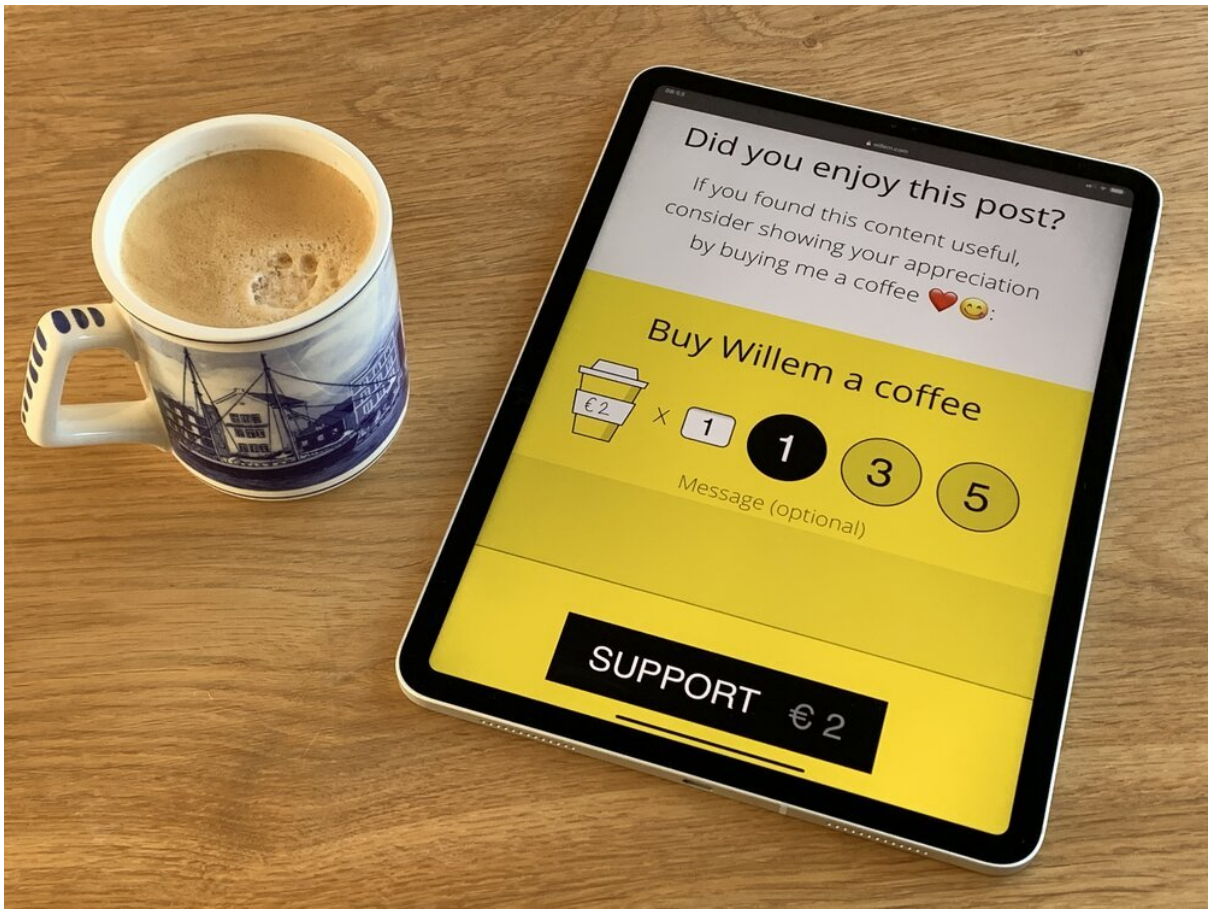
Design und Implementierung eines (Mikro-)Zahlungssystems

Monetarisierung meines Blogs mit Kaffee, Apple Pay und Mollie

Willem L. Middelkoop
Mar. 25, 2020



Online-Zahlungen sind heute wichtiger denn je, da Unternehmen durch das COVID-19-Virus beeinträchtigt werden. Es treibt meine Kunden dazu, neue Wege zu suchen, online Geld zu verdienen. Ich habe ein (Mikro-)Zahlungssystem entworfen und implementiert. In diesem Beitrag geht es darum, Einfachheit durch die Lösung komplexer Herausforderungen zu erreichen.

Neue Technologie verstehen

Wenn ich neue Technologien und Apps entwickle, versuche ich mir immer vorzustellen, wie sie am besten funktionieren würden. Der einfachste Weg, das herauszufinden, ist, das

Produkt für sich selbst zu entwickeln. Genau wie Steve Jobs Apple ihre Präsentations-App Keynote für sich selbst entwickeln ließ. Es ist einer seiner berühmten Witze darüber, ein unterbezahlter Betatester zu sein

Steve Jobs kündigt die Keynote App auf der MacWorld 2003 an und scherzt, dass sie für ihn selbst entwickelt wurde, da er ein unterbezahlter Betatester sei

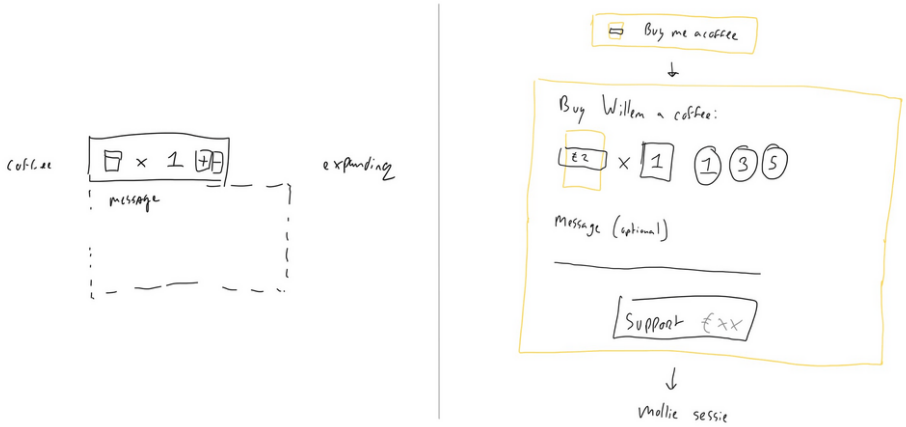
Wenn Sie das Produkt für sich selbst entwickeln, überlegen Sie, was Sie wirklich, *wirklich* wollen. Für mich bedeutet das das absolut einfachste Benutzerinteraktionsmodell: Ich möchte, dass meine Software einfach und leicht zu bedienen ist.

Anwendungsfall: Kauf mir einen Kaffee

Für die Entwicklung der Zahlungstechnologie habe ich mir den Anwendungsfall ausgedacht, einen "Kauf mir einen Kaffee"-Button auf meiner Website hinzuzufügen. Wenn es gut funktioniert, können die Leute damit ihre Wertschätzung für diesen Blog zeigen, indem sie mir einen Kaffee kaufen. Ich denke, dieser Anwendungsfall ist eine perfekte Herausforderung, um ein (Mikro-)Zahlungssystem zu entwerfen (und zu bauen).

Ideen skizzieren

Bevor ich eine einzige Zeile Code programmiere, entwerfe ich Software oft, indem ich Ideen von Hand skizziere. Ich finde den kreativen Prozess des Zeichnens hilfreich, um Ideen, Layouts und Prozesse zu erforschen. Das ist einer der Gründe, warum ich so gerne auf einem Tablet arbeite. Das digitale Skizzieren von Ideen ermöglicht es mir, schnell mehrere Ideen zu erforschen, indem ich sie kopiere und in mehrere Iterationen einfüge.



Interface-Konzeptskizzen, die verschiedene Optionen zum Hinterlassen einer Nachricht und zum Kauf mehrerer Tassen Kaffee untersuchen

Prototyping

Sobald Sie Ideen für Ihre Benutzeroberfläche erforscht haben, ist es an der Zeit, einen Prototyp zu entwickeln. Wenn Sie einen Prototyp erstellen, können Sie die Benutzeroberfläche auf verschiedenen Geräten testen und sehen, wie sie aussieht und sich anfühlt.

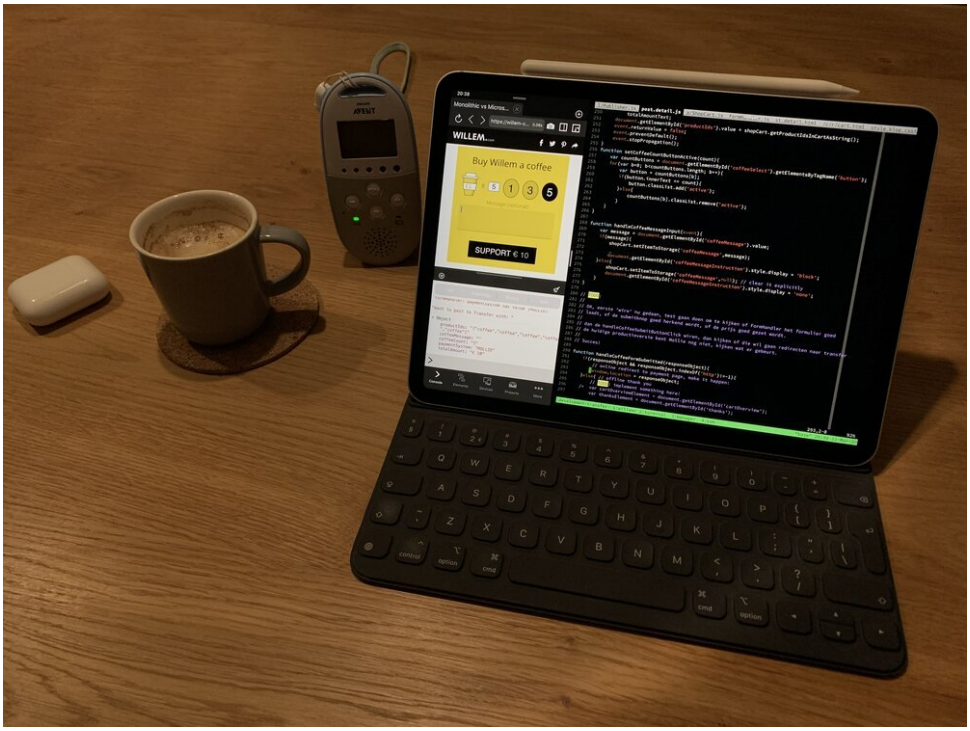
Übersetzung der Designskizze (links) in einen Prototyp (rechts) auf dem iPad

Während des Prototypings schaue ich mir oft die Designskizze an, während ich die erste Version programmiere. Auf dem iPad kann man das mit geteiltem Bildschirm machen, wobei ich die Designskizze direkt neben meinem Code platziere, indem ich VIM in tmux über Mosh mit der brillanten Blink App für iPadOS verwende. Die Kaffeetasse habe ich mit Picta Graphic für iPad erstellt, einer wunderschönen Zeichen-App, die in Vektorgrafiken (SVG) exportiert.

Gestaltung der Kaffeetasse mit Picta Graphic für iPad

Frontend-Entwicklung

Der nächste Schritt ist die Entwicklung einer voll funktionsfähigen Benutzeroberfläche, bei der die Schaltflächen funktionieren. Während der Entwicklung muss man oft herausfinden, was vor sich geht, Fehler verfolgen und Mechanismen feinabstimmen. Ich benutze den Inspect Browser für iPadOS, um eine funktionierende Miniatur-Touchscreen-Oberfläche direkt neben meinem Code zu haben. Das erlaubt mir, die funktionierende Benutzeroberfläche schnell zu testen und zu perfektionieren.

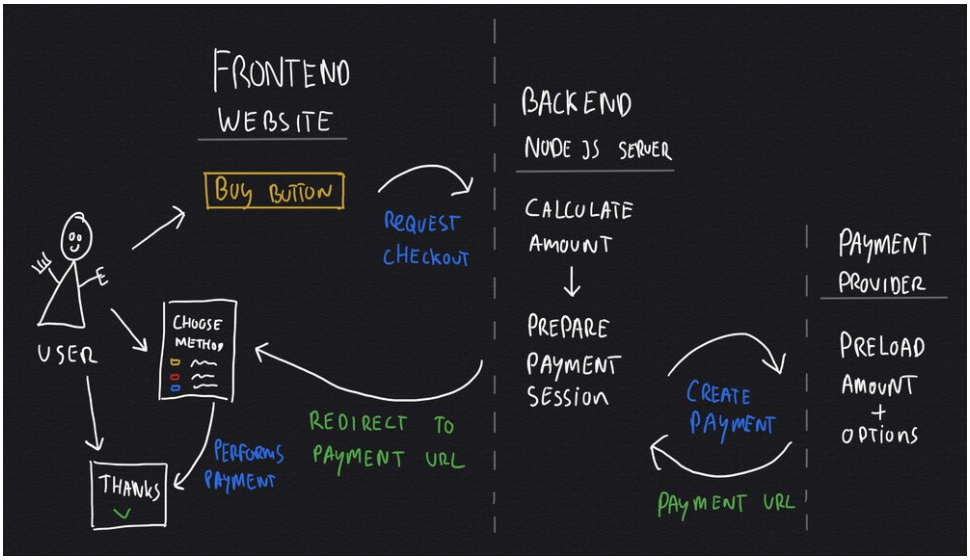


Entwicklung der Benutzeroberfläche mit Blink und Inspect Browser auf dem iPad

Backend-Entwicklung

Ein weiterer wichtiger Schritt ist die Entwicklung der Backend-Technologie zur Vorbereitung und Verfolgung von Online-Zahlungen. Dies ist eine serverseitige Technologie, die die Kommunikation zwischen der Frontend-Benutzeroberfläche (Website) und dem Zahlungsanbieter (Banken, Kreditkarten, Apple Pay usw.) übernimmt.

Das Entwerfen des Backend-Servers ist in der Regel etwas abstrakter als das Entwerfen einer (sichtbaren) Frontend-Benutzeroberfläche. Ich finde das Skizzieren von UML-Sequenzdiagrammen sehr nützlich, um herauszufinden, welche API-Aufrufe in welcher Reihenfolge erfolgen sollen. Sequenzdiagramme visualisieren Programminteraktionen in zeitlicher Abfolge.



Skizziertes Sequenzdiagramm des Zahlungsvorgangs

Sie sehen drei Spalten im Sequenzdiagramm des Zahlungsprozesses: Frontend, Backend und Zahlungsanbieter. Jede Spalte repräsentiert einen anderen Computer, der mit den anderen kommuniziert. Das Frontend läuft auf dem Gerät des Benutzers, der Backend-Server läuft auf einem von mir verwalteten VPS und der Zahlungsanbieter ist ein externer Server. Der Benutzer interagiert mit der Website, indem er auf die Schaltfläche "Kaufen (mir einen Kaffee)" klickt. Dies löst eine Kette von API-Aufrufen aus, die von rechts nach links fließen.

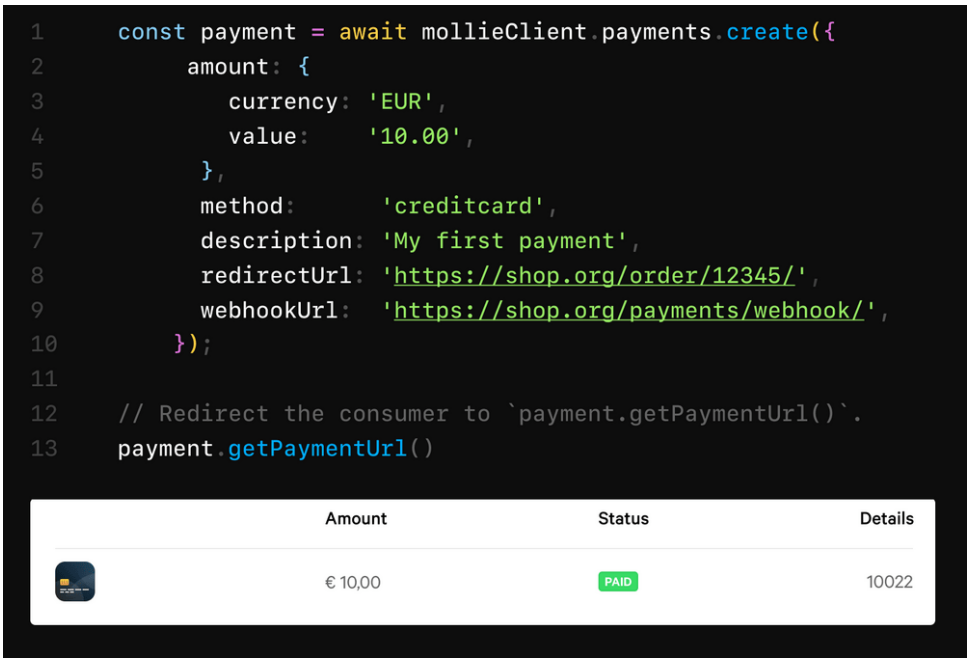
Der "Backend-Server" ist für die Berechnung des Betrags verantwortlich, den der Benutzer bezahlen muss. Die Berechnung erfolgt auf dem Server und nicht im Client, um zu verhindern, dass böswillige Benutzer die Preisberechnung manipulieren (da die Frontend-Technologie über den Webbrowser manipuliert werden kann).

Eine Zahlungssitzung wird vorbereitet, indem der Zahlungsanbieter darüber informiert wird, dass der Benutzer eine Zahlung über einen bestimmten Betrag tätigen wird. Dieser Prozess verwendet einen geheimen API-Schlüssel, der es dem Zahlungsanbieter ermöglicht zu wissen, auf welches Bankkonto die Zahlungen gehen sollen. Der Zahlungsanbieter gibt eine Zahlungs-URL zurück, die verwendet wird, um den Benutzer auf die eigentliche Zahlungsseite umzuleiten.

Mollie ist ein Zahlungsanbieter, der viele verschiedene Zahlungsmethoden unterstützt, darunter Kreditkarten, PayPal, iDEAL, Sofort, Giropay, SEPA, verschiedene Banking-Apps und Apple Pay

Der von mir verwendete Zahlungsanbieter ist Mollie, ein Unternehmen mit Sitz in Amsterdam mit einer fantastischen Zahlungsplattform. Ihre APIs sind gut dokumentiert

und sie bieten Pakete an, um ihr Zahlungssystem einfach in Ihren Backend-Server zu implementieren.



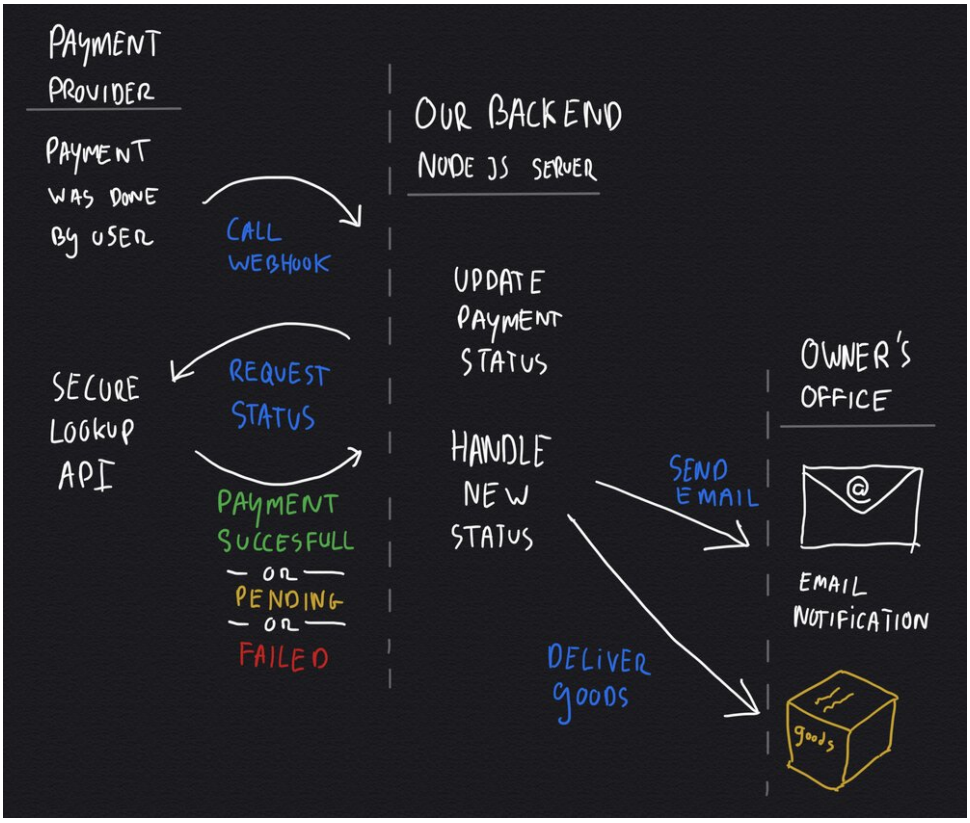
Aufruf von Mollie aus NodeJS mit Promise-basiertem JavaScript

In Übereinstimmung mit meinen anderen Backend-Servern habe ich mich entschieden, den Backend-Server in NodeJS zu implementieren. Er läuft praktisch auf allem (von einem winzigen Raspberry PI bis hin zu großen Computing-Clouds). Alternativ können Sie PHP, Ruby, Python oder eines der anderen verfügbaren Pakete verwenden. Die Mollie API Referenz bietet detaillierte Informationen zu den Werten, die Sie erwarten können.

Asynchrone Aktualisierungen des Zahlungsstatus

Nur wenige Menschen wissen, dass das Abrufen des Zahlungsstatus ein asynchroner Prozess ist, der verschiedene Status haben kann. In einer perfekten Welt wüsste man sofort, ob eine Zahlung erfolgreich ist oder fehlschlägt. Aber aufgrund der verteilten Natur von Online-Zahlungen, bei denen Backends mit verschiedenen Servern (Banken, Kreditkarten usw.) kommunizieren, weiß man nie genau, wann eine Zahlung abgeschlossen sein wird.

Denken Sie an Situationen, in denen ein Benutzer die Zahlungsseite schließt, sein Gerät unbeaufsichtigt lässt oder wenn eine Bank eine Unterbrechung ihres Online-Banking-Systems hat. Es passiert, und es ist Ihre Aufgabe, den Backend-Server so zu gestalten, dass er Aktualisierungen des Zahlungsstatus getrennt vom primären Benutzerinteraktionsprozess verarbeitet.



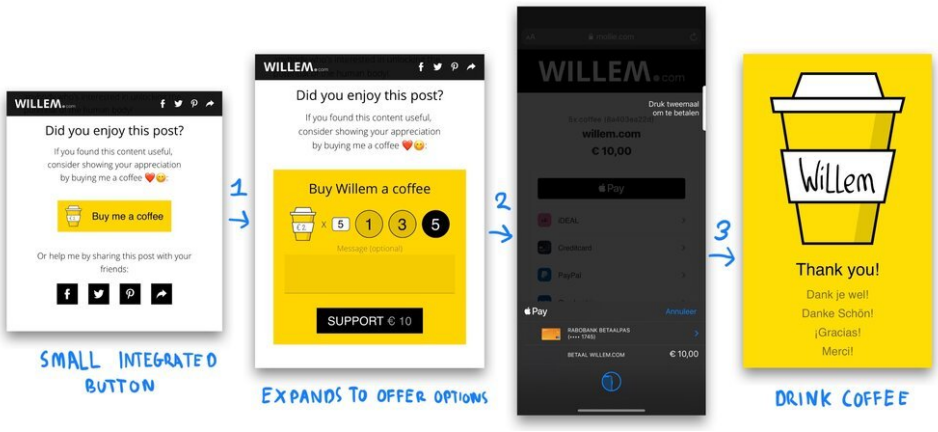
Bearbeitung von Zahlungsstatus-Updates

Der Zahlungsanbieter ruft einen Webhook auf unserem Backend-Server auf. Er liefert eine Nachricht wie "Schau, bei der Zahlung XYZ hat sich etwas geändert". Diese Nachricht sagt nur, dass sich *etwas* geändert hat, sie sagt uns nicht, *was* der neue Status ist. Dies geschieht aus Sicherheitsgründen: Andernfalls könnten böswillige Benutzer den Aufruf unseres Webhooks nachahmen. Stattdessen fordert unser Backend-Server den tatsächlichen Zahlungsstatus über einen sicheren Kanal mit unserem geheimen API-Schlüssel an. Der Zahlungsanbieter gibt den Zahlungsstatus zurück, der dann von unserem Backend-Server verarbeitet wird. Abhängig vom Zahlungsstatus werden entsprechende Maßnahmen ergriffen, wie z. B. das Versenden von E-Mail-Benachrichtigungen oder der Beginn der Warenlieferung.

Fullstack: Kombination von Frontend und Backend

Der letzte Schritt ist die Verbindung des Frontends mit der Backend-Technologie, damit der gesamte Prozess funktioniert. Das ist es, was man "Fullstack-Entwicklung" nennt, da man am gesamten System arbeitet. Fullstack-Entwicklung ist eine andere Art von Kunst, die nur wenige Entwickler wirklich beherrschen, denn sie erfordert, dass man an mehreren Enden gleichzeitig arbeitet (debuggt und entwickelt); oft in verschiedenen Programmiersprachen gleichzeitig!

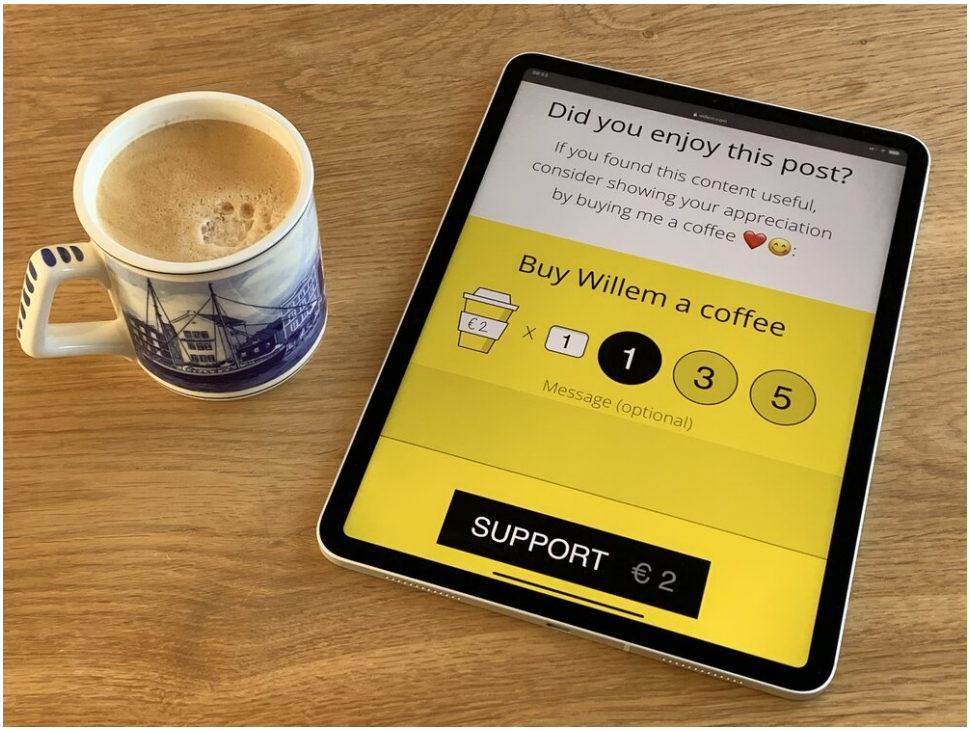
In der Praxis bedeutet dies, dass Sie verschiedene Teile der Programmierung überwachen und sehen, ob sie gut zusammenarbeiten. Sie beobachten die Serverprotokolle, während Sie versuchen, eine Zahlung mit der neu erstellten Frontend-Technologie durchzuführen. Es ist schwierig, aber es ist auch sehr lohnend, da es zu einem funktionierenden Produkt führt!



Frontend mit Backend-Technologie verbinden, einfach wie 1-2-3

Der resultierende "Kauf mir einen Kaffee"-Button ist ein täuschend einfacher 3-Schritte-Prozess für den Benutzer:

- **Schritt 1**: Klicken Sie auf einen einfachen "Kauf mir einen Kaffee"-Button, kein komplexes Bestellformular, kein irritierender Captcha-Code: nur ein einziger Button
- **Schritt 2**: Die Benutzeroberfläche erweitert sich an Ort und Stelle, wobei der Kontext erhalten bleibt und die Möglichkeit geboten wird, eine Nachricht zu hinterlassen oder mir mehrere Tassen Kaffee zu kaufen. Zusätzlich werden andere Oberflächenelemente wie die Social-Media-Buttons automatisch ausgeblendet, um den Fokus auf die Kaffee-Oberfläche zu verstärken.
- **Schritt 3**: Der Bildschirm des Zahlungsanbieters ermöglicht es dem Benutzer, die Zahlung mit jeder verfügbaren Zahlungsmethode durchzuführen, z. B. mit Apple Pay. Wenn die Zahlung erfolgreich ist, ist es Zeit, Kaffee zu trinken!



Zeit für Kaffee

Fazit

Das Entwerfen und Implementieren eines Zahlungssystems ist nicht so schwer, wenn man erst einmal weiß, was man tun muss. Dennoch sind viele Schritte erforderlich, um dieses Verständnis zu erlangen. Obwohl mein "Kauf-mir-einen-Kaffee"-Button einfach erscheint, musste ich viele verschiedene Design- und Programmierherausforderungen lösen.

Man kann die Bedeutung von Einfachheit kaum überschätzen, doch sie ist oft sehr schwer zu erreichen, weil sie ein Verständnis aller Aspekte der Herausforderung erfordert. Um Einstein zu zitieren: *"Die Definition von Genie ist, das Komplexe zu nehmen und es einfach zu machen."* Nun, bitte versuchen Sie den Button selbst: er ist genau hier