PROJECT PLAN

STAGE 23/02/2026 - 22/05/2026

SENNE FABRI
THOMAS MORE GEEL
Toegepaste Informatica

Inhoudsopgave

Introductie

Tijdens mijn laatste jaar in de richting Toegepaste Informatica – Digital Innovation doe ik mijn stage bij Nitor. Mijn opdracht bestaat eruit een Prove Of Concept (POC) te ontwikkelen omtrent een mobiele app in de bouwsector.

In dit document zal ik mijn analyse presenteren en plannen hoe ik het stageproject zal aanpakken.

Achtergrond

In dit onderdeel staat de achtergrond en context van mijn opdracht beschreven. Dit maakt duidelijk waarom de opdracht in de eerste plaats uitgevoerd zal worden.

Nitor

Nitor omschrijft hun eigen missie als volgt: *"Niet elke medewerker komt dagelijks in contact met een computer. Bedrijven streven meer en meer naar digitale workflows. Daarom levert Nitor mobiele oplossingen die de werkprocessen vereenvoudigen en optimaliseren. Ze luisteren naar de behoeften van bedrijven en ontwerpen technologie die naadloos integreert, waardoor hun team effectiever functioneert. Het resultaat? Een gestroomlijnde dataflow, minder administratie en minder fouten.*

Met diepgaande expertise in zowel Apple- als Android -systemen hanteert Nitor een proactieve aanpak, waardoor ze altijd vooroplopen met updates. Zo garanderen ze dat de applicaties van hun klanten altijd optimaal functioneren en hun workflow nooit onderbroken wordt."[1]

De bouwsector

Nitor is gevestigd in de zone Kamp C in Westerlo. Alle bedrijven in Kamp C hebben een ding gemeenschappelijk: Ze werken aan een duurzame bouwsector. Onder andere daarom wil Nitor een app maken die nuttig kan zijn voor de bouwsector.

Ik zal dus tijdens mijn stage hiervoor een Proof of Concept (POC) uitwerken. Het doel is om een mobiele app te maken die werfleiders kan ondersteunen met het melden en opvolgen van problemen.

Scope

In dit onderdeel wordt de scope van het project beschreven. Dit zal duidelijk maken wat belangrijk en noodzakelijk is om het project te laten slagen, welke extra's het resultaat nog beter maken en wat niet geïmplementeerd zal worden.

Must haves

Functionele eisen

De functionele vereisten zijn eerder beperkt. Dat is zo omdat mijn opdrachtgever absolute prioriteit geeft aan het onderzoeken van mogelijkheden en het tot in de puntjes uitwerken van details. Meer informatie hieromtrent staat omschreven in de volgende categorie. Er zal dus gefocust worden op volgende functionaliteiten:

[1] Bron: Officiële website Nitor, geraadpleegd op 23 februari 2026 via https://nitor.be.

- Registratie van een incident
    - Maken van een foto
    - Capteren van de geolocatie
    - Automatisch toevoegen van datum + tijd
    - Automatisch toevoegen van gebruiker
    - Toevoegen van notities
- Tonen van reeds gelogde incidenten
- Afhandelen van incidenten
    - Automatisch toevoegen van datum + tijd
    - Automatisch toevoegen van gebruiker
    - Toevoegen van oplossing
    - Wijzigen van de status
- Uitlezen van informatie uit een NFC-tag

Technische eisen

Mijn opdrachtgever vindt het belangrijk dat er veel tijd en aandacht besteed wordt aan het uitwerken van technische details. De belangrijkste technische vereisten zijn:

- Gebruik van react-native met implementatie van expo
- Gebruik van de local storage van het toestel zodat je geen netwerk nodig hebt (met uitzondering van het inloggen)
- Gebruik van één externe identity provider (bvb Google, Meta, ...)
- Aanmaken van .apk voor installatie op Android
- Definiëren van coding guidelines + automatisch afhandelen ervan
- Implementeren van duidelijke foutafhandeling

Belangrijk is dus dat er nagedacht wordt details. Zo zal ik bijvoorbeeld moeten testen met oudere Android versies, ontbrekende machtigingen, ...

Design eisen

- Gebruik van de Nitor look & feel
- Een duidelijke UX/UI die rekening houdt met de omstandigheden waarin de applicatie gebruikt zal worden.

Nice to haves

Indien het project zeer vlot verloopt en er meer tijd over blijkt te zijn, is er ruimte voor enkele uitbreidingen. Voorbeelden zijn:

- AI assist (bijvoorbeeld auto-tag suggesties op basis van simpele regels)?
- Publishen van applicatie naar TestFlight voor iOS
- Gebruik van meerdere identity providers

- Implementatie van automatische unit tests
- Synchronisatie van data naar een database wanneer wifi beschikbaar is.

Indien er voldoende tijd over is, ben ik vrij om naar eigen wens nog interessante zaken toe te voegen of te onderzoeken.

Buiten de scope

Een voorbeeld van wat buiten de scope is, is het ontwikkelen van zaken zoals een admin-portaal of user management. Dit zijn zaken die Nitor bijna rechtstreeks kan kopiëren van andere projecten. Het is dus niet zo nuttig om hier tijd in te steken.

Het doel van de stageopdracht is echt om bij te leren op het vlak van native mobile development en om de uiterste mogelijkheden van deze case te onderzoeken en uit te werken naar een kleine maar krachtige POC.

Onderzoek

Voordat ik een volledige analyse kan maken is het belangrijk dat ik over de juiste informatie beschik. Daarom is het noodzakelijk om op onderzoek te gaan. In dit hoofdstuk staat mijn onderzoek en de resultaten daarvan gedocumenteerd.

Interviews

Ik heb twee personen geïnterviewd die actief zijn binnen de bouwsector. Deze twee personen vertegenwoordigen in het totaal vijf bedrijven.

Guy

Guy is een zelfstandige projectontwikkelaar. Hij bouwt bedrijfsvastgoed op duurzame sites. De units worden dan verhuurd of verkocht.

Samenvatting van wat Guy me vertelde:

Wie heeft dat gedaan?

Er is altijd de centrale vraag "Wie heeft dat gedaan?". Voor de persoon die het probleem moet oplossen, in het geval van Guy is hijzelf dat, kan dit zeer waardevolle informatie zijn.

Een probleem of beschadiging wordt vaak opgemerkt door verschillende mensen. Voorbeelden hiervan zijn: de klant, een werknemer, de werfleider, de aannemer, … In de praktijk gebeurd het geregeld dat iemand die aanwezig is op de werf een incident ziet gebeuren. Het zou handig zijn moest deze persoon meteen een foto kunnen nemen en het probleem opslaan in de app.

Dit zou het noodzakelijk maken dat verschillende personen kunnen inloggen. Guy denkt dat multi-user een must is.

Terugkoppeling

Momenteel is de realiteit dat iemand die een probleem spot, zoals de werfleider, dit vaak integraal doorstuurt naar de verantwoordelijke (bv de projectontwikkelaar). De terugkoppeling omtrent dit probleem en een eventuele oplossing is vaak een heel gedoe. Een gestructureerde app zou hiervoor een oplossing kunnen zijn.

Handmatige geolocatie

Het automatisch registreren van de geolocatie via de GPS sensor in het toestel is een mooi startpunt. Echter is het volgens Guy belangrijk om ook handmatig de locatie te kunnen aanpassen of corrigeren (details). Als bijvoorbeeld een foto vanop 10m afstand gemaakt wordt kan de fotograaf al aan een andere unit staan waardoor het incident volgens de geolocatie bij een andere unit hoort.

Opvolgingssysteem

Guy ziet het veel groter dan enkel het monitoren van incidenten. Voor hem zou het handig zijn dat er in de app buiten problemen ook en soort checklist bijgehouden kan worden. Het zou dan gaan over wat gebeurd is en wat nog niet. Hij denkt dat werfleiders, maar ook hijzelf als projectontwikkelaar een centraal planningssysteem mist dat over de verschillende betrokken partijen heen gaat.

Ook sprak hij over een mogelijke integratie met externe diensten. Hijzelf houdt bijvoorbeeld al zijn documenten betreffende dat project zoals vergunningen, foto's, contracten, … bij in een Google Drive map. Het zou handig zijn moest dit op de een of andere manier geïntegreerd kunnen worden.

Conclusie

Guy zou meteen voor een app kiezen boven het handmatig nemen en versturen van een foto. Maar het systeem zou dan wel goed in elkaar moeten zitten en volledig genoeg moeten zijn. Hij denk dat enkel een overzicht van incidenten dat beheerd wordt door 1 partij (bijvoorbeeld werfleider of projectontwikkelaar) onvoldoende is.

Ellen

Vertegenwoordigde Architron, Van Roey Services, Iftech, Van Roey (aannemer)

Samenvatting van wat Ellen me vertelde:

Te algemeen

Ellen is van mening dat mijn onderzoeksvraag te breed is. De bouwsector is zeer breed en binnen verschillende sectoren verloopt de werking heel anders.

Zo werkt ze bijvoorbeeld samen met zowel grote als kleine bedrijven, in verschillende specialisaties binnen de bouwsector.

Archisnapper

In veel grote bouwbedrijven wordt de (eerder dure) software Archisnapper gebruikt. Deze software maakt het mogelijk dat er een takenlijst met prioriteiten aangemaakt kan worden die werkmannen op een tablet kunnen raadplagen. Ze moeten als de klus klaar is dan een foto nemen met de tablet. Waar Guy over sprak betreft een soort van takenlijst, wordt volgens Ellen beantwoord door een tools zoals Archisnapper.

GBS (Gebouw BeheersSystemen)

Soort van ticketsysteem waar problemen binnen grote gebouwen (ziekenhuizen, appartementen, …) in aangemaakt kunnen worden. Een centrale dienst zorgt er dan voor dat de juiste bedrijven/techniekers worden aangesproken om het probleem te gaan fixen. Ook zaken zoals water en energie zitten hierin.

NFC-technologie

Ellen kon hier niet veel over kwijt, maar ze denkt dat Iftech dit gebruikt (HVAC bedrijven).

V**erzekeringskwesties**

Toch denkt Ellen dat er (bv binnen Van Roey) nog ruimte is voor een app. Dit gaat ook dezelfde richting op, als wat Guy zei. Ze ziet zo'n app dan eerder als een plaats voor verzekeringskwesties. Het zou handig kunnen zijn als

- iemand iets beschadigd op de werf. Het zou een tijdswinst zijn als er dan een foto van gemaakt zou kunnen worden en wat informatie kan worden ingevuld en vervolgens de verzekeringspapieren automatisch worden opgemaakt.
- gehuurd materiaal zoals bijvoorbeeld een kraan kapot is, dit in een app kan worden ingegeven en dat een centrale dienst dan dit probleem kan oplossen in plaats van dat er manueel moet worden rondgebeld naar o.a. de firma van die kraan.

Conclusie

Ellen denkt dat er zeker ruimte is voor een app, maar dat de onderzoeksvraag en doelgroep beter bepaald moet worden alvorens de exacte noden onderzocht kunnen worden.

Bespreking

Na de interviews heb ik de resultaten gepresenteerd aan mijn opdrachtgever en aan de rest van het team. De informatie die ik gegeven heb komt in grote lijnen overeen met eerder research dat ze eerder al eens gedaan hadden.

Aangezien de stageopdracht een Proof of Concept is, zullen we de vraag naar de extra functionaliteiten voorlopig nog niet meenemen in mijn originele opdracht. Om de kwaliteit van de applicatie te kunnen blijven garanderen zullen we de scope houden zoals beschreven in het onderdeel “Scope”. Echter zal ik de feedback betreft het handmatig aanpassen van de geolocatie wel meenemen.

Voor deze functionaliteit moet ik enkele zaken toevoegen aan de applicatie die niet in de originele scope omschreven staan. Meer uitleg hierover staat vermeld in het onderdeel “Technische analyse”.

Indien ik tijd over heb, en de scope uitgebreid wordt – zoals eveneens omschreven staat in het onderdeel “Scope” – zullen we wel rekening houden met de andere feedback die uit de interviews gekomen is.

Analyse

Dit hoofdstuk gaat over de analyse. Het bestaat uit 2 delen:

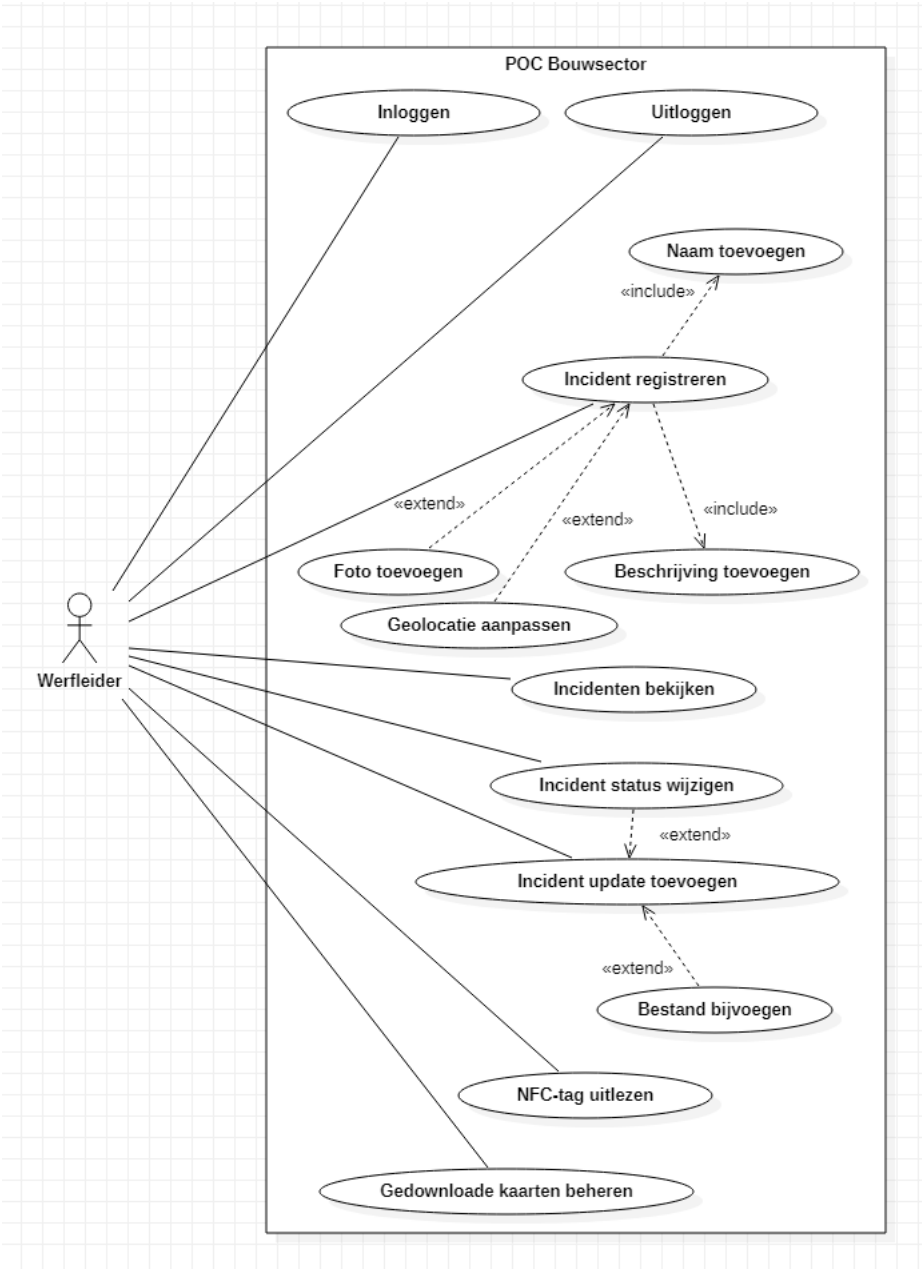
- **Design analyse:**
  In dit onderdeel wordt getoond hoe de applicatie eruit zal zien. Enerzijds worden visuele prototypes getoond, anderzijds worden use cases uitgelegd en wordt de datastructuur beschreven.
- **Technische analyse:**
  Dit onderdeel gaat over de tools die gebruikt zullen worden. Het is belangrijk om te experimenteren met nieuwe technologieën en tegelijkertijd voldoende aan te sluiten bij de werkmethodes van Nitor. Nog belangrijker is uiteraard dat de juiste tools worden gekozen voor de aard en omvang van het specifieke POC.

Design analyse

Use Case Diagram (UCD) en User Stories

Het UCD is bedoeld om de verschillende gebruikers van het systeem en de acties die zij binnen de app kunnen uitvoeren te definiëren. Aangezien het gaat om een POC is dit diagram eerder beperkt.

Use Case Diagram:



Met behulp van user stories worden deze functionaliteiten verder toegelicht. Ze worden beschreven vanuit het standpunt van een niet-technische gebruiker. Op deze manier wordt er extra nagedacht of de functionaliteiten goed en logisch in elkaar zitten voor de gebruiker.

User Stories:

Als een werfeider wil ik...

... inloggen en uitloggen zodat ik mezelf kan identificeren.

... een incident registreren zodat ik problemen kan melden.

... foto's toevoegen zodat ik bewijs of visuele ondersteuning kan opslaan.

... de geolocatie aanpassen zodat ik de exacte plaats van het incident kan opslaan.

... een naam toevoegen zodat ik snel kan zien waar het incident over gaat.

... een beschijving toevoegen zodat ik het probleem kan beschrijven en extra informatie kan meedelen.

... incidenten bekijken zodat ik een overzicht krijg van alle incidenten op de werf.

... een update toevoegen aan een incident zodat ik deze kan documenteren.

... de status van een incident wijzigen zodat ik weet of ik nog actie moet ondernemen.

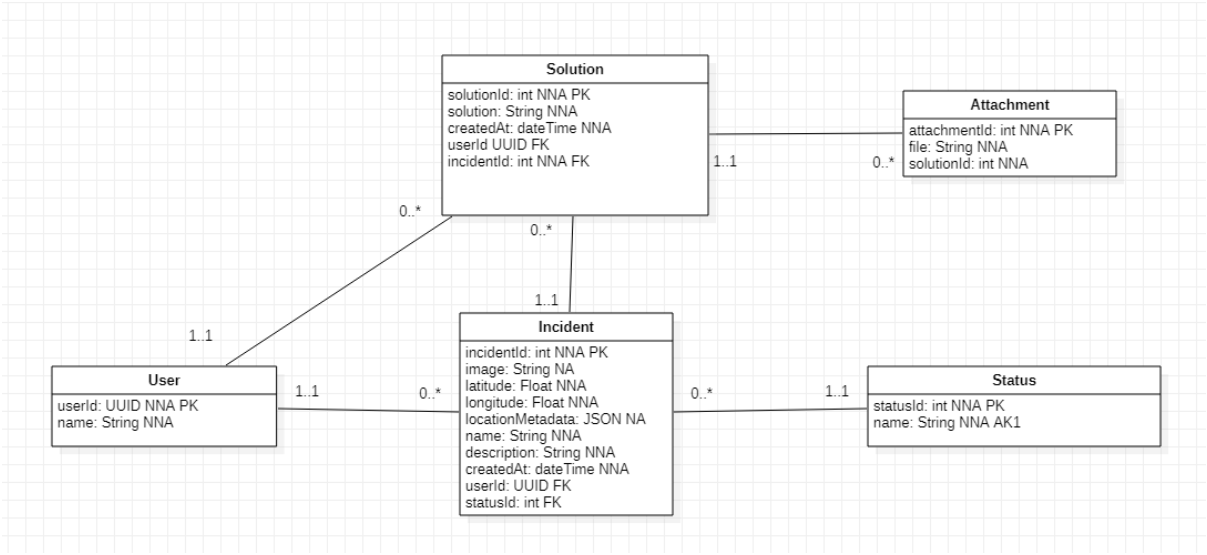
... een bestand bijvoegen zodat bijlagen worden opgeslagen.

... een NFC-tag uitlezen zodat ik de informatie van installaties kan raadplegen.

... offline kaarten beheren zodat ik de app zonder internet kan gebruiken en ruimte op mijn apparaat kan beheren.

Data model

Het data model is bedoeld om een overzicht te krijgen van welke data in de applicatie moet worden opgeslagen.

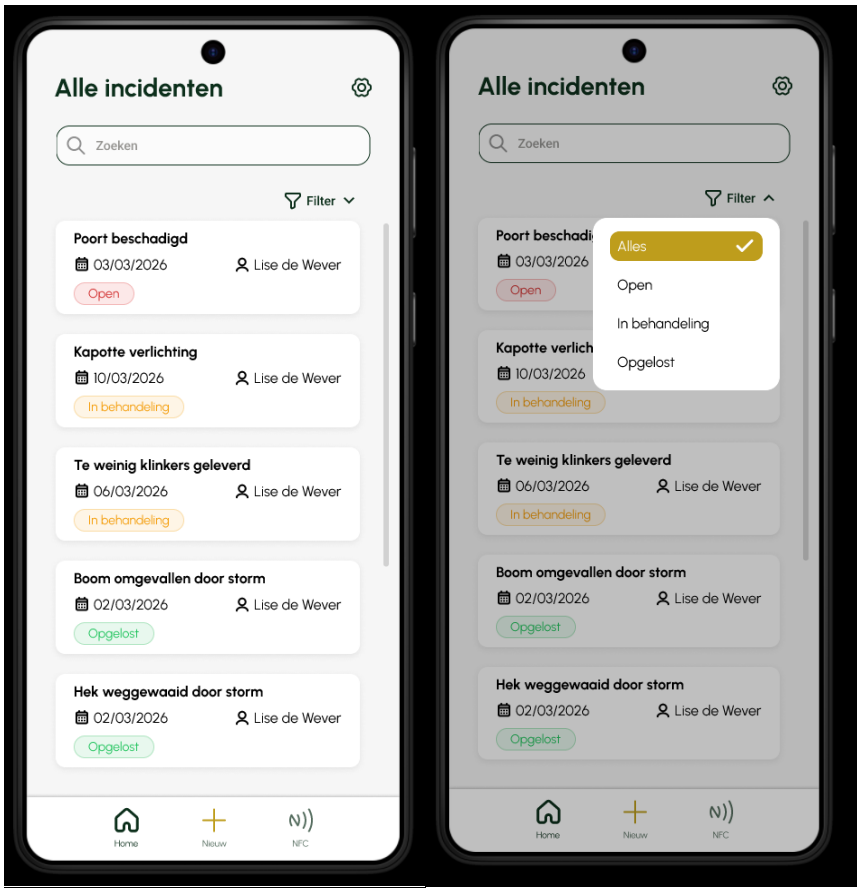


Prototypes

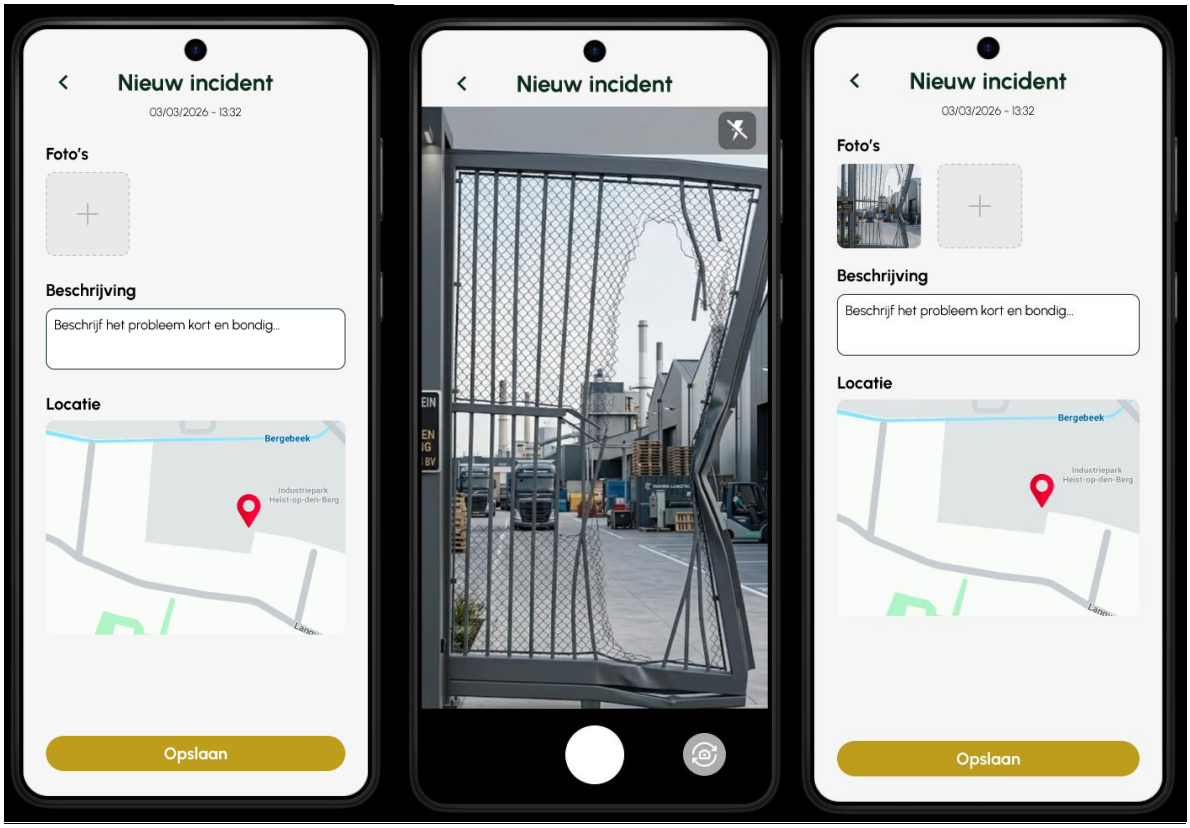
De prototypes zijn bedoeld om te laten zien hoe het eindresultaat er visueel in grote lijnen zal uitzien. Uiteraard is het mogelijk dat hier tijdens de realisatiefase licht van afgeweken zal worden.

<u>Aanmeldscherm:</u>

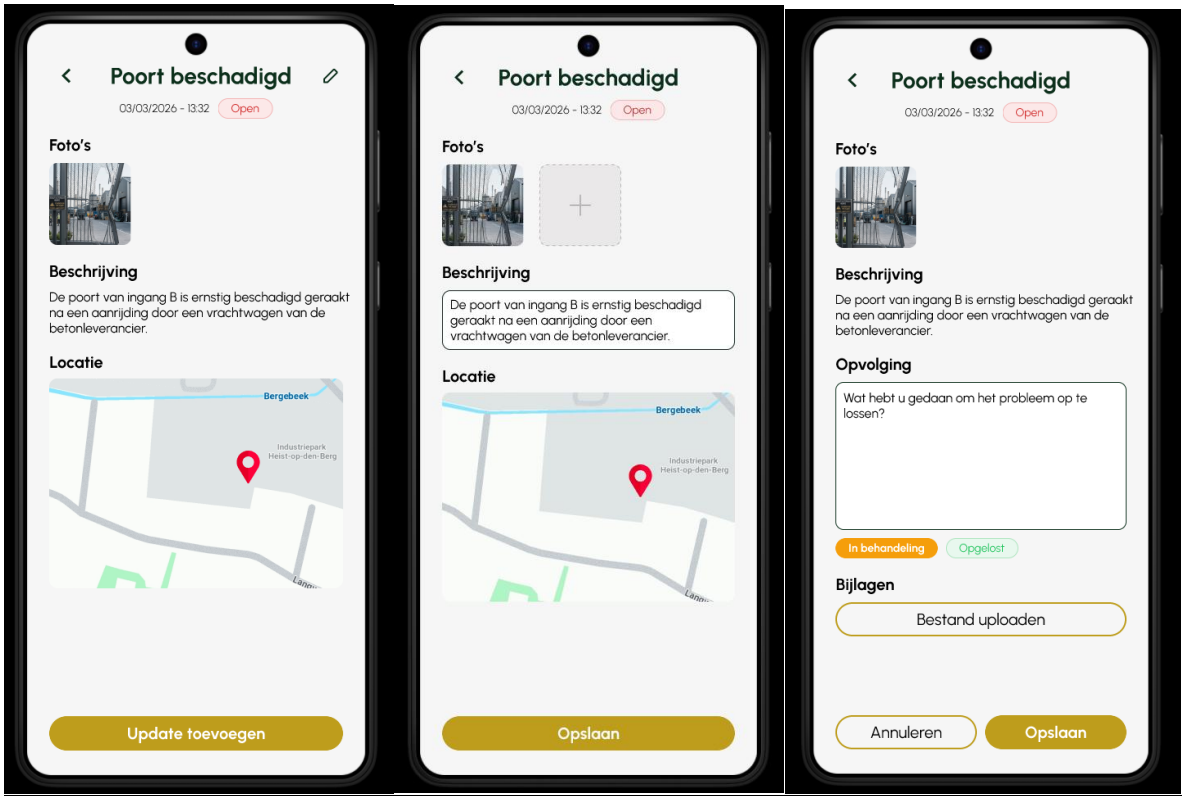
Incidenten overzicht:

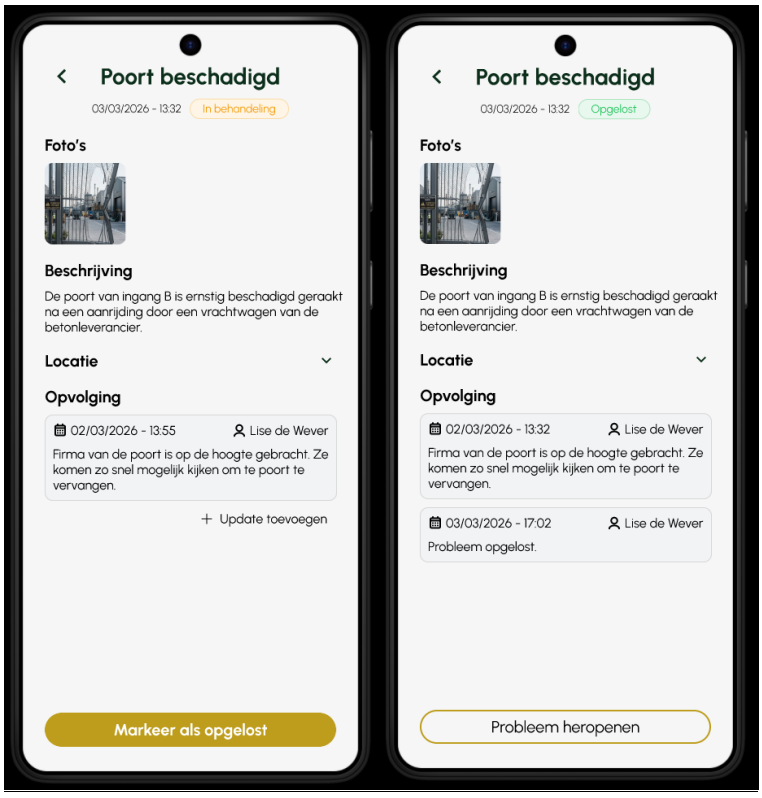


Nieuw incident:

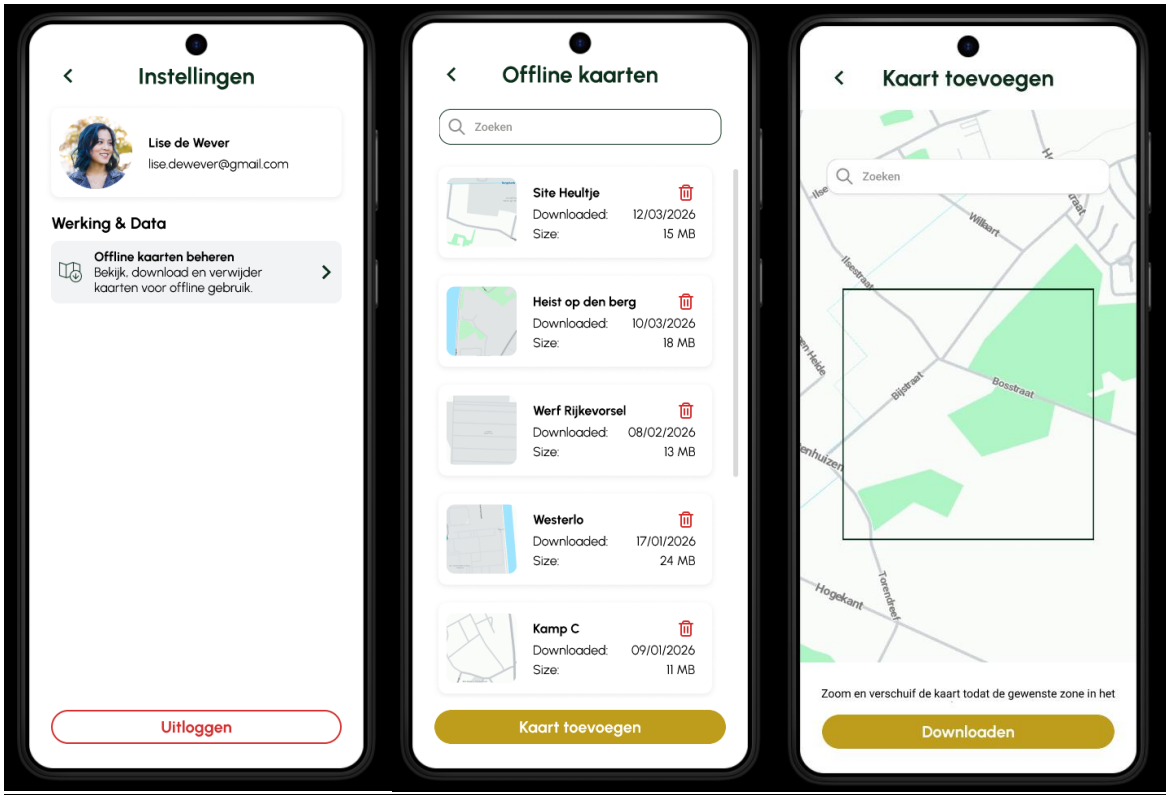


Incident details:





Offline kaarten beheren:



Technische analyse

Framework

Bij Nitor wordt er met Bare React Native gewerkt. Ze bouwen alles van scratch in combinatie met hun eigen, zelfgeschreven framework. Dit omdat een framework zoals Expo enkele jaren geleden niet geschikt was voor complexe apps.

De laatste jaren is Expo echter sterk verbeterd. Tegenwoordig wordt vanuit de documentatie van React Native zelfs het gebruik van een framework zoals Expo aangeraden.

Aangezien Expo de development enorm zal versnellen en ik er al bekend mee ben, zal ik dit gebruiken voor mijn stageopdracht. Zo kan ik een fris inzicht binnen Nitor brengen door te tonen hoe je snel een goede app kunt bouwen met **Expo**.

State

Om van een simpel iets naar een professionele app te gaan, is het belangrijk om aan state management te doen. Bij Nitor wordt hiervoor – en voor nog wat andere zaken – Redux gebruikt. Redux is al jaren de standaard en was vroeger ook de enige goede methode om te gebruiken voor complexe apps. Het heeft echter een relatief hoge leercurve en vereist best veel tijd om boilerplate code te schrijven.

Daarom zal ik gebruik maken van een moderner alternatief: **Zustand**. Deze tool wordt de laatste jaren steeds meer gebruikt. Volgens mijn research zou Zustand zelfs geschikt moeten zijn voor complexe apps. Voor deze POC is het zeker voldoende. Redux zou zelfs wat overkill zijn. Door voor Zustand te kiezen kunnen we de proef op de som nemen en kan in een latere fase ontdekt worden of Zustand inderdaad geschikt is voor complexe apps.

Routing
Aangezien ik gebruik maak van Expo is de keuze omtrent routing snel gemaakt. Ik zal gebruik maken van **expo-router** aangezien dit naadloos aansluit bij Expo development. Gebruik maken van een andere tools zoals react-navigation of react-router zou hier een vreemde keuze zijn.

Offline storage
Tot 2021 was AsyncStorage de enige fatsoenlijke manier om met offline storage om te gaan. Nu is er echter MMKV.

AsyncStorage is traag. Het heet "Async" omdat het asynchroon werkt via de oude React Native "Bridge". Elke keer als je data ophaalt, moet JavaScript wachten op de native kant, wat micro- tot milliseconden vertraging oplevert.

Daarom zal ik gebruik maken van een veel modernere tool: **react-native-mmkv**.

Het is onder de motorkap gebouwd door Tencent (het bedrijf achter WeChat, een app met een miljard gebruikers). Het is dus extreem 'enterprise-proof'. Het maakt ook gebruik van JSI (JavaScript Interface) en C++. Hierdoor is het synchroon. Als je data opvraagt, heb je het *direct*, in dezelfde milliseconde. Uit benchmarks blijkt dat MMKV tot wel 30 keer sneller is met het wegschrijven en lezen van data dan AsyncStorage.

Camera
De meest voor de hand liggende opties voor het aansturen van de camera zijn Expo-Camera en React-Native-Vision-Camera. Ik zal gebruik maken van **expo-camera**. Dit omdat het de gemakkelijkste optie is voor gebruik met Expo. Voor de simpele eis *"Maken van een foto"* is het ook 100% voldoende.
Als we later geavanceerde “vision” functionaliteiten zouden willen toevoegen en bv. een AI-model willen runnen dat live meekijkt door de camera (bijvoorbeeld om automatisch te herkennen dat er een 'kapotte lamp' in beeld is voordat je de foto neemt), dan is expo-camera niet krachtig genoeg.

In dat geval zal er overgestapt moeten worden, maar er is voorlopig niets dat erop wijst dat dat nodig is.

Locatie
Ik zal gebruik maken van **expo-location**. Dit is perfect voor het *"Capteren van de geolocatie"* op het op het moment dat je een nieuw incident aanmaakt. Dit zal opnieuw het beste samenwerken met Expo.

De enige denkpiste om naar alternatieven te zoeken, is dat als we in de achtergrond een hele route willen bijhouden, we beter voor react-native-background-geolocation (betalend) gaan. Maar aangezien dat dat voorlopig totaal niet van toepassing is, is dat voor nu niet nodig.

NFC
Voor NFC biedt Expo helaas geen bibliotheek. Het heeft dit niet ingebouwd. Om NFC te laten werken op iOS (Apple is hier heel streng in), moet je permissies in de core van het iOS-systeem schrijven. In 'Bare' React Native doe je dit handmatig in Xcode. In Expo moet je dit doen via een Config Plugin die deze bestanden genereert tijdens de Prebuild.

Ik zal dus **react-native-nfc-manager** moeten gebruiken. Wel ga ik moeten uitzoeken hoe ik een bestaande Config Plugin hiervoor installeer zodat ik niet uit het Expo-ecosysteem hoef te stappen.

Authenticatie

Supabase en firebase zijn twee opties die het inloggen met Google/Meta super makkelijk maken zonder dat ik zelf een backend hoef te schrijven.

Firebase is gemaakt door Google. Het zit helemaal in haar eigen ecosysteem en gebruikt onder andere google cloud. Het is overigens een *NoSQL* database, wat voor complexe data vaak rommelig wordt.

Een andere mogelijkheid is Supabase. Het is een Open-source alternatief. Het heeft perfecte TypeScript integratie: aangezien het automatisch de TypeScript types van de database kan genereren, wat naadloos aansluit bij mijn .tsx bestanden. Daarom zal ik **Supabase** gebruiken.

Lokale bestandsopslag

Voor het opslaan van de foto's hebben we lokale bestandsopslag nodig. Daarom zal ik **expo-file-system** gebruiken. Dit is opnieuw de ingebouwde bibliotheek van Expo. Daarom zal het het beste samenwerken met Expo. Ik gebruik dan expo-file-system om die foto definitief in een mapje op de telefoon te bewaren. Vervolgens houden we in MMKV alleen het *pad* (de URL-string) naar die opgeslagen foto bij.

Kaarten

Voor het weergeven van kaarten (nodig voor het manueel aanpassen van de locatie) zijn er 2 grote opties. Google Maps en Mapbox. Het makkelijkste om te integreren is Google maps. Helaas is deze API niet goed genoeg voor mijn noden. Mapbox vult hier de gaten op, maar heeft een steilere leercurve.

Zo heef Google maps geen functionaliteiten om kaarten te downloaden. Dit strijkt de "Offline" eis tegen de haren in. Mapbox daarentegen, heeft ingebouwde functionaliteiten voor het downloaden van kaarten. Hiervoor moet ik, zoals eerder vermeld, wel een functionaliteit toevoegen aan de scope. Het gaat hier om het beheren van gedownloade kaarten.

Verder heeft Google maps minder styling opties dan Mapbox om het uiterlijk van de kaart te personaliseren. Ook heeft Mapbox (dankzij OpenStreetMap-data) meer kennis van bv. landelijke gebieden of nieuwe industrieterreinen.

Ik zal dus gebruik maken van **Mapbox**.

Helaas heeft het gebruik van Mapbox nog wel een ander nadeel. Mapbox is een zware native bibliotheek die niet samenwerkt met Expo go. De standaard Expo Go app bevat de C++ en Java code voor Mapbox simpelweg niet. De app zou direct crashen.

Dit zorgt ervoor dat ik niet via Expo Go kan ontwikkelen. Uiteraard is dat geen probleem aangezien ik via **Expo prebuilds** kan werken. Ik zie dit als een kans om bij te leren!

Bouwen en publiceren

Ik zal gebruik maken van **EAS (Expo Application Services)**. Dit houdt in dat ik mijn apps in de cloud kan builden ipv op mijn laptop. Dit is veel sneller en gemakkelijker.

Coding guidelines
Het was ook een eis om coding guidelines te definiëren en automatisch af te handelen. In de eerste plaats zal ik gebruik maken van **ESLint** en **Prettier**. Eslint controleert of de code aan de afgesproken regels voldoet (bijv. geen ongebruikte variabelen). Prettier daarentegen, formatteert de code automatisch zodat alles er hetzelfde uitziet (spaties, quotes, etc.). Indien de ontwikkeling zeer snel en vlot verloopt, zal ik dit nog verder onderzoeken om eventueel nog meer en betere automatische afhandeling van coding guidelines te implementeren.

Unit testing
Ik zal gebruik maken van **React Native Testing Library (RNTL)** voor het testen van UI en logica. Ter ondersteuning zal ik **Jest** gebruiken als test-runner. Dit zal allemaal werken via een Azure DevOps pipeline.

Expo/React versie
De React Native versie staat vast aan de Expo versie. Expo kiest zelf welke versies hij gebruikt zodat het zeker compatibel is.

Wat betreft de Expo versie: de **laatste versie** zou perfect moeten werken. Dit is trouwens ook de veiligste keuze wanneer de app later gepubliceerd zal worden.

Conclusie
De meeste van de bibliotheken die ik nodig heb, werken out of the box met Expo. Enkel de NFC-bibliotheek en Mapbox vallen uit het rijtje. Ze zijn echter wel volledig compatibel met Expo, maar hebben extra configuratie nodig en zorgen ervoor dat ik van Expo Go moet afstappen.

Deze zaken zorgen voor een extra uitdaging, maar die zou zeker en vast wel doenbaar moeten zijn.

Planning
In dit hoofdstuk wordt de planning besproken. Aangezien ik via de Agile-principes zal werken, is het momenteel niet mogelijk om een volledig uitgeschreven planning op te maken. Daarom zal ik me richten op enkele vaststaande tussentijdse activiteiten alsook een algemeen idee van hoe ik het project planningsgewijs zal aanpakken.

Vaste activiteiten
Kick-off meeting
De eerste drie weken van mijn stage zijn bedoeld voor analyse. Als hoogtepunt van deze periode, staat in het begin van week drie een kick-off meeting gepland. In deze vergadering zal ik mijn projectplan presenteren voor mijn stagebegeleider en stagementor. Ook zullen er enkele praktische zaken besproken worden.
Verder dient deze vergadering als een moment om feedback te ontvangen over mijn project plan. Deze kan ik later dan nog verwerken.

Eerste meeting op school
In week vier staat er een meeting gepland met mijn stagebegeleider en enkele medestudenten. Ik zal dan opnieuw mijn projectplan presenteren.

Tussentijdse evaluatie
In week zes zal er een tussentijdse evaluatie plaatsvinden. Deze evaluatie zal worden ingevuld door de stagementor in het stageportaal. Het is in de eerste plaats bedoeld als feedback voor mij. Zo weet ik wat ik al goed doe en waar ik nog in kan verbeteren.

Tweede meeting op school
In week acht staat er opnieuw een meeting op school gepland. Ik zal dan presenteren wat ik al gerealiseerd heb, op basis van een eerste versie van mijn realisatiedocument. Ook zal ik overlopen wat er nog allemaal moet gebeuren.

Verder zal ik met de stagebegeleider de resultaten van de tussentijdse evaluatie bespreken.

Deadline documenten
Week 12 staat als de deadline voor het vervolledigen van alle documenten die ik gemaakt heb. Dit dient als een portfolio van mijn stage.

Algemene planning
Zodra de realisatiefase begonnen is, zal ik starten met het opzetten van mijn project. Ik heb voor mezelf een stappenplan voorbereid van alles wat ik moet doen om mijn ontwikkelomgeving op te zetten, het project aan te maken en alle belangrijke bibliotheken te installeren.

Zo zal ik eerst proberen om een lege maar werkende versie te maken waar alles inzit wat ik nodig heb. Dit verkleint de kans dat ik halverwege de realisatiefase tegen dependentieproblemen oploop.

Vervolgens zal ik starten met het ontwikkelen van functionaliteiten. Ik zal beginnen met de functionaliteiten die de hoogste prioriteit hebben. Zaken met een lagere prioriteit zullen pas later aan bod komen.

In het geval ik in de loop van de realisatiefase merk dat ik tijd over ga hebben, zal ik in samenspraak met mijn stagementor de scope van de opdracht uitbreiden.