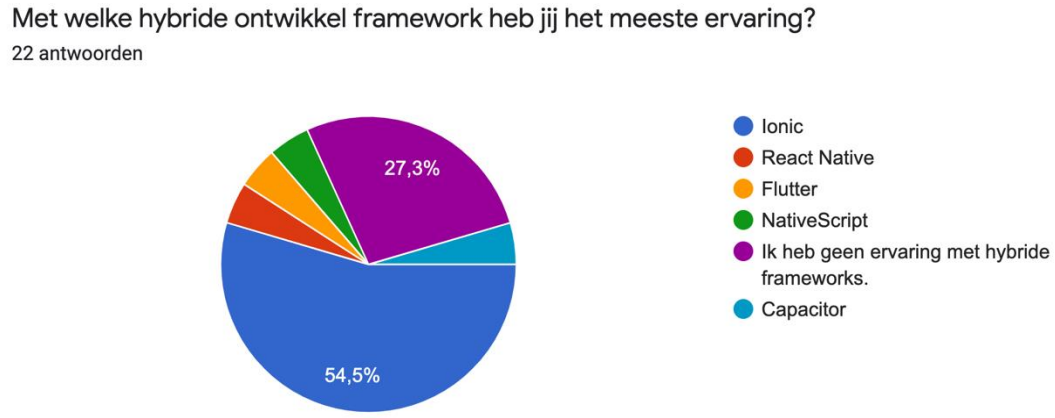
In het bovenstaande diagram is te zien dat 9 respondenten wel in opdracht van CODE14 een mobiele applicatie hebben ontwikkeld en 8 niet.




*Figuur 31 - Diagram met antwoorden op het Formulier. Titel van de vraag: Welke ontwikkel techniek is er destijds gebruikt voor de ontwikkeling van deze mobiele applicatie(s)?. Aantal antwoorden: 17 antwoorden.*

**Toelichting**

In Figuur 31 is te zien dat een groot deel van de ontwikkelde applicaties in een web-ontwikkelingstechniek is gerealiseerd, verder hebben veel respondenten ook ervaring met ontwikkelen van een Android Native Applicatie. Een klein aantal van de respondenten (3) hebben ervaring met het ontwikkelen van een iOS Native Applicatie.

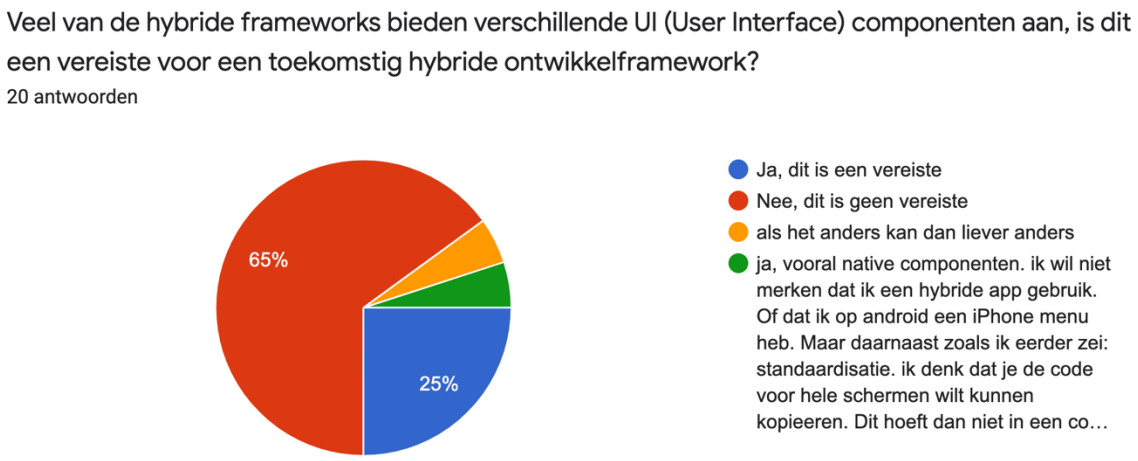



*Figuur 32 - Diagram met antwoorden op het Formulier. Titel van de vraag: Met welke hybride ontwikkel framework heb jij het meeste ervaring?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

In Figuur 32 is te zien dat de meerderheid ervaring heeft met het framework Ionic. Bij deze vraag is de mogelijkheid gegeven om een eigen antwoord in te vullen, een 6-tal respondenten heeft helemaal geen ervaring met hybride frameworks en een enkeling heeft ervaring met Flutter, React Native en NativeScript. Capacitor is toegevoegd als extra optie door een respondent, het

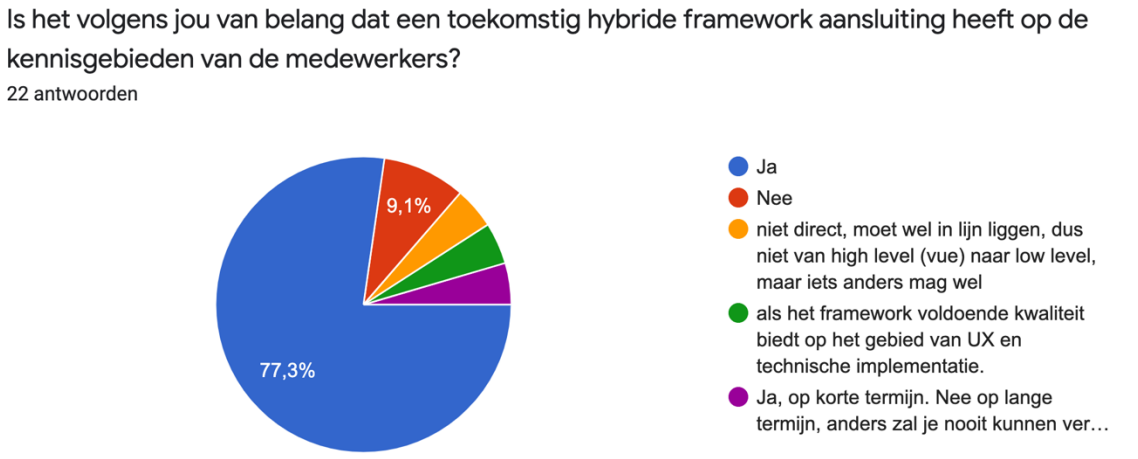
werkelijke aantal met ervaring in Capacitor ligt hoogstwaarschijnlijk hoger gezien dit al sinds eind 2019 de standaard is voor het Ionic framework.




*Figuur 33 - Diagram met antwoorden op het Formulier. Titel van de vraag: Veel van de hybride frameworks bieden verschillende UI (User Interface) componenten aan, is dit een vereiste voor een toekomstig hybride ontwikkelframework?. Aantal antwoorden: 20 antwoorden*

**Toelichting**

In het bovenstaande diagram is duidelijk naar voren gekomen dat de meerderheid van de respondenten het niet van belang vindt om UI componenten in een hybride framework hoeven te hebben. 30% van respondenten wil dit wel, waarbij één respondent heeft aangegeven waarom dit wel benodigd zou zijn.



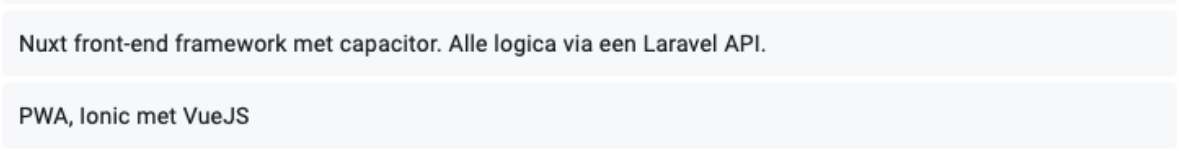

*Figuur 34 - Diagram met antwoorden op het Formulier. Titel van de vraag: Is het volgens jou van belang dat een toekomstig hybride framework aansluiting heeft op de kennisgebieden van de medewerkers?. Aantal antwoorden: 22 antwoorden.*

**Toelichting**

Het overgrote deel van de respondenten vindt dat een hybride ontwikkeltechniek aansluiting moet hebben op de kennis binnen CODE14. Een enkeling geeft hierbij verduidelijking, hierbij wordt er aangegeven dat het niet opeens helemaal anders moet gaan worden, qua ontwikkeltechniek en achterliggende programmeertaal.

**Welke ontwikkeltechnieken denk jij dat aansluiten op de technologiestack en kennis binnen CODE14 als het gaat om hybride applicatieontwikkeling?**


*Figuur 35 - Antwoord 1 t/m 8. Titel van de vraag: Welke ontwikkel technieken denk jij dat aansluiten op de technologiestack en kennis binnen CODE14 als het gaat om hybride applicatieontwikkeling? Aantal antwoorden: 11 antwoorden.*




*Figuur 36 - Antwoord 9 t/m 11. Titel van de vraag: Welke ontwikkel technieken denk jij dat aansluiten op de technologiestack en kennis binnen CODE14 als het gaat om hybride applicatieontwikkeling? Aantal antwoorden: 11 antwoorden.*

**Toelichting**

In figuren 35 en 362 is te zien dat de respondenten verschillende frameworks en ontwikkeltechnieken benoemen, dit sluit goed aan bij de antwoorden welke eerder gegeven zijn. Er worden veel Vue gerelateerde frameworks genoemd en veel aansluitende hybride ontwikkelframeworks zoals Ionic en NativeScript. Ook worden frameworks zoals ReactNative en Flutter genoemd en wordt er specifiek vermelding gemaakt van talen zoals HTML en CSS, gezien daar intern veel ervaring mee is.

**Waar denk jij dat efficiëntie slagen te behalen vallen binnen het ontwikkelen van hybride applicaties?**


*Figuur 37 - Antwoord 1 t/m 9. Titel van de vraag: Waar denk jij dat efficiëntie slagen te behalen vallen binnen het ontwikkelen van hybride applicaties?*

**Toelichting**

De antwoorden welke gegeven zijn in figuur 37 zijn allemaal redelijk uniek. Er worden veel verschillende aspecten welke veel voorkomend zijn in de ontwikkeling van een applicatie waarbij er meer efficiëntie slagen te behalen vallen. Dingen zoals herbruikbare componenten, kijken of de conclusies uit het sales proces daadwerkelijk de oplossing voor het probleem zijn, automatische builds en deploys van de applicatie (Continuous Integration/Continuous Delivery) en het integreren van low code tooling.

## 9.5 Bijlage E – Filteren o.b.v. criteria

| Framework | Criteria: C1 | Criteria: C2 | Criteria: C3 |
|---|---|---|---|
| **React Native (Version 0.64)**<br><br>**Gebruikte bronnen:** (React Native, 2021) | **Voldoet NIET aan criteria**, Het is niet mogelijk om talen zoals HTML, CSS & JavaScript te gebruiken in React Native. Gezien React zijn eigen tags gebruikt, welke tijdens het builden | **Voldoet aan criteria**, het heeft toegang tot deze native functies. React Native heeft geen externe native runtime maar heeft toegang tot bijna iedere native functie. | **Voldoet aan criteria**, Dit is mogelijk, React Native biedt wel eigen componenten aan. Maar ze bieden de mogelijkheid door hun stylingmethoden om deze aan te passen naar eigen wens. |

| | | | |
|---|---|---|---|
| | omgebouwd worden naar native code. | | |
| **Ionic + Capacitor**<br><br>**Gebruikte bronnen:** (Ionic, 2021) | **Voldoet aan criteria,** Ionic ondersteund frameworks zoals Angular, React & Vue. Waarbij Angular & Vue, omdat het in een WebView geladen wordt kan er gebruikk worden gemaakt van HTML, (S)CSS & JavaScript of TypeScript. | **Voldoet aan criteria**, Ionic maakt standaard gebruik van Capacitor als native runtime, welke toegang heeft tot alle benoemde native functies. | **Voldoet aan criteria**, Ionic biedt de mogelijkheid om te combineren met VueJS. Vue ondersteund uit zichzelf het Component Based Development principe en is sluit hierbij ook perfect aan op de kennis binnen CODE14. Ionic biedt wel eigen componenten aan, waar je enigszins beperkt bent met de het naar eigen smaak maken, maar er is geen verplichting om deze te gebruiken. |
| **Flutter**<br><br>**Gebruikte Bronnen:** (Flutter, z.d.) | **Voldoet NIET aan criteria**, Flutter maakt gebruik van Dart als programmeertaal en biedt hierbij ook geen alternatieven. Dart heeft qua schrijfwijze een iets weg van Java. Maar Flutter heeft wel documentatie om uitleg te bieden over de overgang van HTML, CSS & JavaScript naar hun manier van | **Voldoet aan criteria,** Flutter heeft toegang tot een groot aantal packages welke op de Pub pagina staan. Overigens is dit een overzichtelijk platform waarbij er duidelijk aangegeven wordt welke plugins het meest gangbaar zijn. | **Voldoet aan criteria,** In Flutter heb je de mogelijkheid om eigen ontwerpen te realiseren. Het lastige is wel dat ze Google's Material Design vrij forceren in hun documentatie, dit is ook logisch gezien het Framework ontwikkeld is door Google. |

| | | | |
|---|---|---|---|
| | ontwikkelen. Immers is in de voorbeelden te zien dat het soms wat omslachtig is. | | |
| **Xamarin**<br><br>**Gebruikte bronnen:** (Microsoft, Xamarin, z.d.) | **Voldoet NIET aan de criteria**, In Xamarin.Forms wordt met XAML & C# (.NET) geprogrammeerd. Dit sluit niet aan op kennisgebieden van CODE14, er zijn wel een aantal medewerkers (9) welke C# ervaring hebben, maar dit zijn voor namelijk backend developers, naast het feit dat deze ontwikkelaars het niet heel leuk werk vinden, is het van toepassing dat het met het uitwerken van een UI secuur te werk wordt gegaan, verschillende front-end ontwikkelaars van CODE14 geven aan dat dit in het verleden vaak niet het geval is geweest. | **Voldoet aan criteria,** Er is heel lang gezocht naar de verschillende gewenste plugins in de documentatie van Xamarin en mede hierom is de documentatie enigszins onoverzichtelijk bevonden. | **Voldoet aan criteria,** Enigszins wel, maar omdat er geforceerd wordt om van tevoren gedefinieerde componenten te gebruiken, is het lastig in schatten hoe aanpasbaar een component daadwerkelijk is. |

| | | | |
|---|---|---|---|
| **NativeScript**<br><br>**Gebruikte bronnen:**<br>(*Build Truly Native Mobile Apps with Vue | NativeScript*, z.d.) | **Voldoet aan criteria,** Ja en nee, na het zien van de volgende video, is te zien dat NativeScript niet met HTML werkt, maar met Markup. Dit houdt in dat ze elementen hebben zoals <Label>, <ScrollView>, <ActionBar>, etc. Dit is heel gunstig qua performance, want hierdoor is er de mogelijkheid om duizenden nodes tegelijkertijd weer te geven, zonder dat dit ten koste gaat van de performance. Maar dit sluit niet goed aan op de CODE14 workflow, waar HTML de standaard is. | **Voldoet aan criteria,** NativeScript heeft een eigen zogeheten 'marketplace' waar alle bruikbare plugins, templates en code samples te vinden zijn. De genoemde plugins staan hier ook op, immers zijn deze niet allemaal van het development team van NativeScript zelf, maar ook van gebruikers. | **Voldoet aan criteria,** NativeScript werkt met Markup i.p.v. HTML, zonder praktijkervaring kan er geen conclusie getrokken worden in hoeverre er restricties liggen op de aanpasbaarheid van het framework. |
| **Framework 7 + Capacitor**<br><br>**Gebruikte bronnen:**<br>(*CapicatorJS integrating in NuxtJS*, 2020), (Framework 7, z.d.) | **Voldoet aan criteria,** Framework7 heeft de mogelijkheid om te combineren met Vue, React en Svelte (Relatief nieuw open source framework). Vue sluit aan op de huidige technologiestack van CODE14. Immers maken ze hierbij veel gebruik van hun eigen componenten, zoals bijv. <f7-accordion> | **Voldoet aan criteria,** Wanneer Framework 7 i.c.m. Capacitor gebruikt zou worden, zou dit framework gebruik maken van dezelfde plugins als Ionic. | **Voldoet aan criteria,** gezien er met HTML, CSS en JavaScript gewerkt kan worden en er geen verplichting is om hun componenten te gebruiken. Immers is de vraag dan wel, hoeveel nut heeft zo'n framework? |

| | | | |
|---|---|---|---|
| **CoronaSDK (Solar2D)**<br><br>**Gebruikte bronnen:** (Corona Labs, z.d.) & (Solar2D, z.d.) | **Voldoet niet aan de criteria,** CoronaSDK heeft zijn naam gewijzigd in Solar2D. Dit framework werkt niet met Vue gerelateerde frameworks. Daarnaast is het enorm gericht op de 2D engine markt voor 2D games, dit is ook niet een markt waar CODE14 zich op richt. | **Niet meer van toepassing,** CoronaSDK of nu Solar2D is niet meer van toepassing, gezien het zich volledig focust op de 2D game industrie. | **Niet meer van toepassing,** CoronaSDK of nu Solar2D is niet meer van toepassing, gezien het zich volledig focust op de 2D game industrie. |
| **Appcelerator – Titanium**<br><br>**Gebruikte bronnen:** (Appcelerator, z.d.) | **Voldoet niet aan de criteria,** Appcelerator maakt out-of-the-box gebruik van andere technieken. Titanium gebruikt 'Titanium Style Sheets', wat lijkt op CSS, maar het niet is. En de HTML wordt hier vervangen door een markup language. De ondersteuning welke wordt gegeven voor VueJS, moet er alsnog gebruik gemaakt worden van markup elementen. | **Voldoet aan criteria,** In de documentatie van Appcelerator Titanium staan een aantal van de mobiele functies, immers was het een immense zoektocht om aansluitende 'guides' en 'tutorials' te vinden, waarbij er geen toegang was voor de website van Appcelerator Tutorials, deze gaf een foutmelding op veiligheid op het moment van onderzoeken. | **Voldoet aan criteria,** Je hebt de mogelijkheid om een eigen design te maken binnen Appcelerator Studio. Daarnaast is er de mogelijkheid om d.m.v. TSS eigen styling te geven aan elementen. |
| **NuxtJS + Capacitor**<br><br>**Gebruikte bronnen:** (*CapicatorJS integrating in NuxtJS*, 2020) | **Voldoet aan criteria,** Capacitor wordt hier aangeboden als Native Runtime, dit biedt de toegang tot verschillende functionaliteiten, waar NuxtJS fungeert als | **Voldoet aan criteria,** Capacitor biedt de mogelijkheid om gebruik te maken van alle benodigde native functies, en meer. Het voordeel dat NuxtJS met zich meebrengt. | **Voldoet aan criteria,** NuxtJS biedt de mogelijkheid om met standaard Vue componenten (HTML, (S)CSS & javaScript of TypeScript) te ontwikkelen. Hierin zitten geen beperkingen qua UI, |

| | | | |
|---|---|---|---|
| | front-end framework om de UI in te bouwen. | | gezien alle UI componenten zelf ontwikkeld moeten worden, biedt dit dezelfde mogelijkheden als normale web-ontwikkeling. |

*Tabel 10 - Eerste afbakening van relavante frameworks*

**Voor- en nadelen gevonden tijdens eerste filtering**

| Framework | Voordelen | Nadelen |
|---|---|---|
| **React Native** | 1) React Native kan daadwerkelijk compileren naar Native applicaties i.p.v. een WebView, dit heeft positieve impact op de performance van de applicatie. | 1) React Native ondersteund niet het web platform, wel iOS & Android.<br>2) React Native sluit niet goed aan op de huidige technologiestack. |
| **Ionic i.c.m. Vue & Capacitor** | 1) Ionic i.c.m. Capacitor biedt zowel de mogelijkheid om voor Android & iOS Applicaties te builden, als voor Web d.m.v. Electron.<br>2) Capacitor biedt ook de mogelijkheid om PWA's (Progressive Web Apps) te builden.<br>3) Ionic heeft een heel ecosysteem voor CI/CD. (AppFlow, betaald platform) | 1) Capacitor bouwt applicaties in een WebView, wat voor mindere performance zorgt t.o.v. bijv. React Native of NativeScript. |

| | | |
|---|---|---|
| **Flutter** | 1) Flutter heeft een grote community en is gemaakt door Google. Daarnaast heeft het support voor veel platformen: iOS, Android, Linux, macOS en Web (Chrome, Safari, Firefox & Edge) volgens (Flutter, 2021). | 1) Het is een te groot nadeel dat er alleen in Dart geprogrammeerd kan worden, gezien er weinig front-end developers binnen CODE14 ervaring hebben met soort gelijke talen of Dart zelf. Hierdoor ligt de learning curve te hoog en kan er niet snel van start gegaan worden met dit framework. |
| **Xamarin** | 1) Xamarin biedt ondersteuning voor iOS-, Android- en Windowsapplicaties. | **1)** Niet gericht op huidige kennis.<br>2) Is qua IDE (Integrated Development Environment) gericht op **Visual Studio Code**. De meeste CODE14 medewerkers werken met **PHPStorm**. In eerste instantie zou dit niet een heel groot probleem moeten zijn, maar de huidige codestyling is in *.xml* files geschreven en deze zijn niet te integreren in Visual Studio Code. Ook zijn er enorm wat plugins verzameld voor PHPStorm, waar voor er alternatieven gezocht zouden moeten worden in **Visual Studio Code.**<br>3) Geen ondersteuning voor **web-applicaties.** |
| **NativeScript i.c.m. Vue** | **1)** NativeScript heeft directe verbinding met de native API's, net als bij React Native. | 1) NativeScript ondersteund niet het web platform.<br>2) NativeScript werkt met markup elementen, om deze vervolgens |

| | | |
|---|---|---|
| | 2) NativeScript heeft de mogelijkheid om gebruikt te worden i.c.m. Vue, Angular en React.<br>**3)** NativeScript heeft een playground waar doormiddel van een drag-and-drop systeem, kant en klare componenten getest kunnen worden. | rechtstreeks om te zetten in native elementen. |
| **Framework 7 i.c.m. Vue & Capacitor** | **1)** Ook ondersteuning voor web-applicaties.<br>2) Veel handige code-samples, voor directe integratie. | 1) Op de website staan er tutorials vanaf 2015 t/m 2019, wat niet heel erg bij de tijd is.<br>2) Framework 7 maakt out-of-the-box nog gebruik van Cordova, combinatie met Capacitor is mogelijk maar is omslachtig in het initialisatieproces, gezien er dan in de CLI voor web-only gekozen moet worden en vervolgens apart Capacitor geïnstalleerd moet worden.<br>3) Framework 7 is Open-Source, wat goed is. Maar wordt niet ondersteund door een achterliggend bedrijf, dus updates is volledig afhankelijk van de community en ontwikkelaar van het framework. In tegenstelling tot Ionic, React Native, Xamarin & Flutter |
| **CoronaSDK** | N.v.t. | 1) Het lijkt erop dat CoronaSDK zich volledig heeft gefocust op het wezen van een 2D Engine. Hun website geeft aan dat ze een naamswijziging hebben doorgevoerd naar Solar2D en linken hierbij ook naar een |

| | | |
|---|---|---|
| | | nieuwe website. Naast dit, sluit het framework niet aan op de eerste criteria vanuit CODE14, waardoor CoronaSDK (Solar2D) niet meegenomen wordt naar fase 2. |
| **Appcelerator** | 1) Appcelerator heeft een eigen IDE, waar een vorm van Low-code toegepast wordt met drag-and-drop elementen. Dit zou handig kunnen zijn, maar deze vorm van low-code is niet van toepassing voor CODE14, gezien hier een leertraject aan hangt en dit alleen toegankelijk zou zijn voor de medewerkers met designervaring. | 1) Eerste opzicht erg rommelig qua website en documentatie moeilijk te vinden, documentatie welke gevonden is, is qua structuur niet overzichtelijk t.o.v. de voorgaande frameworks.<br>2) Heeft wel Vue support, maar is geen specifieke documentatie over te vinden. (> 30 minuten gezocht). |

*Tabel 11 - Gevonden voor- en nadelen van relevante frameworks*

## 9.6 Bijlage F – Onderbouwing uitkomsten rubric per onderwerp

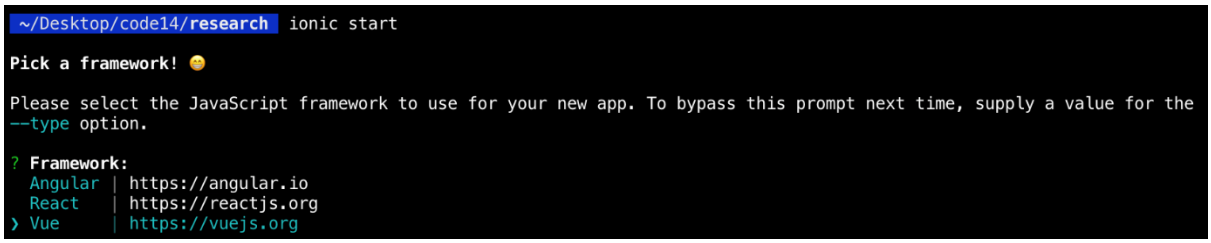
### 9.6.1 R1 - Installatie en development proces (10%)

**Ionic** – Cijfer: 9

Ionic is eenvoudig op te zetten. De eerste stap is om de CLI te installeren d.m.v. het commando *npm install -g @ionic/cli*, hiervoor is **NPM (Node Package Manager)** benodigd. Vervolgens heb je de mogelijkheid om *ionic start* commando uit te voeren, hierna worden er een aantal vragen gesteld, zoals: Welk JavaScript framework wil je gebruiken? Wat is de naam van het project? Wil je een startertemplate gebruiken? (Tabs, Sidemenu, Blank of List). Daarnaast bieden ze ook een App Wizard aan, welke rechtstreeks in het versiebeheersysteem naar keuze een repository voor je aanmaakt.

Ionic i.c.m. Capacitor biedt meerdere hot reloading mogelijkheden aan, zo is het mogelijk om via een extern device te testen, maar ook via een browser. Dit maakt het ontwikkelproces erg prettig en gemakkelijk.

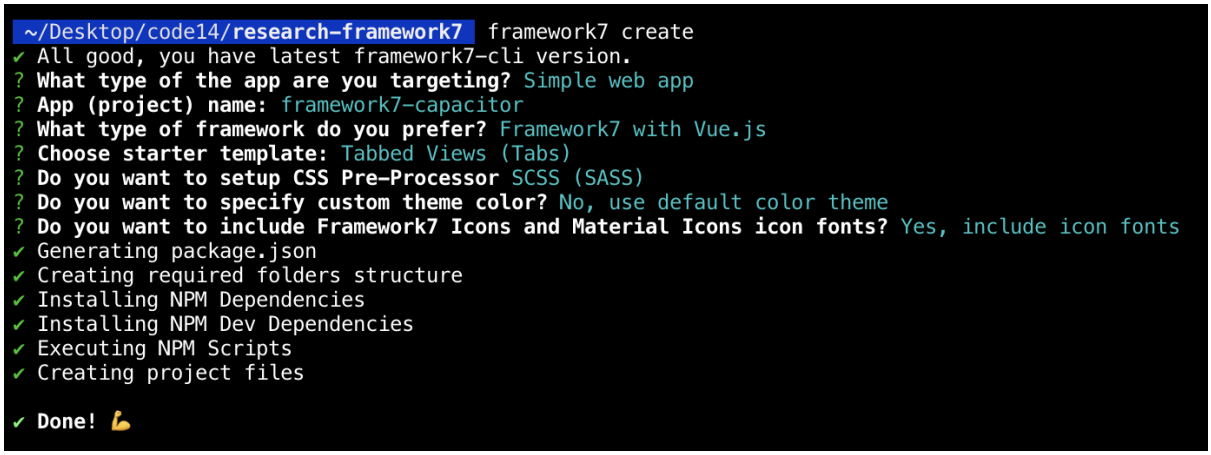
Al met al is er weinig te klagen over het opzetten en ontwikkelen binnen het Ionic framework. CODE14 medewerkers zijn gewend om via de CLI te werken, dus dit moet geen probleem zijn.




*Figuur 38 - CLI Ionic, Setup van nieuw project*

**Framework 7 – Cijfer: 9**

Ook binnen Framework 7 is het opzetten van een nieuw project een 'piece of cake'. Het enige wat er benodigd is de framework7-cli. Vervolgens kan het framework7 create commando uitgevoerd worden, deze stelt net als bij Ionic een aantal vragen. Zoals wat het doeleinde is voor de applicatie en welke runtime je wilt gebruiken (Simple web app, Cordova of Capacitor.)




*Figuur 39 - CLI Framework7, Opzetten nieuw project*

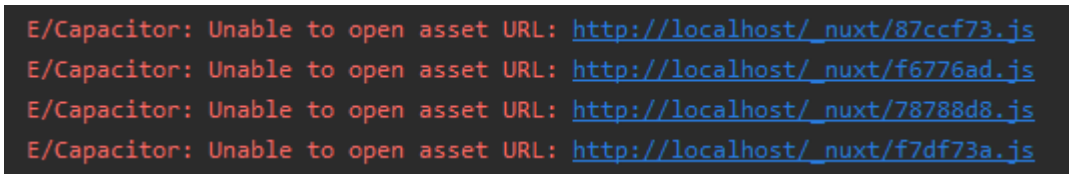
Het development proces is daarnaast hetzelfde als bij Ionic, er is mogelijkheid tot hot reloading. Wanneer de –ui flag toegevoegd wordt aan het framework7 create commando, zal er een localhost venster openen waar dezelfde vragen worden gesteld voor het opzetten van het project. Dit is overzichtelijker dan in de CLI.

**NuxtJS Capacitor – Cijfer: 6**

NuxtJS met capacitor is iets omslachtiger. Dit komt omdat NuxtJS van origine geen hybride development framework is, dus biedt in de CLI ook niet de optie om Capacitor toe te voegen. Maar wanneer Capacitor eenmaal toegevoegd is, kan dit gebruikt worden zoals in ieder ander

framework. Voor een duidelijk uitleg is er de tutorial van (Pastori, 2020) gevolgd, hierin wordt stap voor stap omschreven hoe je aan een bestaand Nuxt project, Capacitor kan toevoegen.

Om uit te vinden of dit daadwerkelijk functioneert, is er besloten een eerste opzet voor de applicatie te maken in NuxtJS gezamenlijk met Capacitor. Het initialisatieproces was iets meer werk, gezien de config files van Capacitor juist moest worden ingesteld. Wanneer er een 'nuxt build' en 'nuxt generate' wordt gedraaid, wordt er een dist folder gemaakt in de root van het project. Hierin staat de statisch gegenereerde webapplicatie. Deze is vervolgens met 'npx cap add android' over te zetten naar de Androidproject. Immers lopen we hierbij tegen een probleem aan, zoals te zien is in Figuur 40.




*Figuur 40 - Error in DebugConsole van Android Studio*

De gegenereerde 'dist' map, heeft alle javascript bestanden in de onderliggende '_nuxt' map gestopt. Android kan hier niet mee overweg, gezien dit platform alle map- en bestandsnamen beginnend met een '_' negeert. Hiervoor is een workaround te maken door een bestand genaamd *build.gradle* aan te passen in het gegenereerde Android project, en hier de code uit in Figuur 41 toe te voegen in de defaultConfig.

```
defaultConfig {
    ...
    aaptOptions {
        ignoreAssetsPattern '!.svn:!.git:!.ds_store:!*.scc:!CVS:!thumbs.db:!picasa.ini:!*~'
    }
}
```

*Figuur 41 - Code welke toegevoegd moet worden aan build.gradle file*

### 9.6.2 R2 – Community (10%)

**Ionic – Cijfer: 9**

Ionic heeft op 29-03-2021, 43.900 stars op GitHub (GitHub, Ionic, z.d.). En is van de drie overgebleven framework dus het framework met de meeste stars. Daarnaast heeft Ionic een eigen forum waar actief op gediscussieerd wordt en een community pagina waar verschillende live meetups op te vinden zijn. . Ionic heeft ieder kwartaal een live evenement, waarbij ze nieuwe features en ontwikkelingen tentoonstellen. Al met al heeft Ionic een grote en hechte community, waar zeker hulp te vinden is.

**Framework7 – Cijfer: 8**

Framework7 heeft op 29-03-2021, 16.084 stars op GitHub (GitHub, Framework7, z.d.). Net als Ionic heeft Framework7 een eigen forum waarop gediscussieerd en geholpen wordt. Er bestaat ook een blog maar hier wordt niet heel vaak op gepost.

**NuxtJS + Capacitor – Cijfer: 7**

NuxtJS heeft op 29-03-2021, 34.600 stars op GitHub (GitHub, NuxtJS, z.d.). Capacitor heeft op GitHub 5.400 stars (GitHub, Capacitor, z.d.). Immers zal er op het criteria community wel een punt aftrek zijn voor deze combinatie, het punt blijft dat NuxtJS niet gefocust is op hybride applicatieontwikkeling en daarmee per definitie al een minder grote community heeft dan een Ionic of Framework7 op dit gebied, ook online is er weinig te vinden over deze combinatie in vergelijking met de andere twee frameworks. Net als Ionic en Framework 7 heeft Nuxt een blog, maar ook hier wordt niet heel veel op geplaatst.

### 9.6.3 R3 – Ondersteunde platformen (15%)

In vergelijking met de voorheen afgevallen frameworks heeft Capacitor veel te bieden op dit gebiedt. Alle frameworks hebben hierbij een 9 gekregen, omdat hierbij er nog geen ondersteuning is voor bijvoorbeeld Smartwatch platformen zoals Tizen- of WearOS.

**Ionic + Vue + Capacitor – Cijfer: 9**

Ionic i.c.m. Vue & Capacitor ondersteund **Android, iOS, Web, PWA's, en Electron** applicaties (Lynch, z.d.). Capacitor ondersteund dit uit zichzelf, Ionic is hierin alleen de 'leverancier' van de UI en UX elementen en Vue is hierin het front-end framework.

**Framework7 + Vue + Capacitor Cijfer: 9**

Gezien Framework7 i.c.m. Vue & Capacitor net als Ionic gebruik maakt van Capacitor, ondersteund dit dezelfde platformen, namelijk **Android, iOS, Web, PWA's en Electron applicaties**. Het enige verschil hierin is dat er gebruik gemaakt wordt van Framework7 UI en UX-elementen.

**NuxtJS + Capacitor – Cijfer: 9**

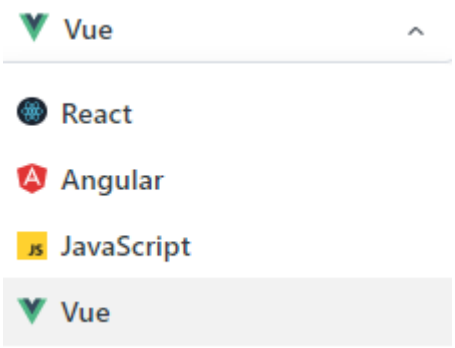
Ook hier wordt er als native runtime weer gekozen voor Capacitor. Net zoals Ionic en Framework7 ondersteund deze combinatie dan ook **Android, iOS, Web, PWA's, en Electron applicaties.** Dus de verschillende platformen zijn goed.

### 9.6.4 R4 – Documentatie (15%)

Om een goed oordeel te kunnen geven over de bijbehorende documentatie van een framework, is er gekeken naar vindbaarheid van onderwerpen, uiterlijk van website (Leesbaarheid) en compleetheid van de documentatie. Gezien de drie overgebleven combinaties allemaal Capacitor gebruiken, zal deze in een apart kopje meegenomen worden.

**Ionic – Cijfer: 8.5**

Ionic heeft goede overzichtelijke documentatie. Ze leveren complete handleidingen aan, waar het initialisatieproces mee versneld kan worden, doormiddel van onder andere de App Wizard. Daarnaast hebben ze duidelijk afscheiding tussen Components, CLI (Command-Line Interface) en Native documentatie, zodat er niet onnodig lang gezocht moet worden tussen irrelevante informatie. Er wordt gefilterd op de documentatie behorend bij het gekozen front-end framework (of juist geen framework), hierbij heb je de keuze uit *AngularJS*, *ReactJS*, *VueJS* en *Javascript*.

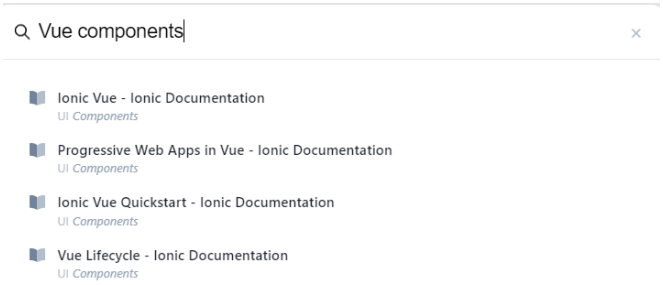



*Figuur 42 - Bruikbare frameworks in Ionic*

*Figuur 43 - Bruikbare frameworks in Ionic*

Er zijn gehele walkthroughs voor basis applicaties voor het maken en opslaan van foto's en er kan in de documentatie gelezen worden hoe dingen zoals lifecycle, PWA's, testen, troubleshooting en configuratie in elkaar zitten.

Zoekfunctionaliteit is goed geoptimaliseerd, waarbij je met steekwoorden zoals 'Vue', 'React' of 'Angular' al kan filteren.




*Figuur 44 - Vue componenten in zoekresultaten*

Het is goed leesbaar, goed vindbaar en er is bij ieder component wel een code sample te vinden. Het voordeel wat Ionic heeft ten opzichte van de andere frameworks is dat er vaak in de documentatie rechtstreeks gerefereerd wordt naar de native runtime Capacitor, gezien deze door hetzelfde bedrijf ontwikkeld is.

*Figuur 45 - Vue componenten in zoekresultaten*

**Framework7 – Cijfer: 8**

De documentatie van Framework7 is overzichtelijk en duidelijk. Het is net als bij Ionic te filteren op basis van een gekozen front-end framework, met hierbij de opties voor *VueJS*, *ReactJS* en *Svelte*. Daarnaast is er ook afzonderlijke documentatie voor de CLI (Command-Line Interface) en de CORE/API. Ook heeft Framework 7 handleidingen gemaakt, immers zijn deze al wel wat ouder, de laatste is geplaatst op 7 Oktober, 2019.

De zoekfunctionaliteit op de website is iets minder goed geoptimaliseerd dan die van Ionic. Als de gebruiker zich de Framework7 Vue documentatie begeeft en vervolgens het woord

‘Components’ in de zoekbalk typt, komt er een lijst met React componenten aan voren, wat niet wenselijk is.

Al met al is het goede documentatie, alles wat een ontwikkelaar in eerste instantie nodig heeft, is te vinden, maar er is ruimte voor verbetering.

**NuxtJS - Cijfer: 8**

NuxtJS heeft goede documentatie. Er zijn starters handleidingen, installatie hulp en er wordt duidelijk uitgelegd wat de mogelijkheden van het framework zijn. Je kunt het framework online uitproberen, zonder dat je iets hoeft te installeren. Recent heeft NuxtJS ook video courses toegevoegd aan hun website, waar de ontwikkelaar zelf kan kiezen uit onderwerpen zoals; Async data, Routing, Statemanagement with VueX, Organizing large projects, SEO en nog veel andere onderwerpen. Je kunt toegang krijgen tot een preview van deze course, maar het gehele pakket kost geld.

**Capacitor – Cijfer: 8**

Capacitor heeft goed leesbare en overzichtelijke documentatie, het heeft wat weg van Ionic, maar dit komt voornamelijk omdat het door hetzelfde bedrijf ontwikkeld is. Het is makkelijk om features te vinden in de documentatie, welke bruikbaar zijn wanneer een bepaalde versie van Capacitor in een project geïmplementeerd is.

Er zijn veel handleidingen, immers zijn deze wel vaak gericht op één specifieke front-end stack, meestal één van de stacks welke Ionic ondersteund, zoals Angular, React of Vue.

### 9.6.5 R5 - Prestatie

Gezien alle drie de combinaties gebruik maken van Capacitor en er in prestatie weinig verschil zal zijn tussen de frameworks. Hierdoor is de keuze gemaakt om een kort literatuuronderzoek te doen naar de verschillen in prestatie tussen Frameworks zoals React Native of NativeScript in vergelijking met Capacitor.

Testtoestel: **Samsung Galaxy S20 – Exynos 990 Chip**

**Framework 7 v6 + Capacitor (v2.4.7) – Cijfer: 8** neemt bij een nieuwe Androidapplicatie zo’n **10,99MB** in beslag. De gebruikers ervaring is erg soepel. (Clean Install - Globaal inladen van alle beschikbare componenten)

**Ionic Vue V5.6.0 + Capacitor (v2.4.7) – Cijfer: 8** neemt bij een nieuwe Androidapplicatie zo’n **9,96MB** in beslag. (Clean Install - Globaal inladen van alle beschikbare componenten)

**NuxtJS + Capacitor (v2.4.7) – Cijfer: 8** neemt bij een nieuwe Androidapplicatie zo’n **8,38MB** in beslag. (Clean install)

De componenten van zowel Ionic als Framework 7 verlopen heel goed op het testtoestel. Na installatie van de frameworks zijn deze direct omgezet in uitvoerbare Androidapplicaties om te kijken hoeveel opslag deze out-of-the-box in gebruik nemen. Hierbij kwam Nuxt als beste uit de test, wat ook te verwachten was, gezien dit geen UI elementen meelevert.

### 9.6.6 R6 – Learning Curve, Toepasbaarheid en Toegankelijkheid

De learning curve ligt voor alle drie de frameworks erg laag, waarbij bij NuxtJS het laagst omdat dit al geïntegreerd is in de huidige front-end stack. Voor Ionic en Framework7 zou de documentatie doorgenomen moeten worden voor de bruikbare componenten, maar dit is geen verandering in workflow of stack.

**Ionic – Cijfer: 8**

Ionic i.c.m. Vue & Capacitor zou heel toegankelijk zijn, voorheen is er binnen CODE14 al gekozen voor Ionic i.c.m. Angular & Cordova, maar hier konden maar weinig medewerkers mee overweg, omdat Angular dusdanig anders was dan Vue.

**Framework7 – Cijfer: 8**

Qua learning curve ligt Framework7 op hetzelfde niveau als Ionic, gezien er componenten zijn waarvan de documentatie gelezen zouden moeten worden. Qua workflow is dit hetzelfde als de meeste front-enders nu doen. Er wordt gewoon in HTML, CSS en JavaScript geschreven en er zijn geen limitaties m.b.t. het stijlen van eigen componenten.

**NuxtJS – Cijfer: 7**

Gezien er CODE14 breed al gewerkt wordt met Nuxt, heeft dit geen verdere learning curve, al zijn er natuurlijk altijd aspecten aan een framework waarin verbetert of geleerd kan worden, dus hier scoort het framework beter op dan bijv. Ionic of Framework7. Qua toepasbaarheid, is het in combinatie met Capacitor wel prima, maar het opzetten van een nieuw hybride project is lastig en dat is weer een struikel punt op toegankelijkheid, het zal niet direct werken, er zijn mogelijk dingen welke in de toekomst tegen aangelopen wordt, puur omdat NuxtJS initieel niet ontwikkeld is als onderdeel van een hybride stack.

### 9.6.7 R7 - Builden en Deployment (CI/CD)

Een productie-klaar product op de App Store of Play Store zetten kan soms een ingewikkeld proces zijn, bij de verschillende frameworks wordt er gekeken hoe makkelijk dit te doen is.

De native runtime Capacitor zorgt er bij alle drie de frameworks voor dat er een ‘native’ applicatie gebouwd wordt, weliswaar in een WebView maar voor het live zetten van een productieklare applicatie doet dat er niet toe. De gegenereerde bestanden vanuit Capacitor zijn te openen in XCode of in Android Studio (afhankelijk van het platform), deze zijn vervolgens om te zetten naar

een uitvoerbare applicatie voor het corresponderende platform, welke vervolgens op de App- of Play store te zetten is.

**Continuous Integration-/ Continuous Deployment of Delivery (CI/CD)**

Dit zijn belangrijke termen als het aankomt op het live zetten van een applicatie. Voornamelijk Apple (iOS) keurt applicatie niet vaak in één keer goed, ze moeten voldoen aan specifieke eisen en worden vaak met de hand getest voordat ze op de App Store mogen komen. Ook CODE14 heeft hier in het verleden vaak 'problemen' mee gehad. Dit proces kan versimpeld worden door CI/CD tooling zoals AppFlow. Dit is een product van Ionic, welke direct vanuit het versiebeheersysteem getriggerd wordt om een deployment verzoek te doen bij de App/Play store(s). Daarnaast wordt er doormiddel van een Pipeline, al grotendeels gecontroleerd of de applicatie voldoet aan bepaalde platform specifieke eisen. Hierdoor is de ontwikkelaar al op de hoogte van de problemen in de applicatie en hoeft deze niet te wachten op afkeuring vanuit het corresponderende platform.

**Ionic + Capacitor – Cijfer: 9**

In tegenstelling tot NuxtJS, is Ionic ervoor gemaakt om samen te werken met een native runtime en heeft hier Capacitor als grote voorkeur, gezien het ontwikkeld is door hetzelfde bedrijf. Hierdoor is er meer zekerheid over een goed build proces, waarbij er rekening gehouden wordt met de vereisten van een gekozen platform, zoals Android of iOS.

**Framework7 + Capacitor – Cijfer: 9**

Framework7 heeft ook een goede samenwerking met native runtimes, gezien het framework daadwerkelijk voor hybride development gebouwd is. Het maken van een uitvoerbare Androidapplicatie is dan ook vrij eenvoudig te doen.

**NuxtJS + Capacitor – Cijfer: 6**

NuxtJS en Capacitor is niet optimaal op dit vlak. Tijdens het ontwikkelen van een klein prototype, kwamen er al build problemen naar voren, waar initieel geen foutmeldingen voorbijkwamen. Zoals het niet inladen van JavaScript files in een Android build. Mede hierdoor zal er bij het opzetten van een project en het builden van een uitvoerbaar .apk bestand, veel handmatig aangepast worden voordat een CI/CD oplossing toegepast kan worden en zelfs dan kun je niet met zekerheid zeggen, gezien de combinatie niet officieel ondersteund wordt.

### 9.6.8 R8 - Plugins, Native Functies

Zoals in fase 2 ondervonden is, heeft Capacitor toegang tot alle vooraf aangegeven native functies zoals Camera, NFC, Bluetooth, LocalStorage en meer. Daarnaast is Capacitor een native runtime waaraan doorontwikkeld wordt, ieder kwartaal worden er nieuwe features

gepresenteerd tijdens een online evenement en groeit dit framework mee met de mogelijkheden van hedendaagse smartphones en tablets.

**Ionic – Cijfer: 8.5**

Ionic heeft op hun eigen website documentatie voor het implementeren van verschillende native functies. Hier staan niet altijd code voorbeelden voor VueJS bij, dit komt omdat VueJS het laatste platform is waar officiële ondersteuning voor gekomen is, dus mogelijk is deze documentatie nog in de maak. Immers is de VueJS implementatie vaak gelijk aan de implementatie van plain JavaScript, waarbij de enige aanpassing is dat de functie in de 'methods' gezet moet worden en elders getriggerd moet worden, in VueJS. Gezien er specifieke documentatie is op de website van het framework zelf en er meer zekerheid is m.b.t. de samenwerking van het framework en de native runtime, heeft Ionic hier een halve punt hoger gekregen dan Framework7.

**Framework7 – Cijfer: 8**

Framework7 biedt geen documentatie aan voor native functies & plugins. Dit komt omdat dit framework niet officieel geaffilieerd is met een native runtime zoals Cordova of Capacitor. Immers, om een applicatie te kunnen builden voor een specifiek platform, is er wel een native runtime nodig, hiervoor wordt verwezen naar de documentatie van de gekozen native runtime in het initialisatie proces. Verder is de combinatie tussen dit framework en Capacitor in staat dezelfde functies aan te roepen als Ionic, het zou in theorie zelfs mogelijk zijn om de documentatie van Ionic te gebruiken om in Framework7 een native functie aan te roepen, gezien dit gebaseerd is op dezelfde native runtime.

**NuxtJS – Cijfer: 7**

Ook NuxtJS is niet officieel geaffilieerd met Capacitor of Cordova. Capacitor zelf heeft alle benodigde documentatie voor het ontwikkelen van een hybride applicatie met gebruik van native functies zoals Camera, Bluetooth, etc. Hierdoor heeft het in theorie dezelfde mogelijkheden als Ionic. Het nadeel aan deze combinatie is wel, dat frameworks zoals Ionic of Framework7 vaak elementen hebben ontwikkeld, zoals animaties passend bij het platform, welke nauw samenwerken met de te gebruiken native functies.

## 9.7 Bijlage G – Requirements VYBER hybride applicatie

| Naam | Requirement | MoSCoW | F/NF |
|---|---|---|---|
| A1 | Als gebruiker wil ik doormiddel van een e-mailadres en wachtwoord mijzelf kunnen registreren voor de applicatie. | Must | F |
| A2 | Als gebruiker wil ik doormiddel van een geregistreerd account met accountgegevens zoals e-mailadres en wachtwoord kunnen inloggen binnen de applicatie. | Must | F |
| A3 | Als gebruiker wil ik doormiddel van het geregistreerde e-mailadres van het account een 'wachtwoord vergeten' aanvraag kunnen doen. | Must | F |
| A4 | Als gebruiker wil ik na een wachtwoord vergeten aanvraag, doormiddel van een link in een e-mailbericht mijn wachtwoord kunnen resetten of wijzigen. | Must | F |
| A5 | Als gebruiker wil een VYBER-fiets kunnen registreren en koppelen aan mijn account. | Must | F |
| A6 | Als gebruiker wil ik de mogelijkheid hebben om na succesvolle authenticatie te kunnen navigeren door de applicatie d.m.v. een tab-bar of navigatiebalk. | Must | NF |
| A7 | Als gebruiker wil ik na succesvolle authenticatie doorverwezen worden naar het overzicht van fietsen, mits ik meerdere fietsen heb gekoppeld aan mijn account. | Should | NF |
| A8 | Als gebruiker wil ik na succesvolle authenticatie doorverwezen worden naar het dashboard van mijn Vyber Fiets, mits ik maar één fiets geregistreerd heb. | Should | NF |
| A9 | Als gebruiker wil op het fiets dashboard duidelijk kunnen zien van welke fiets er gegevens worden weergegeven. | Must | NF |
| A10 | Als gebruiker wil ik van de geselecteerde fiets gegevens kunnen inzien op het dashboard, zoals: Fiets model naam, Fiets model afbeelding, Accupercentage, Fietsradius tonen, Kilometerstand, Knop voor locatie tonen, Killmodus knop, Laatste update van LoRa connectie | Must | NF |
| A11 | Als gebruiker wil ik bij een nieuw account waar nog **geen** 'Connected Ride' pakket op afgesloten is, de mogelijkheid krijgen om deze af te sluiten. | Would | NF |

| | | | |
|---|---|---|---|
| A12 | Als gebruiker wil ik een 2e hands VYBER-fiets kunnen koppelen aan mijn profiel, mocht deze al gekoppeld zijn aan die van een ander. | Would | NF |
| A13 | Als gebruiker wil algemene gegevens kunnen wijzigen zoals: Profielfoto, naam, e-mailadres, telefoonnummer, geslacht, geboortedatum, NAW-gegevens. | Must | F |
| A14 | Als gebruiker wil ik de optie voor push-notificaties (berichtgeving) kunnen in of uitschakelen binnen de applicatie. | Must | F |
| A15 | Als gebruiker wil ik betalingsgegevens kunnen wijzigen mits ik een 'Connected Ride' abonnement bezit. | Must | F |
| A16 | Als gebruiker wil ik een handleiding van de fiets terug kunnen vinden in de applicatie, zodat ik deze altijd te kan vinden. | Must | NF |
| A17 | Als gebruiker wil ik graag de algemene voorwaarden kunnen lezen voor de applicatie en het product. | Must | NF |
| A18 | Als gebruiker wil ik het privacy beleid van de applicatie kunnen vinden en lezen. | Must | NF |
| A19 | Als gebruiker wil ik een andere gebruiker met een VYBER-id kunnen toevoegen aan een fiets. | Must | F |
| A20 | Als gebruiker wil ik specifieke fietsinformatie kunnen inzien, zoals: Kilometerstand, dagteller, type/model, Bike-ID, Batterij type en de registratiedatum. | Must | NF |
| A21 | Als gebruiker wil ik mijn abonnement van VYBER Connected Ride kunnen opzeggen. | Must | F |
| A22 | Als gebruiker moet ik verzekeringsvragen invullen als ik een Vyber Connected Ride verzekering wil afsluiten. | Must | NF |
| A23 | Als gebruiker wil ik binnen 3 maanden mijn Vyber Connected Ride basic pakket willen upgraden naar een All-Inclusive verzekeringspakket. | Must | NF |
| A24 | Als gebruiker wil ik de applicatie kunnen installeren via de iOS App Store en de Google Play Store. | Must | NF |

*Tabel 12 - Requirements VYBER applicatie*

## 9.8 Bijlage H – Schattingen VYBER hybride applicatie

| Functionaliteit | | Uren design | Uren frontend | Uren backend | Uren CMS | Totaal frontend |
|---|---|---|---|---|---|---|
| Deployment | | | 16 | | | 322 |
| Project setup | | | 8 | 6 | | |
| OTAP inrichten | | | | | | |
| | | | | | | |
| | | | | | | |
| **Navigatie** | | 0 | 12 | | 0 | |
| | | | | | | |
| **Login** | | | | | | |
| | Start | | 6 | | | |
| | Login | | 6 | | | |
| | Vyber-ID maken | | 12 | | | |
| | No-email ID maken | | 6 | | | |
| | set password mail sent | | 4 | | | |
| | set password SMS sent (o.b.v. mail sent) | | 1 | | | |
| | password reset - step 1 mail | | 4 | | | |
| | password reset - step 1 phone-number (o.b.v step1 | | 1 | | | |
| | reset password mail sent (o.b.v. mail sent) | | 2 | | | |
| | reset password sms sent (o.b.v. mail sent) | | 2 | | | |
| | e-mail known modal popover | | 6 | | | |
| | | | | | | |
| **Scan bike** | | | | | | |
| | Scan first bike | | 4 | | | |
| | QR Code front-end | | 4 | | | |
| | QR code implementation | | 8 | | | |
| | Add BIN manually | | 4 | | | |
| | Free connected ride page | | 8 | | | |
| | Connected Ride Upsell (O.b.v. free connected ride p | | 3 | | | |
| | Connected Ride Upsell - Standalone page | | 8 | | | |
| | Connected Ride Upsell - No form | | 12 | | | |
| | Connected Ride Upsell - Yes form | | 12 | | | |
| | Afhandeling fouten | | 4 | | | |
| | | | | | | |
| **Devices** | | | | | | |
| | Devices | | 6 | | | |
| | Menu overlay | | 8 | | | |
| | Devices - profile | | 6 | | | |
| | | | | | | |
| **Device - Dashboard** | | | | | | |
| | Dashboard header | | 4 | | | |
| | V-points component | | 2 | | | |
| | Kilometerstand | | 4 | | | |
| | Onderhoud over | | 2 | | | |

*Figuur 46 - Eerste pagina van schatting VYBER-hybride applicatie o.b.v. ontwerp*

| | | |
|---|---|---|
| | Interactive buttons, battery charging, etc. | 3 |
| | Locked notification | 2 |
| | Kill modus activated | 2 |
| | Locatie van je fiets | 4 |
| | No connected ride (o.b.v. upsell template) | 4 |
| | | |
| **Device - Detail** | | |
| | Device - detail page | 4 |
| | Device - bike information | 6 |
| | Device - service | 4 |
| | Device - garantie bepaling | 4 |
| | Device - gebruikers | 8 |
| | Device - gebruiker uitnodigen | 16 |
| | Device - abonnement wijzigen | 16 |
| | Device - connected ride upsell | 8 |
| | | |
| **Device - no subscription** | | |
| | Device - Dashboard - No Connected Ride | 4 |
| | No Subscription - Connected Ride Upsell | 4 |
| | | |
| **Profiel** | | |
| | Profile - Landingspage | 8 |
| | Profile - General information | 8 |
| | Profile - Payment information | 2 |
| | Profile - Logout modal | 4 |
| | Profile - Privacy & Terms and Conditions | 6 |
| | | |
| **Branding** | | |
| | App Icon | 2 |
| | Splash Screen | 2 |
| | | |
| **Deeplinking** | | |
| | Research | 8 |
| | Implementation | 8 |

*Figuur 48 - Schatting o.b.v. pagina's in het ontwerp*

| | |
|---|---|
| **Subtotaal uren** | 328 |
| **Testing** | 49,2 |
| **Projectmanagement** | 49,2 |
| | |
| **Totale uren** | 426,4 |
| **Uurtarief** | 100 |
| | |
| | |
| **Prijs** | 42640 |
| | |

*Figuur 47 - Totale uren*