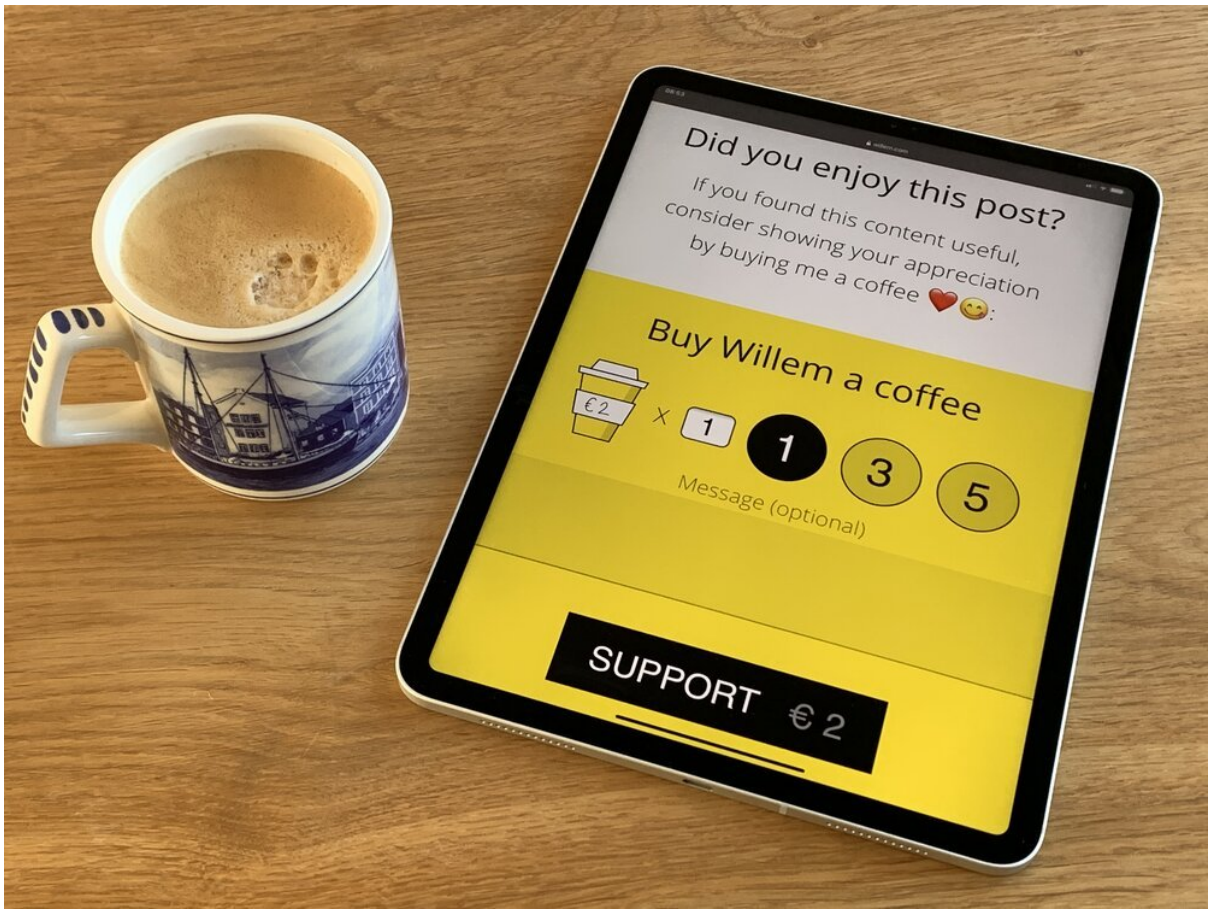
Het ontwerpen en implementeren van een (micro)betaalsysteem

Mijn blog te gelde maken met koffie, Apple Pay en Mollie

Willem L. Middelkoop
25 mrt. 2020



Online betalingen zijn nu belangrijker dan ooit, aangezien bedrijven worden ontwricht door het COVID-19 virus. Het drijft mijn klanten ertoe om nieuwe manieren te zoeken om online geld te verdienen. Ik heb een (micro)betaalsysteem ontworpen en geïmplementeerd. Deze post gaat over het bereiken van eenvoud door complexe uitdagingen op te lossen.

Nieuwe technologie begrijpen

Bij het ontwikkelen van nieuwe technologie en apps probeer ik me altijd voor te stellen hoe het het beste zou werken. De makkelijkste manier om daar achter te komen, is om het product voor jezelf te ontwerpen. Net zoals Steve Jobs Apple hun presentatie-app,

Keynote, voor zichzelf liet ontwikkelen. Het is een van zijn beroemde grappen over hem als onderbetaalde betatester

Steve Jobs kondigt Keynote aan op MacWorld 2003, grappend dat hij het voor zichzelf gebouwd heeft, omdat hij een onderbetaalde bètatester is

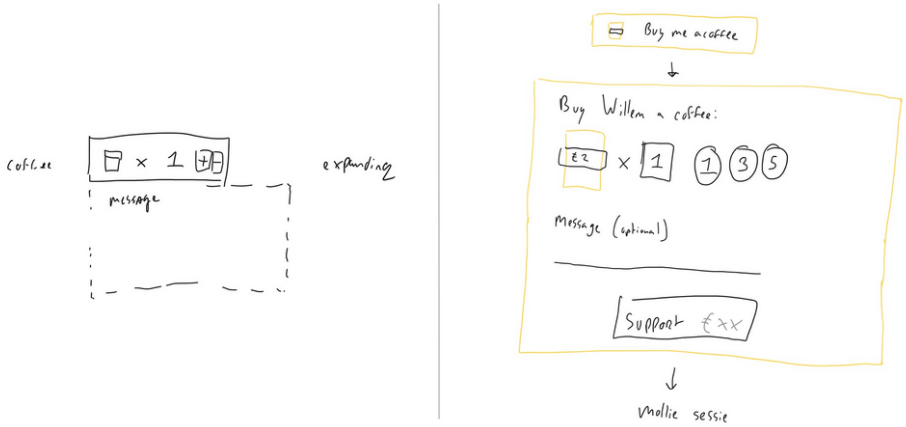
Als je het product voor jezelf ontwerpt, denk dan na over wat je echt, *echt*, zou willen. Voor mij betekent dit het meest eenvoudige gebruikersinteractiemodel mogelijk: ik wil dat mijn software simpel en makkelijk is.

Use case: koop me koffie

Voor het ontwikkelen van de betaaltechnologie heb ik de use case bedacht om een ”koop me koffie”-knop aan mijn website toe te voegen. Als het goed werkt, kunnen mensen het gebruiken om hun waardering voor deze blog te tonen door mij een koffie te kopen. Ik denk dat deze use case een perfecte uitdaging is om een (micro)betaalsysteem te ontwerpen (en te bouwen).

Ideeën schetsen

Voordat ik ook maar één regel code programmeer, ontwerp ik software vaak door ideeën met de hand te schetsen. Ik vind het creatieve proces van tekenen nuttig bij het verkennen van ideeën, lay-outs en processen. Het is een van de redenen waarom ik zo graag op een tablet werk, digitaal ideeën schetsen stelt me in staat om snel meerdere ideeën te verkennen door ze te kopiëren/plakken in meerdere iteraties.



[image] Interface concept schetsen, verschillende opties verkennend om een bericht achter te laten en meerdere kopjes koffie te kopen

Prototyping

Zodra je ideeën voor je gebruikersinterface hebt verkend, is het tijd om een prototype te ontwikkelen. Wanneer je een prototype bouwt, kun je de gebruikersinterface op verschillende apparaten testen, zien hoe het eruit ziet en aanvoelt.

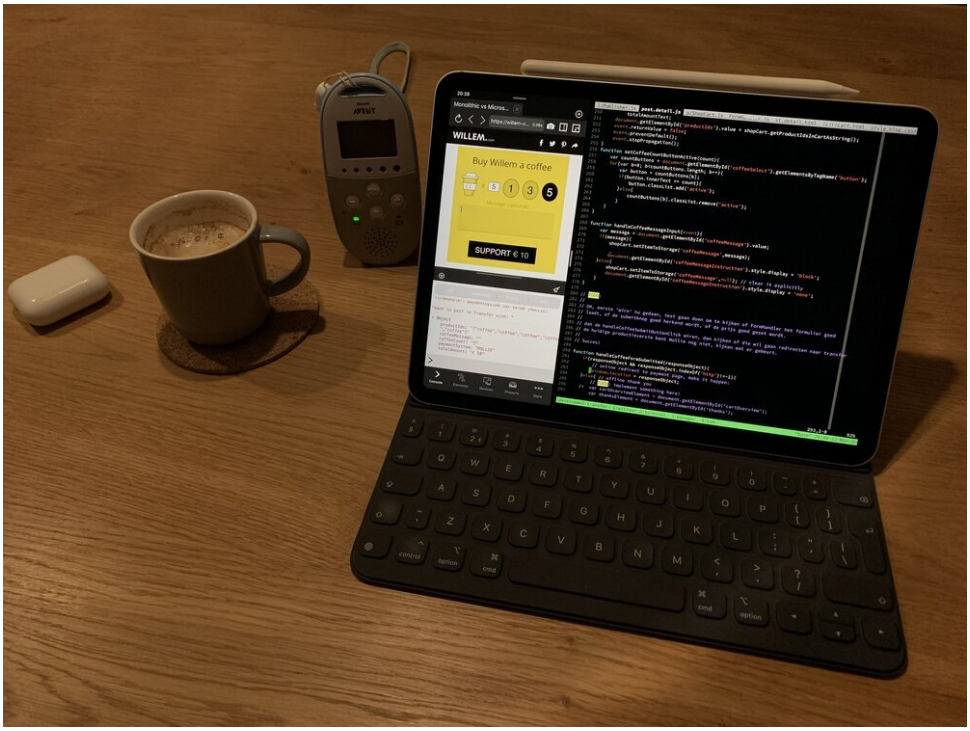
[image] Vertaling van de designschets (links) naar een prototype (rechts) op iPad

Tijdens het prototyping kijk ik vaak naar de ontwerpschets terwijl ik de eerste versie programmeer. Op de iPad kun je dit doen met behulp van split screen, waarbij ik de ontwerpschets direct naast mijn code plaats, met behulp van VIM in tmux via Mosh met de briljante Blink app voor iPadOS. Ik heb het koffiekopje gemaakt met Picta Graphic voor iPad, een prachtige teken-app die exporteert naar vectorafbeeldingen (SVG).

[image] De koffiekop ontwerpen met Picta Graphic voor iPad

Frontend-ontwikkeling

De volgende stap is het ontwikkelen van een volledig werkende gebruikersinterface, waarbij de knoppen werken. Tijdens de ontwikkeling moet je vaak uitzoeken wat er aan de hand is, bugs opsporen en mechanismen finetunen. Ik gebruik de Inspect Browser voor iPadOS om een werkende miniatuur-touchscreeninterface direct naast mijn code te hebben. Het stelt me in staat om de werkende gebruikersinterface snel te testen en te perfectioneren.

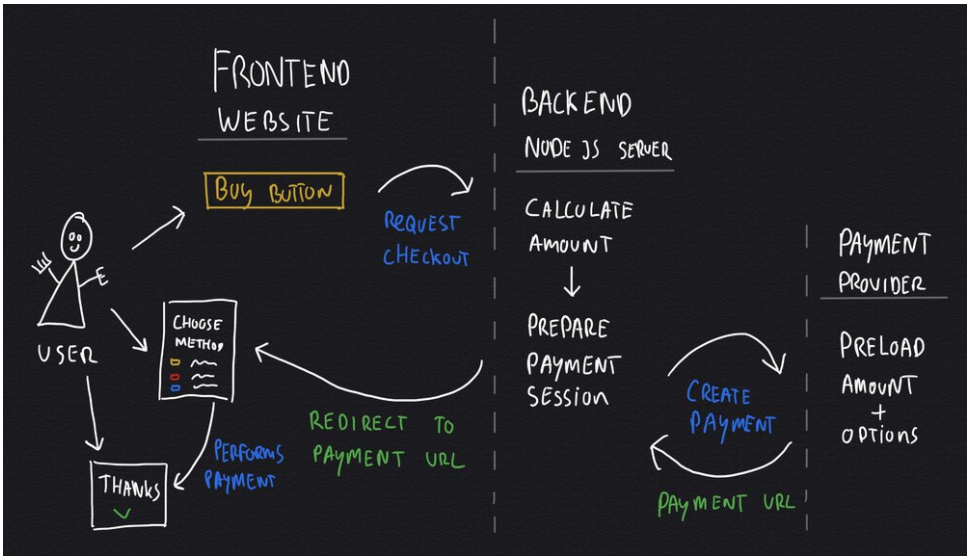


[image] De gebruikersinterface ontwikkelen met Blink en Inspect Browser op iPad

Backend-ontwikkeling

Een andere cruciale stap is het ontwikkelen van de backend-technologie om online betalingen voor te bereiden en te volgen. Dit is server-side technologie die de communicatie tussen de frontend gebruikersinterface (website) en de betalingsaanbieder (banken, creditcards, Apple Pay, etc.) afhandelt.

Het ontwerpen van de backend-server is meestal iets abstracter in vergelijking met het ontwerpen van een (zichtbare) frontend gebruikersinterface. Ik vind het schetsen van UML-sequentiediagrammen erg handig bij het uitzoeken welke API-aanroepen in welke volgorde moeten plaatsvinden. Sequentiediagrammen visualiseren programmainteracties in tijdsequentie.



[image] Geschetste sequentiediagram van het betaalproces

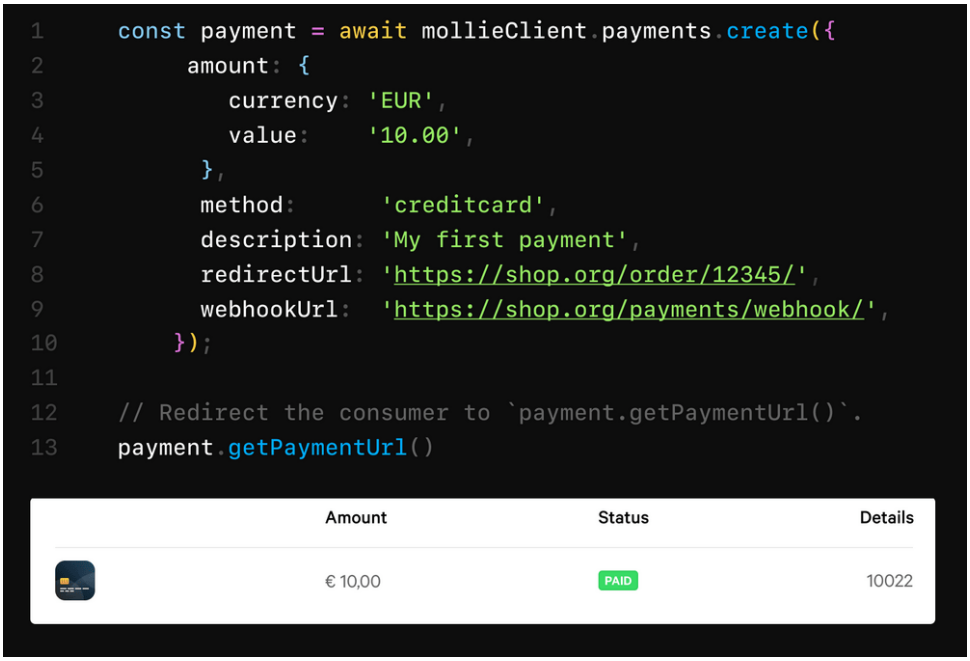
Je ziet drie kolommen in het sequentiediagram van het betalingsproces: frontend, backend en betalingsaanbieder. Elke kolom vertegenwoordigt een andere computer die met de anderen communiceert. De frontend draait op het apparaat van de gebruiker, de backend-server draait op een VPS die door mij wordt beheerd en de betalingsaanbieder is een externe server. De gebruiker interageert met de website door op de knop ”koop (me koffie)” te klikken, dit activeert een reeks API-aanroepen die van rechts naar links stromen.

De ”backend-server” is verantwoordelijk voor het berekenen van het bedrag dat de gebruiker moet betalen. Het wordt berekend op de server, niet in de client, om te voorkomen dat kwaadwillende gebruikers de prijsberekening manipuleren (aangezien frontend-technologie kan worden gemanipuleerd via de webbrowser).

Een betalingssessie wordt voorbereid door de betalingsaanbieder te informeren dat de gebruiker een betaling voor een bepaald bedrag zal doen. Dit proces maakt gebruik van een geheime API-sleutel waarmee de betalingsaanbieder weet naar welke bankrekening betalingen moeten gaan. De betalingsaanbieder retourneert een betalings-URL, die wordt gebruikt om de gebruiker om te leiden naar de daadwerkelijke betaalpagina.

Mollie is een betaalprovider die veel verschillende betaalmethoden ondersteunt, waaronder creditcards, PayPal, iDEAL, Sofort, Giropay, SEPA, diverse bank apps en Apple Pay

De betalingsaanbieder die ik gebruik is Mollie, een in Amsterdam gevestigd bedrijf met een fantastisch betaalplatform. Hun API's zijn goed gedocumenteerd en ze bieden pakketten om hun betaalsysteem eenvoudig te implementeren in je backend-server.



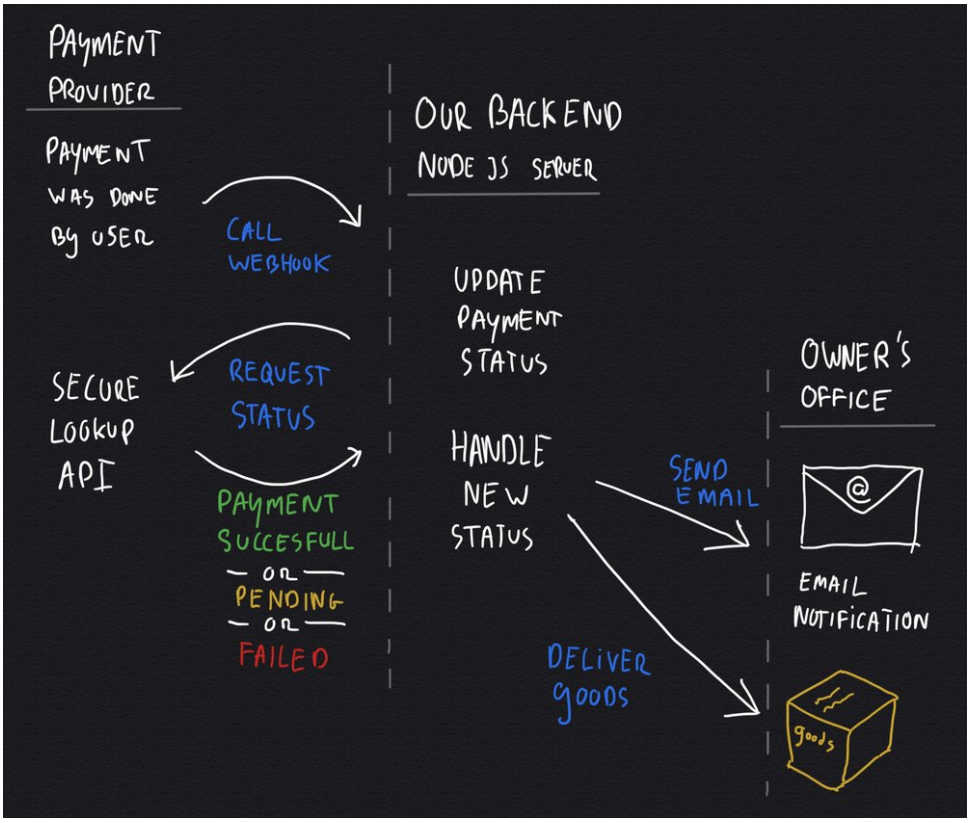
[image] Mollie aanroepen vanuit NodeJS met behulp van promise based JavaScript

In harmonie met mijn andere backend-servers, kies ik ervoor om de backend-server te implementeren in NodeJS. Het draait op praktisch alles (van een kleine Raspberry PI tot grote computerclouds). Als alternatief kun je PHP, Ruby, Python of een van de andere beschikbare pakketten gebruiken. De Mollie API referentie biedt gedetailleerde informatie over de waarden die je kunt verwachten.

Asynchrone updates van de betalingsstatus

Weinig mensen beseffen dat het ophalen van de betalingsstatus een asynchroon proces is dat verschillende statussen kan hebben. In een perfecte wereld weet je meteen of een betaling slaagt of mislukt. Maar vanwege de gedistribueerde aard van online betalingen, waarbij backends communiceren met verschillende servers (banken, creditcards, etc.), weet je nooit precies wanneer een betaling zal worden voltooid.

Denk aan situaties waarin een gebruiker de betaalpagina sluit, zijn of haar apparaat onbeheerd achterlaat, of wanneer een bank een storing heeft met hun online banksysteem. Het gebeurt, en het is jouw taak om de backend-server zo te ontwerpen dat updates van de betalingsstatus afzonderlijk van het primaire gebruikersinteractieproces worden afgehandeld.



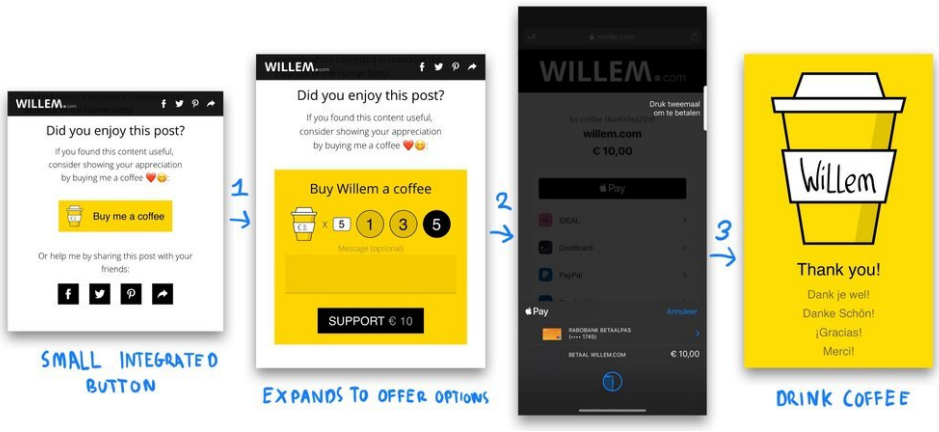
[image] Omgaan met updates van de betaalstatus

De betalingsaanbieder roept een webhook aan op onze backend-server. Het levert een bericht zoals "Kijk, er is iets veranderd aan betaling XYZ". Dit bericht zegt alleen dat *iets* is gewijzigd, het vertelt ons niet *wat* de nieuwe status is. Dit is om veiligheidsredenen: anders zouden kwaadwillende gebruikers de oproep naar onze webhook kunnen nabootsen. In plaats daarvan vraagt onze backend-server de daadwerkelijke betalingsstatus op via een beveiligd kanaal met behulp van onze geheime API-sleutel. De betalingsaanbieder retourneert de betalingsstatus die vervolgens door onze backend-server wordt afgehandeld. Afhankelijk van de betalingsstatus worden passende acties ondernomen, zoals het versturen van e-mailmeldingen, het starten van de levering van goederen.

Fullstack: Frontend en backend combineren

De laatste stap is het verbinden van de frontend met de backend-technologie, zodat het hele proces werkt. Dit is wat mensen "fullstack-ontwikkeling" noemen, omdat je aan het hele systeem werkt. Fullstack-ontwikkeling is een ander soort kunst die maar weinig ontwikkelaars echt beheersen, het vereist dat je tegelijkertijd aan meerdere kanten werkt (debuggen en ontwikkelen); vaak in verschillende programmeertalen tegelijk!

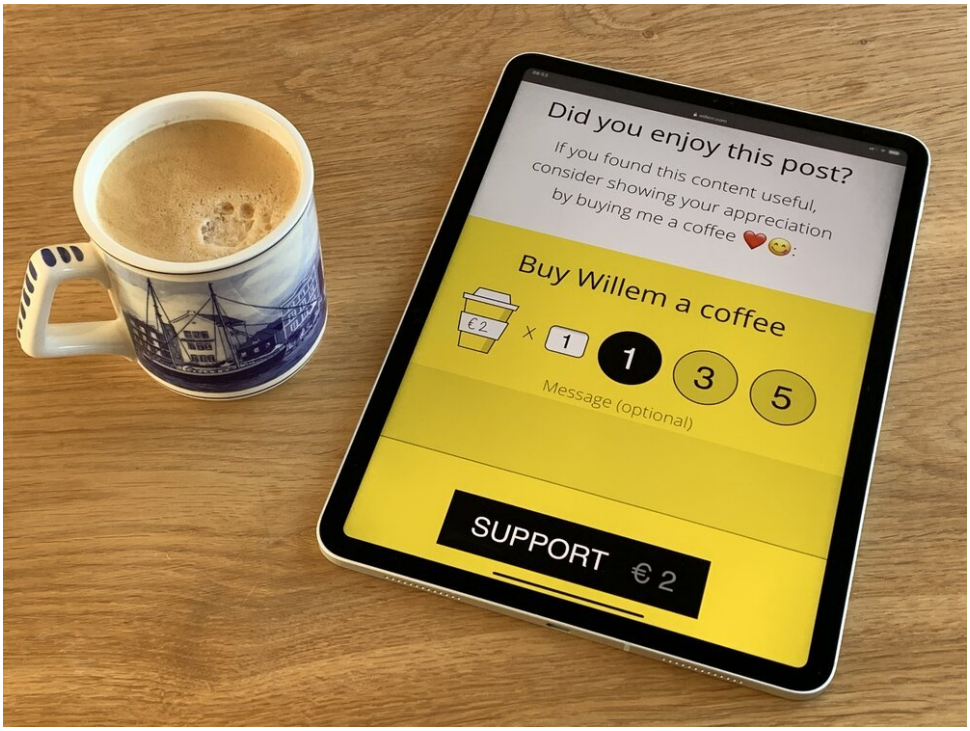
In de praktijk betekent dit dat je verschillende delen van de programmering in de gaten houdt, om te zien of ze goed samenwerken. Serverlogs bekijken terwijl je probeert een betaling uit te voeren met behulp van de nieuw gemaakte frontend-technologie. Het is moeilijk, maar het is ook zeer lonend omdat het resulteert in een werkend product!



[image] Frontend met backend technologie verbinden, simpel als 1-2-3

De resulterende ”koop me een koffie”-knop is een bedrieglijk eenvoudig proces van 3 stappen voor de gebruiker:

- **stap 1**: klik op een simpele ”koop me een koffie”-knop, geen ingewikkeld bestelformulier, geen irritante captcha-code: gewoon een enkele knop
- **stap 2**: de interface breidt zich uit op zijn plaats, behoudt de context en biedt de mogelijkheid om een bericht achter te laten of me meerdere kopjes koffie te kopen. Bovendien worden andere interface-elementen, zoals de social media-knoppen, automatisch verborgen om de focus op de koffie-interface te vergroten
- **stap 3**: het scherm van de betalingsaanbieder stelt de gebruiker in staat om de betaling uit te voeren met behulp van elke beschikbare betaalmethode, zoals Apple Pay. Wanneer de betaling slaagt, is het tijd om koffie te drinken!



Tijd om koffie te drinken

Conclusie

Het ontwerpen en implementeren van een betaalsysteem is niet zo moeilijk als je eenmaal weet wat je moet doen. Toch zijn er veel stappen nodig om dat begrip te bereiken. Hoewel mijn "koop-me-een-koffie"-knop eenvoudig lijkt, vereiste het het oplossen van veel verschillende ontwerp- en programmeeruitdagingen.

Het is moeilijk om het belang van eenvoud te overschatten, maar het is vaak erg moeilijk te bereiken omdat het een begrip vereist van alle aspecten van de uitdaging. Om Einstein te citeren: *"De definitie van genie is het complexe nemen en het simpel maken."* Nu, voel je vrij om de knop zelf te proberen: hij staat hier