Conceptfase - Adviesrapport

Groep F1 Coding:

Stijn Van Riel
Matthias Reynders
Brent Belien
Walid Amhaini
Ruben Vanderborght
Jens Weyen

Inhoudstafel

1. Inleiding

Dit adviesrapport bevat het onderzoekswerk dat we verricht hebben om de applicatie voor Elision te kunnen maken. We beginnen met het verwachte resultaat op te stellen, bekijken dan de benodigde bouwstenen en tonen de verwachte user experience. Tot slot kijken we ook toekomstgericht naar hoe de applicatie ook gebruikt zal kunnen worden na de corona crisis.

Graag willen we Bram Heyns, Marnix Meuwissen en Benjamin Camps danken voor hun hulp en feedback tijdens het opstellen van dit adviesrapport.

2. Shared Vision

2.1. Verwacht Resultaat

Aan het einde van de realisatiefase zullen wij voor Elision een reservatiesysteem opleveren dat schaalbaar is met de nodige vereisten. Er wordt rekening gehouden met de GDPR maatregelen omwille van de data die onze applicatie verwerkt om te functioneren. Ook zullen wij de applicatie voorzien van een dashboard met nuttige informatie voor de gebruiker zoals: welke zone van het gebouw is druk en hoe lang/hoe vaak is het daar meestal druk.

2.2. Project Requirements

Onze oplossing moet voldoen aan de volgende basiscriteria:

- Reserveren van stoelen op een specifieke datum/uur.
- Lunch overzichtelijk plannen om drukte te vermijden.
- Mogelijkheid om klanten te registreren voor ze te ontvangen.
- Gebruiksvriendelijk en beschikbaar voor elke werknemer.
- Admin tools voor verschillende hiërarchische niveaus.
- Aanpasbaarheid/schaalbaarheid van de applicatie om de applicatie ook na de coronacrisis relevant te houden.
- Dashboard met statistieken om gebruik in kaart te brengen.

2.3. Risico’s

Tijdens elk IT project zijn er verschillende risico’s waar rekening mee gehouden moet worden. Om deze risico’s te beperken moet er vooraf een onderzoek gedaan worden om deze te identificeren. Nadat we weten welke risico’s er allemaal zijn kunnen we preventief oplossingen bedenken om deze risico’s te beperken.

Potentiële risico’s

Fysieke schade

Op het kantoor kan er natuurlijk altijd iets misgaan. Een sensor die defect gaat door hitte, een microcontroller wordt nat tijdens het kuisen, ...

Weinig business value

Het afgeleverde product moet waarde leveren aan de klant. Na een lange periode van development is het natuurlijk de bedoeling dat Elision er ook effectief iets kan met doen.

Niet gebruiksvriendelijk

Een van de grootste problemen met de huidige applicatie was dat de UI heel ongebruiksvriendelijk was voor gebruikers. De applicatie moet intuïtief zijn en beschikbaar zijn voor iedereen.

Niet alle hardware beschikbaar

Om een werkende applicatie af te leveren zijn er natuurlijk verschillende hardware onderdelen nodig. Enkele camera’s, sensoren en andere IOT producten die het mogelijk maken om de applicatie af te werken. Indien we niet kunnen samenkomen tijdens de realisatiefase moet de hardware ook bij de correcte student geraken.

Slechte communicatie tussen het team en opdrachtgever

Voor de uitwerking van een IT project is er veel communicatie nodig tussen het team maar ook zeker tussen de opdrachtgever en het team. Om dit vlot te laten verlopen zijn er enkele maatregelen nodig.

Oplossingen

Fysieke schade

Voorkomen is natuurlijk altijd beter als genezen, daarom is het een prioriteit om de sensoren en camera's te beschermen tegen hun omgeving door ze op veilige plaatsen te zetten. Ook moet er een duidelijke overeenkomst zijn tussen de opdrachtgever en het projectteam. Het is vanzelfsprekend dat wij als leverancier en ontwikkelaar van de applicatie niet verantwoordelijk kunnen zijn voor eventuele schade aan de sensoren door slecht- of misbruik ervan.

Weinig business value leveren

Er moet een duidelijk beeld zijn van de situatie waar het bedrijf zich nu in bevindt. Zo kunnen we duidelijk maken of ons product ook effectief een meerwaarde brengt. Op dit moment zijn er minder gemotiveerde werknemers omdat ze vaker vanuit thuis werken. Om deze productiever te maken en vaker naar kantoor te krijgen in hun voordeel maar ook zeker in dat van de hele Xplore group is het essentieel om een goed werkende applicatie te maken.

Niet gebruiksvriendelijk

Om een van de grootste problemen van de huidige applicatie op te lossen zijn er enkele maatregelen nodig. De applicatie moet vaak getest worden tijdens development door ons maar ook zeker door iemand die de applicatie nog niet gezien heeft. Zo krijgen we een beter beeld van hoe een nieuwe gebruiker de applicatie gebruikt. Daarnaast moet er ook rekening gehouden worden met toegankelijkheid, zodat alle mogelijke gebruikers zonder problemen de applicatie kunnen gebruiken.

Niet alle hardware beschikbaar

De vereiste sensoren moeten duidelijk gecommuniceerd worden met Elision zodat zij deze beschikbaar kunnen stellen voor ons voor gebruik en de productie van de applicatie. Dit moet natuurlijk gebeuren in samenspraak met Elision.
In het geval dat er tijdens de realisatie van thuis uit moet gewerkt worden, moet deze hardware ook op de correcte locatie geraken.

Communicatie tussen het team en opdrachtgever

Gedurende heel het project is het van groot belang om een duidelijke communicatie te hebben tussen de verschillende teamleden en de opdrachtgever. Tijdens de conceptfase hebben we besloten om op iedere vrijdag een online meeting te doen om de stand van zaken te bespreken binnen het team. Als we feedback willen of vragen hebben kunnen we Elision altijd bereiken via mail. Hierna kunnen we verder met de gegeven feedback of eventueel een online meeting doen om de feedback te bespreken.
Tijdens de realisatiefase zullen regelmatige meetings met Elision ook essentieel zijn om een goed eindproduct af te leveren.

3. Bouwstenen

Onze applicatie zal opgebouwd zijn uit verschillende bouwstenen, die samen komen tot het gehele product.

3.1. Hosting

Om de webapplicatie overal beschikbaar te maken, gemakkelijk te scalen en deze stabiel te doen draaien zouden we graag werken in de cloud. Dit gaat in het algemeen veel voordelen met zich meebrengen. Hier bekijken we enkele cloud providers en hun specifieke voor- en nadelen.

Google Cloud

Binnen google cloud wordt er een product aangeboden genaamd App engine. Deze toepassing kan eenvoudig web apps draaien en opvolgen. De prijs wordt bepaald aan de hand van het aantal uren actief en de virtuele CPU kracht.
Vele mensen raden het aan omdat het snel en eenvoudig in gebruik is. Echter kwamen we na onderzoek ook veel beperkingen tegen. Aangezien we met Java en Python gaan werken hebben we hier ook specifiek naar gekeken.

Voordelen	Nadelen
- Zeer gebruiksvriendelijk	- Eigen shell / Eigen commands
- Stabiel en snel (datacenter in België etc)	- Ondersteund enkel python modules die in python geschreven zijn (⅓ niet bruikbaar)
- Prijs naargelang gebruik (gratis bij weinig gebruik)	- Java applicaties kunnen enkel met een thread werken. (+ nog andere java beperkingen)
	- Geen toegang tot file system, stagen via command line.

Google cloud heeft een groot aantal beperkingen voor ons project. De webapplicatie kan gewoon crashen door bijvoorbeeld niet ondersteunde modules door app engine of door java limieten die google opstelt. Het is echter wel een van de performantste oplossingen.

Azure (Microsoft)

Microsoft biedt binnen Azure een service aan genoemd App service. Deze service kan een web applicatie zoals wij maken eenvoudig beheren. Azure is het meeste gebruikte cloud platform op de markt momenteel. Hieronder vindt u de voor en nadelen ervan.

Voordelen	Nadelen
- 10 gratis web apps met beperkte CPU capaciteit	- Azure platform is soms moeilijk in gebruik
- Prijs naargelang gebruik (gratis bij weinig gebruik)	- Monitoring is beperkt en moeilijk om duidelijk weer te geven
- Eenvoudig schaalbaar	
- Werkt eenvoudig samen met andere Microsoft toepassingen	

Azure is eerder gemaakt voor bedrijven die al volledig werken met Microsoft en Windows en deze toepassingen willen integreren naar de cloud. Het kan een mogelijkheid zijn om Azure te gebruiken binnen ons project.

Amazon Web services (AWS)

Amazon Web services is een van de grootste spelers op de markt. Het is bekend omwille van zijn goede security en stabiliteit. Het heeft zoals de andere aanbieders alle faciliteiten in huis om onze web app correct te doen werken. Binnen het project zullen we vooral met application integration aan de slag gaan.

Ook voor IoT zijn er vele extensies beschikbaar dewelke we kunnen gebruiken om te connecteren naar onze sensoren. Denk hierbij aan AWS IoT core of AWS IoT analytics.

Voordelen	Nadelen
- Prijs naargelang gebruik	- Prijzen soms onduidelijk (Bill Shock)
- Zeer gemakkelijk in gebruik	
- Goede support door het groot aantal gebruikers	
- Zelden uitval van bereikbaarheid (grote uptime)	
- Vele IoT extensies	

Dit is een van de betere opties om toe te passen binnen ons project. Het team heeft er globaal weinig ervaring mee maar dit mag geen beïnvloedbare factor zijn omdat het zeer gebruiksvriendelijk is.

Conclusie

Uit dit onderzoek komt naar voren dat zowel Azure van Microsoft als zowel Amazon web services beide geschikt zijn voor ons project.

3.2. Software

Programmeertalen

Om de oplossing te kunnen ontwikkelen moet er natuurlijk een keuze gemaakt worden voor de programmeertaal die we willen gebruiken in de back-end van onze applicatie.

Java

Voor de back-end van onze applicatie zou Java de juiste keuze zijn. In combinatie met de onderstaande frameworks is het een zeer krachtige en dynamische oplossing dat gepast is bij onze applicatie. Het geeft ons ook de mogelijkheid om objectgeoriënteerd te werk te gaan aan onze applicatie.

Python

Voor het creëren van AI applicaties voor het monitoren van gezichten bijvoorbeeld zal Python de beste programmeertaal zijn. Er is een groot arsenaal aan beschikbare libraries die ons leven makkelijker zal maken. Zoals: Keras en Tensorflow.

Frameworks

Om ons het leven makkelijker te maken kunnen we gebruik maken van enkele frameworks om de integratie van de applicatie makkelijker te maken.

Spring

Spring zal gebruikt worden als Java framework om de website te creëren aan de hand van het Model View Controller principe. Met dit Framework kan je verschillende routes creëren door API’s te maken die bereikt kunnen met Post, Get, Delete en Update methodes.

Inbreng vanuit Elision

Omdat Elision en zijn dochterbedrijven vooral gespecialiseerd zijn in Python en Java is de keuze voor deze twee talen natuurlijk snel gemaakt. Het maakt het voor ons, maar ook zeker voor Elision makkelijker om ons eventueel te helpen als we vragen hebben of vast zitten met enkele problemen.

Front-end

In dit onderdeel onderzoeken we welke talen en tools we gaan gebruiken voor het ontwerpen van de front-end van de applicatie.

Programmeertaal

Voor de front-end zullen we HTML, CSS, Javascript en eventueel nog JQuery gebruiken.

Frameworks

Criteria	Angular	React
Gemaakt door	Google	Facebook
Programeertaal	TypeScript	JavaScript
Data binding	Two-way binding	One-way binding
DOM	Real DOM	Virtual DOM
Compatibility	Updates nodig	Volledig backward compatible

Angular

Angular hanteert het MVC-model (Model, View, Controller), wat als een soort brug functioneert tussen de front-end en back-end. Angular wordt ondersteund door verschillende platformen (web, mobiel en desktop), wat het perfect passend maakt bij dit project voor het maken van een web-based application.
Door het gebruik van het MVC-model zorgt het ervoor dat de code gestructureerd blijft en het makkelijk is om de code te begrijpen. Angular wordt onder andere gebruikt door: Google, Nike, Forbes en Paypal.

React

React is een Javascript framework wat vooral gebruikt wordt voor maken van UI componenten. Tegenwoordig worden websites vooral volgens het MVC-model geschreven. React wordt vaak omschreven als de “view” van het MVC-model. Enkele bekende React voorbeelden: Facebook, Instagram, WhatsApp en Dropbox.

Conclusie

Omdat Angular het MVC-model hanteert en React alleen het view gedeelte hanteert, is de keuze voor Angular snel gemaakt. Ook hebben we al ervaring met Angular vanuit de Hogeschool, wat het werken met Angular makkelijker maakt voor ons. Verder is de combinatie van Angular en Spring sterk en laat het ons toe om een mooi geheel te creëren.

3.3. IoT

Doorheen het systeem gaan de IoT sensoren een ondersteunende rol spelen. Voornamelijk de hoeveelheid mensen in kaart brengen en hun locatie weergeven is hierbij belangrijk.

Microcontroller

De microcontroller zal voornamelijk dienen als tussenpersoon voor de sensoren en het cloudplatform. Omdat we onze hosting en verwerking in de cloud kunnen doen moet de controller niet zo performant zijn. Voornamelijk uptime en ease of use worden hierbij belangrijk.

We gaan in totaal 5 modules nodig hebben: 3 met camera's voor de hotspots (Lounge, Keuken en Ontspanningsruimte) en 2 zonder camera voor de ingang met RFID en de sensoren om aantal personen die binnen zijn te tellen.

We bekijken in dit deel twee microcontrollers in meer detail:
Enerzijds hebben we de Raspberry Pi, en anderzijds de ESP 32.

Raspberry Pi

De Raspberry Pi is intussen al zeer bekend development board, dat capabel is om een volledige linux desktop omgeving te draaien. Omdat het een volledige desktop omgeving beschikt zijn we vrij om onze development omgeving en programmeertaal te kiezen.
We beschikken op dit moment al over enkele RPi Model 3B+, die een quad-core CPU aan 1.4GHz heeft en 1GB Ram. Een nieuwe Raspberry Pi kost ongeveer 40 euro.

Een bijkomend detail: indien we de Raspberry Pi zouden nemen voor de camera's, zouden we hiervoor ook nog specifieke camera modules voor moeten kopen. Deze kunnen al snel even duur zijn als een standalone ESP 32-CAM.

ESP32

De ESP 32 is ontworpen als een Low-Cost alternatief voor de Arduino dat ook over wifi en bluetooth beschikt. Wegens de lage kostprijs beschikt het bordje over slechts dualcore processor @ 240 mHz en 520 kilobit RAM. Dit verhindert de optie om complexe berekeningen te doen op de module.
Omdat de ESP 32 gebaseerd is op de arduino, moeten we de code ook schrijven in C++ op de Arduino IDE. Daarnaast moeten we de code ook flashen op de microcontroller, wat Over-The-Air updates zeer moeilijk maakt.
Het grootste voordeel van de ESP 32 is de lage kostprijs: Een nieuwe module kost minder dan €10.
Daarnaast is ook de kleine footprint een voordeel van de ESP32. De raspberry Pi heeft een oppervlakte dat 3x zo groot is, en zal ook meer stroom verbruiken.

ESP-32 Cam

Deze aangepaste ESP 32 module heeft een ingebouwde camera. Dit staat toe om een simpele webserver te draaien op de module om zo live beelden te streamen. De bandbreedte is wel gelimiteerd, waardoor je een heel slechte bitrate krijgt op hogere resoluties. De ideale balans tussen bitrate en resolutie vind je bij de VGA resolutie van 640x480. Dit is niet zo hoog, maar voor ons doeleinde is dit genoeg. Daarnaast wordt de USB-Header achterwege gelaten op dit bordje, waardoor je een andere microcontroller moet gebruiken om de initiële setup te doen. Als oplossing hiervoor kunnen we onze eigen USB headers er op solderen, wat wel wat finesse vereist, of met de seriële communicatie pinnen van een ander bordje te werken.Deze module kost ~ €13.

Sensoren

In het systeem zullen enkele sensoren verwerkt zitten die ons gaan toestaan de “people flow” in kaart te brengen.

Personen tellen

Door 2 afstandssensoren te plaatsen aan de in- en uitgang, kunnen we zien wanneer er mensen binnen en buitengaan. Zo kunnen we op een betrouwbare manier weergeven hoeveel mensen er op een gegeven moment in het kantoor zijn en kunnen we preventief tegen mensen zeggen of er nog plaats is in het kantoor.

In een productieomgeving kan gekozen worden om een radar “beam-break” systeem te implementeren. Deze zijn het meest betrouwbaar, maar deze lopen zeer snel op in kost. Daarom bekijken we enkele goedkopere alternatieven die ook gemakkelijker integreren met een Raspberry Pi of ESP32.

Een eerste optie is om een Passieve Infrarood Sensor te gebruiken. Deze worden vaak bij automatische verlichting en alarmsystemen gebruikt, en zijn zeer goed in het detecteren van beweging. Het grootste nadeel hiervan is dat we niet kunnen zien of een persoon naar binnen of buiten gaat. Door de brede “kijkhoek” van de sensor is het ook niet haalbaar om 2 sensoren te plaatsen en zo de juiste richting te bepalen.

Een goed alternatief hierop is een Laser receiver. Deze sensor gaat een signaal sturen wanneer de laserstraal onderbroken wordt. Door 2 van deze sensoren naast elkaar te plaatsen kunnen we zo kijken of er iemand binnen of buiten gaat. Deze modules zijn niet 100% betrouwbaar, maar de lage kost maakt hen wel geschikt voor ons Proof of Concept, ter vervanging van de duurdere radar-emitter.

Enkele aandachtspunten met deze module zijn:

- Dubbele tellingen wanneer mensen hun ledematen apart van het lichaam de laser onderbreken.
- Meerdere mensen die samen de laser maar 1 keer onderbreken.

Een laatste optie is om een “PaxCounter” te gebruiken. Dit is een soort van antenne die de wifi signalen van apparaten gaat onderscheppen, hun MAC-adres aanvraagt en zo kan tellen hoeveel Smartphones er in de buurt zijn. Door de gelimiteerde range en het feit dat verdiepen onder/boven ook geteld gaan worden raden we deze module niet aan.

Camera’s

Om “Hotspots” vanop afstand te kunnen bekijken willen we enkele camera’s plaatsen. Zo kunnen werknemers kijken of er bijvoorbeeld plaats is aan de koffiemachine.
De hotspots die we zeker in de gaten willen houden zijn:

- De Lounge
- De ontspanningsruimte
- De keuken

Voor de camera’s raden we aan om een ESP 32-CAM te gebruiken. Deze goedkope microcontroller wordt geleverd met een cameramodule en kan zijn eigen webserver hosten met een livestream van de beelden. De bandbreedte is wel gelimiteerd wegens het gebrek aan processing power (om een goede bitrate te halen kunnen we niet veel hoger dan 720p) , maar voor onze doeleinden is dit goed genoeg. Omdat we rechtstreeks naar de

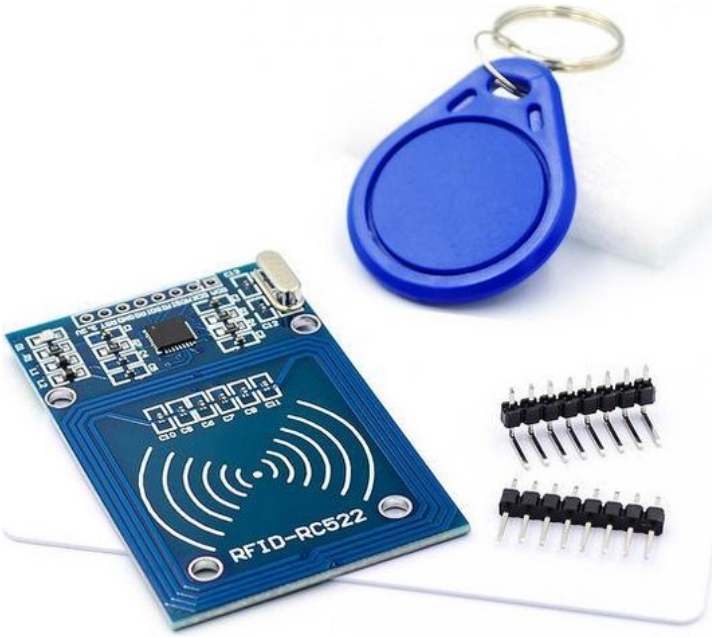
eindgebruiker livestreamen en niets opslaan, vergemakkelijkt dit ook onze privacyregelementering. Een nadeel hiervan is wel dat we de camera's niet kunnen gebruiken als beveiligingsmaatregel.
Een alternatief is om een cameramodule aan een Raspberry Pi te hangen. Dit zal vergelijkbare resultaten opleveren, alleen zal de RPi hogere resoluties en meer gebruikers aankunnen.
Als laatste alternatief hebben we vernomen dat er ook enkele AWS Deep Lens camera's beschikbaar zijn. Deze camera's zijn ontworpen voor machine learning en zullen meer dan geschikt zijn om simpele personenherkenning toe te passen. Indien we met een ander cloudplatform dan AWS gaan werken, gaat deze camera wel geen optie zijn, omdat de meeste functies van de camera gelinkt zijn aan AWS applicaties.

RFID

Om 100% zeker te zijn hoeveel mensen er in het kantoor zijn, kunnen we ook aan Acces Control doen met RFID Chips. Hierbij worden alle werknemers badges gegeven die ze moeten scannen om binnen te gaan. Zo kunnen we makkelijk bijhouden wie er op het kantoor is. Dit staat ons ook toe om gereserveerde stoelen toe te kennen (of vrij te zetten indien iemand niet komt opdagen en vergeet zijn reservering te annuleren).

Omdat we hierbij personen moeten linken aan het ID in de badge, moeten we hier wel goed letten op vlak van privacy. Prioritair hieraan is een goede beveiliging van de persoonlijke data die gelinkt wordt aan de badge.

De module die we voorstellen om te gebruiken is de **RC522**. Deze module is zeer goedkoop en staat ons toe om zowel RFID chips te lezen en hun ID te verwerken.



Communicatie van de IoT componenten

Bewegingssensoren

Om te tellen hoeveel mensen er in het kantoor zijn maken we gebruik van 2 bewegingssensoren. Deze sensoren hangen we simpelweg aan de digitale GPIO pinnen van de microcontrollers. Wanneer de er beweging gedetecteerd wordt, gaat een digitale pin van status veranderen.

Camera

De camera modules kunnen we rechtstreeks verbinden met onze microcontroller. Zowel de Raspberry Pi als de ESP 32-CAM hebben een connector om de camera met een FFC kabel te verbinden. Het beeld van de camera wordt rechtstreeks gehost op de microcontroller en de applicatie laadt deze URL in.

RFID

De RFID module kan met zijn microcontroller communiceren door gebruik te maken van SPI. SPI werkt in een Master-Slave relatie door middel van 4 kabels: Clock, Mosi (Master out, slave in), Miso (Master in, slave out) en SS (Slave Select). Het voordeel hiervan is dat je op één microcontroller meerdere modules kan aansluiten.

Microcontrollers

Onze microcontrollers zullen de data die ze inlezen met de sensoren doorsturen naar de applicatie op het cloudplatform. Deze communicatie gaat gebeuren via MQTT. De microcontrollers verbinden met het internet en subscriben aan een hub die gehost wordt op het cloudplatform. Indien een sensor data wilt doorsturen, published het deze data naar de cloud-applicatie. Een webapplicatie kan dan luisteren naar het channel (door zelf te subscriben) en dan de data verwerken terwijl ze binnenkomt.

Conclusie

In de onderstaande tabel vindt u onze aanbevelingen voor de vereiste componenten.

We raden aan om 2 raspberry pi's te gebruiken om de personen aan de inkom te tellen aangezien we hierover reeds beschikken. Voor de camerasystemen aan de 'Hotspots' bevelen we 3 ESP 32-CAM's aan.
De tellingen doen we met 2 lasersystemen, bestaande uit zowel een emitter als een receiver. Als extra kunnen we aan access control doen met de RFID Modules, maar omdat deze heel wat extra werk vereisen voor niet zo veel payoff willen we als nice to have vermelden.

Modelnaam	**Aantal**	**Prijs**	**Link**
Raspberry Pi	2	/	(De studenten beschikken hier zelf al over)
ESP32-CAM	3	€11/stuk	https://www.tinytronics.nl/shop/en/communication/bluetooth/esp32-cam-wifi-and-bluetooth-board-with-ov2640-camera

Laser Emitter Module	2	€1,50/stuk	https://www.tinytronics.nl/shop/en/lighting/laser/red-laser-module-5v-650nm
Laser Receiver Module	2	€2,00/stuk	https://www.tinytronics.nl/shop/en/sensors/light/red-laser-sensor-module
RC522 RFID Module	1	€4,50	https://www.tinytronics.nl/shop/en/communication/rfid/rfid-kit-mfrc522-s50-mifare-with-card-and-key-tag
S50 RFID Key Tag	3	€1/stuk	https://www.tinytronics.nl/shop/en/communication/rfid/s50-mifare-key-tag

3.4. Dashboards

Op het dashboard willen we de informatie met betrekking tot de essentiële bedrijfsprestaties voor belangrijke klanten weergeven. In dit geval gaat het voornamelijk over het aantal werknemers die zich bevinden in de Corda Campus.

Tools

In de wereld van Business Intelligence zijn er veel verschillende tools. De hoofdspelers op deze markt zijn Power BI, Qlik Sense en Tableau.

Tijdens onze opleiding zijn we al in contact gekomen met enkele van deze tools. Hierdoor hebben we een goed beeld van de voor- en nadelen. Power BI is de tool van Microsoft die handig is maar op zich niet veel mogelijkheden heeft om het datamodel aan te passen of dimensioneel te maken. Qlik sense heeft deze mogelijkheid wel. Door het gebruik van de script editor kan je het model aanpassen op maat van de visualisaties die je wil gebruiken voor de opdracht. Ook is Qlik de meer gebruiksvriendelijke tool van de drie. Verder is het mogelijk om deze tool te integreren in een website door gebruik te maken van de API van Qlik Sense. Tableau is voor deze opdracht echter overkill omdat de visualisaties die we willen maken niet zeer geavanceerd zijn.

Rekening houdend met de vorige argumenten denken we dat de beste tool van deze opdracht Qlik Sense is.

Visualisaties

Een voorbeeld van de visualisaties die we zouden kunnen maken zijn het aantal reserveringen per dag. Een bar chart zou goed kunnen werken voor deze visualisatie. Ook kan je nog kleuren gebruiken om een limiet of een gevaarlijk hoog aantal van reservaties aan te duiden. Deze visualisaties kunnen in Qlik Sense gemaakt worden in dashboards. Ook zou het verder leuk zijn

als we het dashboard direct kunnen integreren op onze tool zodat de office manager aan deze belangrijke visualisaties kan.

Een andere meer complexe visualisatie zou het gebruiken van een map zijn en daarop aanduiden waar mensen zitten en hoeveel mensen zich bevinden in de keuken bijvoorbeeld. Maar dit is een visualisatie die een nice to have zal zijn omdat het vooral een custom oplossing zal zijn voor die maat van de opdracht zelf gemaakt moet worden.

De belangrijkste KPI's (Key Performance Indicators) in deze context zijn de aantal reserveringen, en de vergelijking ervan tegenover de maximum bezetting. Ook de verdere onderverdeling voor de keuking bijvoorbeeld is belangrijk. Met deze KPI's kan je besluiten als office manager dat het een gevaarlijke situatie is of niet.

Conclusie

Het lijkt ons de beste keuze om de dashboarden te maken in Qlik Sense. De visualisaties voor deze opdracht zullen eerder beperkt zijn, maar vormen nog altijd een essentieel deel van de applicatie.. Het gaat hier vooral over de onderverdeling zodat je eventuele gevaren kan opsporen en actie kan ondernemen tegen deze gevaren. Bijvoorbeeld als er te veel mensen tijdens de middag in de keuken zijn. Door hier actie tegen te nemen gaan werknemers zich ook veiliger voelen en dus ook meer geneigd zijn om te komen werken.

4. User Experience

4.1. User

User Experience (UX) is het gevoel dat een gebruiker krijgt bij het vervullen van een taak. Het is de bedoeling dat de gebruiker dit zo effectief, efficiënt en prettig mogelijk kan doen. Hierdoor zal de gebruiker tevreden zijn en de applicatie blijven gebruiken.

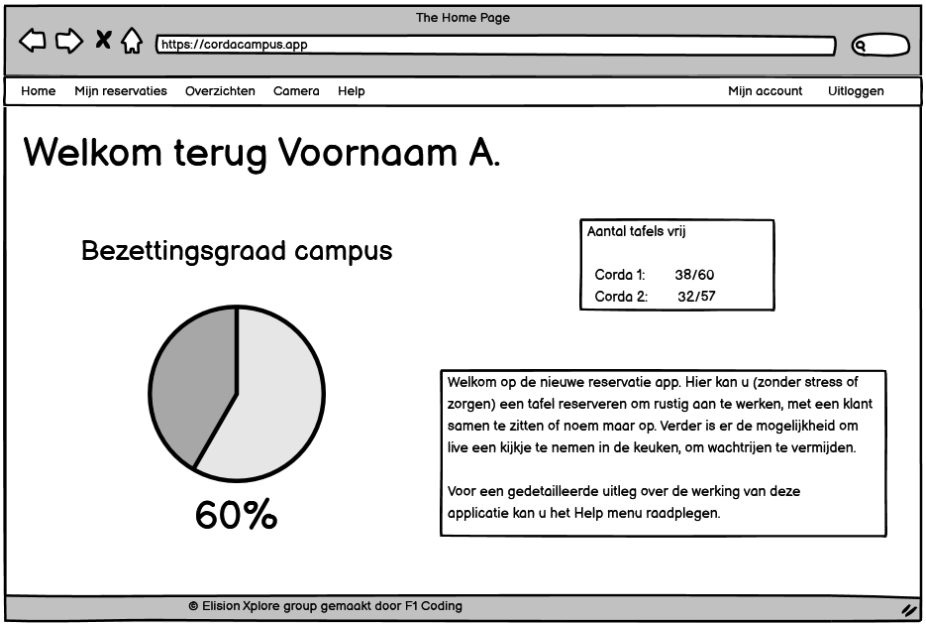
Hoe zorgen we voor een goede User Experience?

We weten wat de frustraties zijn bij de huidige reservatie applicatie. Het focus ligt op het feit dat het reserveren op een makkelijke en vlotte manier moet kunnen gebeuren.

De applicatie moet geopend kunnen worden op zowel een computer als een smartphone. Hierbij gaan we er vooral op letten dat het er op een smartphone gebruiksvriendelijk uitziet, want de smartphone zal nota bene het vaakst gebruikt worden om reservaties te boeken. Ook gaan we er op letten dat er niet teveel informatie op een pagina staat en dat de pagina's er overzichtelijk uitzien.

Voorbeeld

Hieronder kunt u een voorbeeld zien van hoe de homepagina eruit zou kunnen zien. De gebruiker kan in één oogopslag de bezettingsgraad zien en hoeveel tafels er nog vrij zijn. Hierdoor weet de gebruiker meteen hoe druk het is en of een reservatie voor op dat moment mogelijk is.



4.2. Admin

Monitoring kan bezien worden als het bewaken van een applicatie. Dit houdt voornamelijk in dat de administrator kan toekijken dat alles volgens de planning loopt en er zich geen onvoorziene problemen opduiken.

Hoe kunnen we zorgen voor Monitoring?

We kunnen voor monitoring een aparte pagina maken, op deze pagina staat veel informatie die ervoor zorgt dat de applicatie “bewaakt” kan worden.

Monitoring kunnen we voorzien door het maken van een changelog, concreet voor ons project wil dit zeggen wanneer er een persoon een tafel claimed er automatisch een regel tekst met daarin de informatie van de reservatie (geen persoonsgegevens) en een timestamp verschijnt, zo kan men wanneer er iets fout loopt ook gaan zoeken waar er iets is fout gelopen om de fout op te lossen.

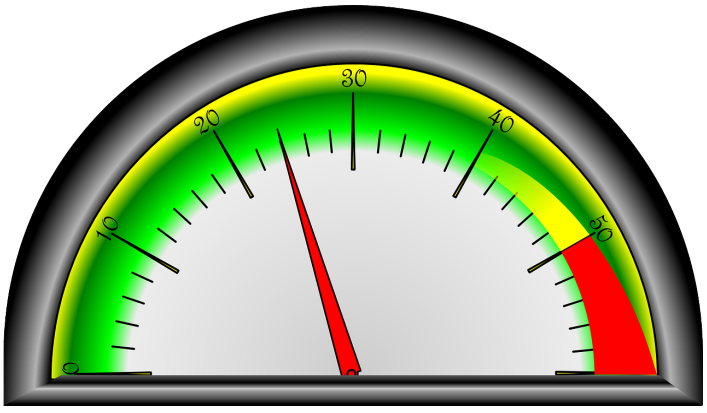
Wat is Analyseren?

Analyseren is het onder de loep nemen van hoe iets werkt en daaruit conclusies trekken of iets goed/slecht werkt.

Hoe kunnen we analyseren mogelijk maken?

Het analyseren van gegevens kan iets zeer ingewikkeld worden wanneer je veel data hebt, als dit het geval is dan gebruiken we het best grafieken om deze gegevens voor te stellen.

Het analyseren van deze data kan gedaan worden door een aparte pagina aan te maken die alleen toegankelijk is aan de administrator/floormanager, op deze pagina zetten we enkele grafieken en kerncijfers.
Een voorbeeld van een grafiek is een meter waarbij de maximum bezettingsgraad staat en die vergelijkt met de huidige bezettingsgraad.
Een voorbeeld van een kerncijfer is dan het cijfer van het totaal aantal reservaties van een bepaalde dag,week,maand,… .



Conclusie

Onze applicatie zal bestaan uit twee onderdelen: Een user-side en een admin-side.
In de user side zal men afspraken kunnen maken en de huidige bezetting kunnen bekijken. Aan de admin-side kan men ook historische data raadplegen alsook aan user administratie kunnen doen.
Om voor een goede UX te zorgen plegen we regelmatig tests met externe gebruikers, om zeker te zijn dat de applicatie gebruiken een prettige ervaring is.

5. Toekomstperspectief

5.1. Adaptability

Aanpassingsvermogen of adaptiviteit is het vermogen van organismen en samenlevingen om te veranderen. Dit heeft betrekking op de mate waarin zij kunnen reageren op veranderde omstandigheden van buitenaf die de wisselwerking beïnvloeden, ofwel het zich kunnen instellen op gewijzigde eisen en omstandigheden in het milieu.

Wat is de toepassing van adaptability voor onze applicatie?

De adaptability van iets wil aangeven hoe aanpasbaar de applicatie is wanneer de omgeving gewijzigd wordt. Specifiek wil dit zeggen dat wanneer bijvoorbeeld Elision van Corda Campus 1 verhuist naar Corda Campus X dat er geen problemen zouden mogen zijn om de applicatie te kunnen aanpassen aan de gewijzigde omstandigheden (bv. nieuw grondplan/andere hoeveelheid tafels).

Hoe gaan we zorgen voor adaptability?

Concreet kunnen we dit opdelen in 2 Stappen:

Stap 1: Objectgeoriënteerd programmeren
Stap 2: Zorgen voor scalability

Objectgeoriënteerd programmeren zal ervoor zorgen dat we kunnen werken met objecten, deze objecten en talloze variaties hiervan zullen de aanpasbaarheid van onze applicatie verbeteren.

De scalability van ons project zal ervoor zorgen dat onze applicatie kan uitgebreid worden indien er uitbreidingen moeten komen.

Post-corona

Deze applicatie is een reservatiesysteem, de applicatie zal dus ook na de huidige corona omstandigheden kunnen worden gebruikt voor het reserveren van tafels/ruimtes voor bijvoorbeeld het afspreken met een klant.

5.2. Scalability

Het is van belang dat onze webapplicatie op elk moment bereikbaar is voor iedereen. We moeten er dus voor zorgen dat onze systemen overeenstemmen met het aantal te verwachten gebruikers.

Gebruikers

Er werken redelijk veel mensen voor de cronos groep. Echter zullen deze mensen nooit altijd gelijk reservaties bekijken etc. We kunnen er vanuit gaan dat er in het begin wanneer enkel hasselt met de applicatie werkt, er ongeveer 30-35 mensen tegelijk op de piekperiode zullen werken met de applicatie.

Het aantal gebruikers zal altijd binnen de limieten van het aantal personeel van de cronos groep blijven aangezien deze enkel intern wordt gebruikt. Er moet dus geen capaciteit worden voorzien voor duizenden mensen en schaalbaarheid is niet echt zeer belangrijk voor dit project.

Data

Onze webapplicatie genereert enkel maar kleine data en niet bijvoorbeeld beeld en video’s. Dit zorgt ervoor dat we onze server capaciteit ook beperkt kunnen houden. Echter moet dit wel goed opgevolgd worden. En bij extreme data pieken moet er snel kunnen worden ingegrepen.

Server Capaciteit

Omdat we via een cloud platform werken is capaciteit geen probleem. We kunnen eenvoudig capaciteit bij voorzien of werken met automatische capaciteit. Onze web app zou dus in normale omstandigheden altijd bereikbaar moeten zijn.

Conclusie

Schaalbaarheid is niet zeer belangrijk binnen ons project aangezien we een vast aantal medewerkers hebben en de data flow ook redelijk constant is. Wanneer er toch een piek zou ontstaan kan er automatisch extra servercapaciteit worden bij voorzien.

5.3. Presets/Scenario's

Om de applicatie gebruiksvriendelijk te maken willen we graag gebruik maken van bepaalde 'presets'. Dit zorgt ervoor dat de administrators minder werk hebben bij het instellen van de web applicatie. Aangezien niet alles voorspelbaar is komt er ook een mogelijkheid om een eigen configuratie te maken en op te slaan.

Scenario 1: Groen

In dit scenario zijn alle stoelen en werkplekken beschikbaar. Het systeem van reservatie is in deze situatie niet verplicht maar kan wel worden gebruikt om bijvoorbeeld een ruimte af te huren om een gesprek te hebben met een klant. Men kan dus alle stoelen selecteren. Wel wordt in dit scenario drukte gemonitord in de keuken zodat medewerkers direct kunnen zien hoe druk het is en hun lunch misschien even willen uitstellen.

Scenario 2: Oranje

In dit scenario zijn de helft van de normale stoelen beschikbaar. Medewerkers wordt aangeraden om thuis te werken maar naar kantoor komen kan nog steeds. Klanten ontvangen is nog steeds mogelijk. Er wordt verplicht gewerkt met een reservatiesysteem. Dit scenario kan worden gebruikt wanneer onderlinge afstand noodzakelijk is. Lunch wordt bij voorkeur genuttigd aan de werkplek.

Scenario 3: Rood

Thuis werken wordt zoveel mogelijk verplicht. Klanten of sollicitanten ontvangen is nog steeds mogelijk maar enkel na reservatie. De capaciteit wordt teruggebracht naar max 1/5. Lunch wordt aan de werkplek genuttigd. Admins moeten reservaties opvolgen en waar nodig mensen weigeren.

Scenario 4: Zwart

Thuiswerk is verplicht in dit scenario. De applicatie zal dit duidelijk vermelden en zal geen reservaties aannemen. Een eigen boodschap kan worden ingesteld waarom de app niet in gebruik is. Dit scenario kan ook worden gebruikt wanneer men geen gebruik meer wilt maken van de app of als deze een storing heeft.

Scenario 5: Eigen keuze

Administrators geven een aantal te reserveren stoelen in voor een bepaalde dag. Met deze flexibiliteit kunnen ze bijvoorbeeld wanneer er werkzaamheden zijn, een verdieping buiten dienst stellen. Dit scenario moet zelf worden ingesteld dus er moet voor worden gezorgd dat dit ook gebruiksvriendelijk is. Men kan het aantal plaatsen, maximum aantal klanten en regels/boodschappen zelf instellen.

Conclusie

Toekomstperspectief creëren we door het invoeren van aanpasbare scenario's, het zorgen voor adaptability en scalability. Deze zaken zullen ervoor zorgen dat onze applicatie in de toekomst ook nuttig zal kunnen blijven.

6. Besluit

Voor de hosting van ons reservatiesysteem komen zowel Amazon web services als Azure van Microsoft in aanmerking. De taal waarin we schrijven zal voor de back-end Java (Spring framework) zijn. IoT en AI wordt voornamelijk geschreven in Python. Twee bekende libraries die ons leven makkelijker kunnen maken zijn Keras en Tensorflow. We raden aan om 2 raspberry pi's te gebruiken en 3 ESP32-CAM's. Voor de front-end hebben de APP-ers binnen ons team een voorkeur voor Angular. Het dashboard waar men bijvoorbeeld het aantal reservaties per dag kan zien stellen we op met Qlik Sense.

Binnen onze applicatie zijn er 2 rollen: User voor een werknemer en Admin voor een office/floor manager. Voor een werknemer moet reserveren en bekijken van reservaties zo vlot mogelijk verlopen. Hierbij gaan we ook opletten dat het er op een smartphone ook gebruiksvriendelijk uitziet. Voor de Admin is monitoring vooral van belang en zal hij/zij historische data kunnen raadplegen en de logging van de applicatie bijvoorbeeld een regel tekst wanneer een tafel wordt gereserveerd.

Tenslotte als toekomstperspectief willen we een succesvolle applicatie af leveren ook op lange termijn door adaptability, scalability en presets/scenario's. Adaptability realiseren we door objectgeorienteerd te programmeren en onze applicatie scalable te maken. De scalability hangt af van het aantal gebruikers en data. Beide zien we niet als een struikelblok voor in de toekomst aangezien we een vast aantal medewerkers verwachten en kleine data. Als laatste pijler hebben we aanpasbare scenario's, deze zullen ervoor zorgen dat onze applicatie in de toekomst ook nuttig zal kunnen blijven.

7. Bijlages

7.1. GDPR

De Algemene Verordening Gegevensbescherming (AVG) (Engels: General Data Protection Regulation, GDPR) is een Europese verordening die de regels voor de verwerking van persoonsgegevens door particuliere bedrijven en overheidsinstanties in de hele Europese Unie standaardiseert. Het doel is niet alleen om de bescherming van persoonsgegevens binnen de Europese Unie te garanderen, maar ook om het vrije verkeer van gegevens binnen de Europese interne markt te waarborgen. De verordening geldt wereldwijd voor alle ondernemingen en organisaties die persoonsgegevens bijhouden en verwerken van natuurlijke personen in de Europese Unie, onafhankelijk of er al dan niet betaald wordt voor diensten of producten.

AVG betreft privacy en persoonsgegevens. Het gaat hier over het beschermen van natuurlijke personen bij de verwerking van persoonsgegevens. In andere woorden: in alle gevallen waar we identificeerbare gegevens van mensen verzamelen, verwerken, opslaan, bewaren, etc.

Grondslag

Het begint natuurlijk met de vraag waarom en welke persoonsgegevens nodig zijn om te verzamelen, bewerken en opslaan, de grondslag. Dit is vaak niet een beslissing die in een ontwikkelteam gemaakt wordt, maar is er juist een die vanuit de organisatie al is vastgesteld. Het gaat hier namelijk vaak om contractuele verplichtingen, wettelijke verplichtingen of zogeheten vitale belangen. Als klein team of zelfstandige ontwikkelaar kom je hier nog wel eens mee in aanraking. Je klant heeft een goed idee voor een stuk software dat gegevens verzamelt van haar klanten. In die gevallen kun je dus zeker voor elk van de verzamelde gegevens de nodige vragen stellen:

- Hebben we toestemming van de gebruiker nodig?
- Wat is het belang voor de organisatie (je klant)?
- Is er een wettelijke verplichting?
- Is er een contractuele verplichting (tussen de klant en haar klanten bijvoorbeeld)?

Toestemming gebruiker

De gebruiker van de applicatie moet toestemming geven. Als software ontwikkelaar betekent dit, dat je nu ook de expliciete toestemming moet verkrijgen van de gebruikers. Bijvoorbeeld door het toepassen van “Privacy by design” of “Privacy by default”, waarbij gebruikers expliciet toestemming moeten geven door een vinkje te zetten (en dat dit dus niet standaard aangevinkt mag staan).
Belangrijk hierbij: geldigheid van de toestemming, maar ook het scenario, dat gebruikers deze toestemming moeten kunnen intrekken.

Privacy Impact Assessment

Organisaties zijn verplicht om op periodieke basis interne systemen en processen te controleren op privacy risico's en effecten. Een dergelijk assessment heeft als output een rapport, dat inzicht geeft in de compliance, effecten en risico's van het beheer en verwerken van persoonsgegevens. Dit document wordt vaak door software architecten tijdens het ontwerpen gebruikt, maar is ook voor teams nuttig bij het maken van software.

Voorwaarden DPIA (data protection impact assessment):

- Een systematische beschrijving van de beoogde gegevensverwerkingen en de doeleinden hiervan. Beroept u zich op een gerechtvaardigd belang als grondslag voor de verwerking? Neem dit dan ook op in de beschrijving.
- Een beoordeling van de noodzaak en de proportionaliteit van de verwerkingen. Dat houdt in: is het verwerken van persoonsgegevens op deze manier noodzakelijk op uw doel te bereiken? En is de inbreuk op de privacy van de betrokkene niet onevenredig in verhouding tot dit doel?
- Een beoordeling van de privacyrisico's voor alle betrokkenen.
- De beoogde maatregelen om zowel de risico's aan te pakken (zoals waarborgen en veiligheidsmaatregelen) alsook aan te tonen dat u aan de AVG voldoet.

Bewijslast

AVG is een verordening en in die zin ook gewoon wetgeving. Het is daarom belangrijk dat we bij de ontwikkeling van de software (ontwerp, realisatie, uitlevering, etc.) deze zeker volgen. Door de AVG al van het begin te volgen kunnen we goed verantwoorden waarom we welke data willen verwerken.

Privacy by Design

Privacy is niet iets wat je zomaar achteraf aan een applicatie kunt toevoegen. Zeker bij nieuwe software of functionaliteit is het dus belangrijk hier direct bij het ontwerp al mee te beginnen. Welke aspecten hier een rol spelen hangt natuurlijk helemaal af van het type oplossing, maar in het algemeen zullen de volgende onderdelen een rol gaan spelen:

Ethische vraagstukken spelen een belangrijke rol bij het ontwerpen van software. Gaan we goed om met de gegevens van onze eindgebruikers en nemen we geen onnodige risico's? Stappen we niet te gemakkelijk over risico's heen en programmeren we defensief genoeg? Het gebeurt nogal eens dat teams onder druk hier niet voldoende rekening mee houden of iets over het hoofd zien. Daarom is het belangrijk dat we dit vooraf goed vastleggen.

De manier waarop systemen en gegevens afgeschermd zijn, moet natuurlijk ook voldoende gedocumenteerd zijn. Maar dataveiligheid gaat ook over het niet onnodig opslaan van gegevens. Het zoveel mogelijk weglaten van identificeerbare gegevens is bijvoorbeeld een prima standaard maatregel.

De kwaliteit van de opgeslagen data moet passen bij het beoogde doel. Het gebeurt soms, dat we uitgebreide persoonsgegevens opslaan voor bijvoorbeeld statistische doeleinden. In die gevallen is het vaak beter om de gegevens in kwaliteit te laten afnemen (bijvoorbeeld door te consolideren of te anonimiseren) in plaats van in ruw formaat te bewaren.

Het is belangrijk gegevens te kunnen bewaren zolang ze van belang zijn (zie Grondslag). Maar als gegevens niet meer nodig zijn, mogen deze niet onnodig bewaard worden. Hetzelfde geldt voor niet-relevante gegevens.

Binnen het ontwikkelingsproces

In het software ontwikkelingsproces is het belangrijk dat het team op de hoogte is van de verantwoordelijkheid die hoort bij het werken met persoonsgegevens. Documenten als het Privacy Impact Assessment Report kunnen hier goed bij helpen, maar een basiskennis van de AVG blijft een vereiste.

Daarnaast is het ook “common sense”. Er zijn veel voorbeelden te bedenken:

- We beperken ons tot de noodzakelijke gegevens en slaan niets extra’s op.
- Er komen geen direct tot een persoon herleidbare gegevens in log files terecht.
- Ontwikkelaars hebben geen toegang tot productie gegevens.

Onderhoud van eigen gegevens

De AVG zegt ook iets over het onderhoud van de eigen persoonsgegevens. Dit zijn de bekende 5 rechten: Recht op inzage, Recht op wijzigen, Recht om vergeten te worden, Recht op overdracht en Recht op informatie. In de praktijk betekent dit vaak, dat we hier functionaliteit voor realiseren in onze toepassingen. Denk aan schermen voor het beheren en downloaden van onze persoonlijke gegevens.

Ten behoeve van testen of analyseren van data zal een organisatie verzamelde gegevens bewerken alvorens het hiervoor te gebruiken. Technieken als maskeren, anonimiseren, pseudonimiseren en versleutelen spelen hier een belangrijke rol. Ook hoeft er niet altijd echte

data gebruikt te worden, maar kan er gewerkt worden met gegenereerde of artificiële data. Hoewel dit zich vaak buiten het zicht van de ontwikkelaars afspeelt, komen ze hier wel mee in aanraking. Bijvoorbeeld bij het gebruik van een OTAP omgeving waarbij de OTA omgevingen gemaskeerde data gebruiken.

Bronnen:
Fogg S. (2019) GDPR for Dummies: Simple GDPR Guide for Beginners Geraadpleegd op 16 november 2020, https://termly.io/resources/articles/gdpr-for-dummies/
Burger J. (2018) AVG voor ontwikkelaars, Geraadpleegd op 16 november 2020, https://vxcompany.com/2018/09/24/avg-voor-ontwikkelaars/
Autoriteit Persoonsgegevens, Data Protection Impact Assessment, Geraadpleegd op 16 november 2020, https://autoriteitpersoonsgegevens.nl/nl/zelf-doen/data-protection-impact-assessment-dpia

7.2. Voorlopig datamodel