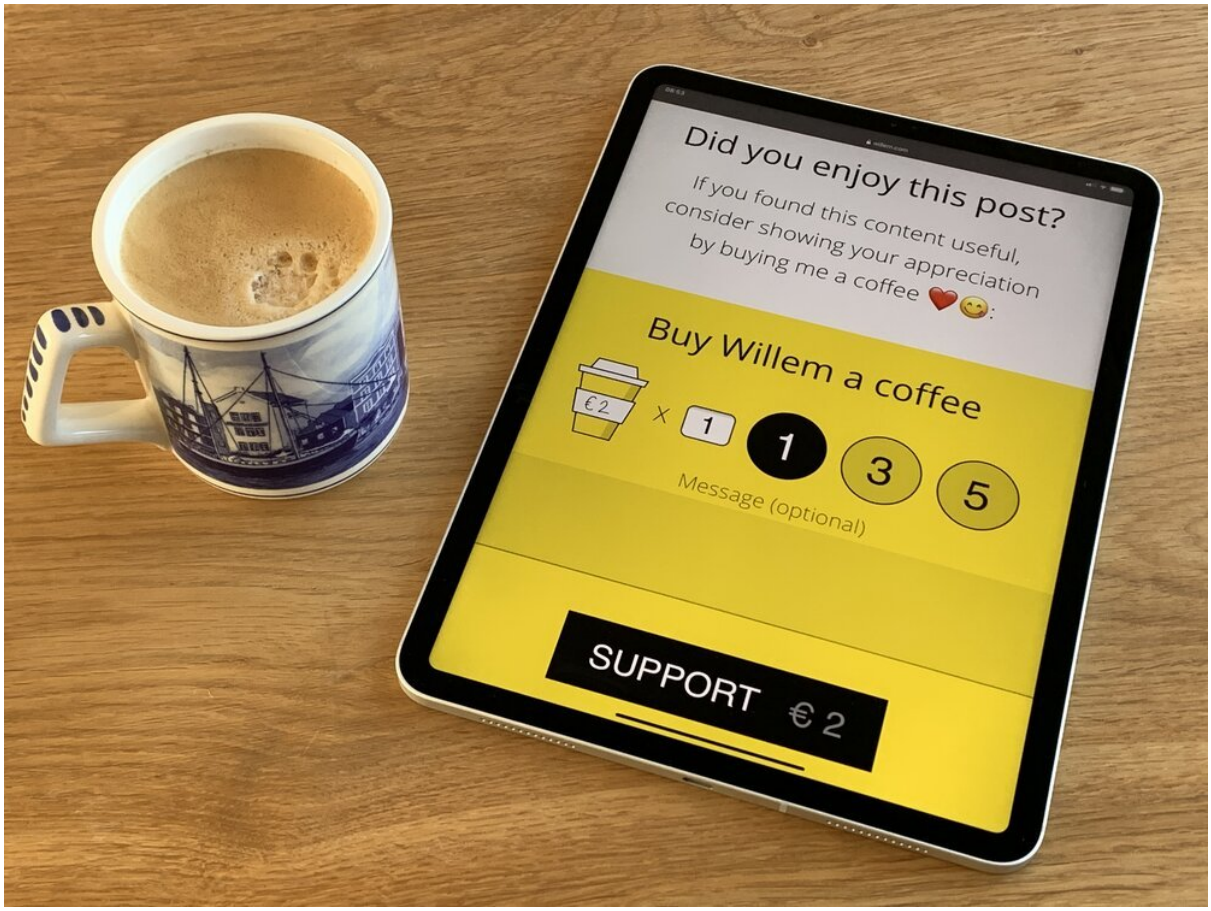
Diseño e implementación de un sistema de (micro)pagos

Monetizando mi blog con café, Apple Pay y Mollie

Willem L. Middelkoop
Mar. 25, 2020



Los pagos online son ahora más importantes que nunca ya que los negocios se ven afectados por el virus COVID-19. Esto impulsa a mis clientes a buscar nuevas formas de ganar dinero online. Diseñé e implementé un sistema de (micro)pagos. Esta publicación trata sobre cómo lograr la simplicidad resolviendo desafíos complejos.

Entendiendo la nueva tecnología

Al desarrollar nueva tecnología y aplicaciones, siempre trato de imaginar cómo funcionaría mejor. La forma más fácil de averiguarlo es diseñar el producto para uno mismo. Al igual

que Steve Jobs hizo que Apple desarrollara su aplicación de presentaciones, Keynote, para él mismo. Es una de sus famosas bromas sobre ser un probador beta mal pagado

Steve Jobs anunciando la aplicación Keynote en la MacWorld 2003, bromeando que la había creado para sí mismo, siendo él un beta tester mal pagado

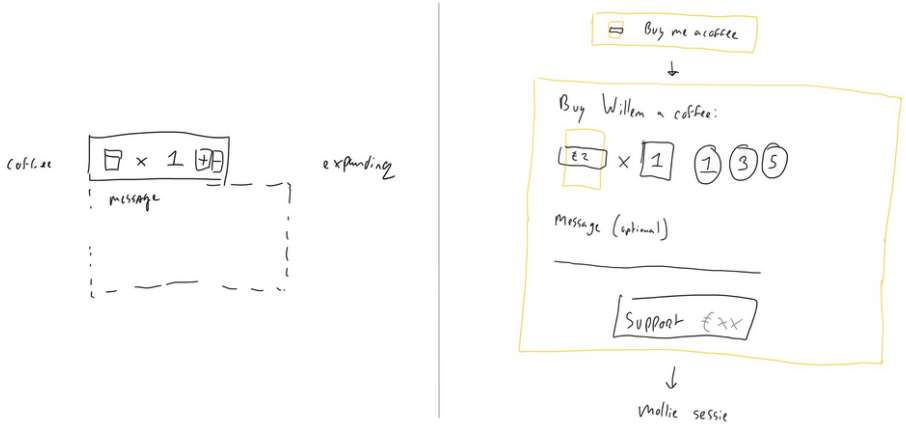
Si diseñas el producto para ti mismo, piensa en lo que realmente, *realmente*, querrías. Para mí, esto significa el modelo de interacción de usuario más simple posible: quiero que mi software sea simple, fácil.

Caso de uso: invítame un café

Con el propósito de desarrollar la tecnología de pago, se me ocurrió el caso de uso de agregar un botón de "invítame un café" a mi sitio web. Si funciona bien, la gente puede usarlo para mostrar su aprecio por este blog comprándome un café. Creo que este caso de uso es un desafío perfecto para diseñar (y construir) un sistema de pago (micro).

Bosquejando ideas

Antes de programar una sola línea de código, a menudo diseño software bosquejando ideas a mano. Encuentro que el proceso creativo de dibujar es útil para explorar ideas, diseños, procesos. Es una de las razones por las que me gusta tanto trabajar en una tableta, el bosquejo digital de ideas me permite explorar rápidamente múltiples ideas copiándolas/pegándolas en múltiples iteraciones.



[image]

Prototipado

Una vez que hayas explorado ideas para tu interfaz de usuario, es hora de desarrollar un prototipo. Cuando construyes un prototipo, puedes probar la interfaz de usuario en diferentes dispositivos, ver cómo se ve y se siente.

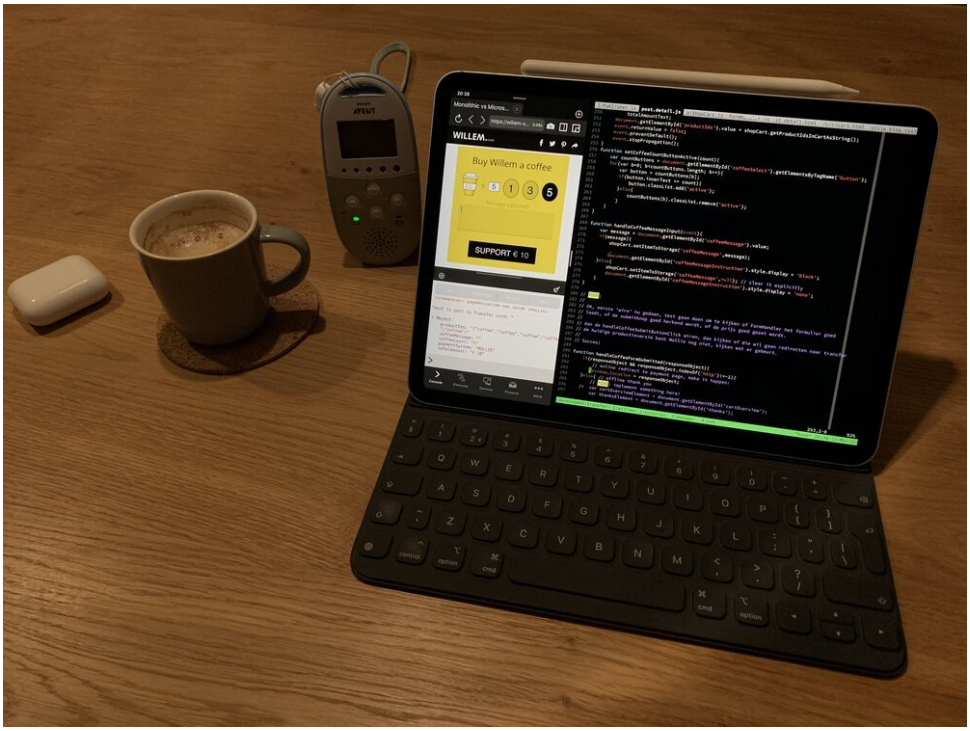
Bocetos de concepto de interfaz, explorando diferentes opciones para dejar un mensaje y comprar varias tazas de café

Durante el prototipado, a menudo miro el boceto de diseño mientras programo la primera versión. En iPad puedes hacer esto usando la pantalla dividida, donde coloco el boceto de diseño justo al lado de mi código, usando VIM en tmux sobre Mosh con la brillante aplicación Blink para iPadOS. Creé la taza de café usando Picta Graphic para iPad, una hermosa aplicación de dibujo que exporta a imágenes vectoriales (SVG).

[image]

Desarrollo frontend

El siguiente paso es desarrollar una interfaz de usuario totalmente funcional, haciendo que los botones funcionen. Durante el desarrollo, a menudo necesitas averiguar qué está pasando, rastrear errores y ajustar mecanismos. Utilizo Inspect Browser para iPadOS para tener una interfaz de pantalla táctil en miniatura que funcione justo al lado de mi código. Me permite probar y perfeccionar rápidamente la interfaz de usuario en funcionamiento.

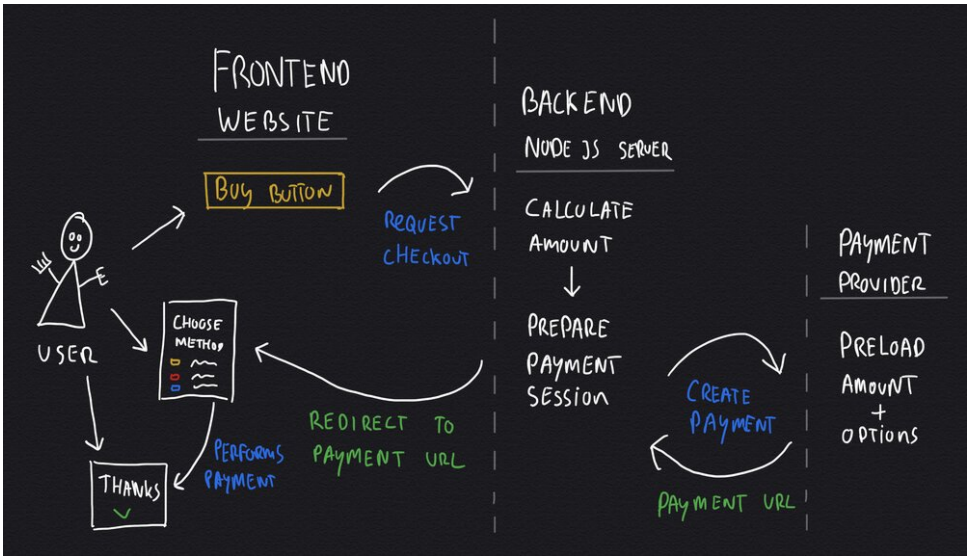


Traduciendo el boceto de diseño (izquierda) en un prototipo (derecha) en iPad

Desarrollo backend

Otro paso crucial es desarrollar la tecnología backend para preparar y rastrear los pagos en línea. Esta es la tecnología del lado del servidor que manejará la comunicación entre la interfaz de usuario frontend (sitio web) y el proveedor de pagos (bancos, tarjetas de crédito, Apple Pay, etc.).

Diseñar el servidor backend suele ser un poco más abstracto en comparación con el diseño de una interfaz de usuario frontend (visible). Encuentro que los diagramas de secuencia UML son muy útiles para averiguar qué llamadas a la API deben ocurrir en qué orden. Los diagramas de secuencia visualizan las interacciones del programa en secuencia de tiempo.



[image]

Se ven tres columnas en el diagrama de secuencia del proceso de pago: frontend, backend y proveedor de pagos. Cada columna representa una computadora diferente que se comunica con las demás. El frontend se ejecuta en el dispositivo del usuario, el servidor backend se ejecuta en un VPS administrado por mí y el proveedor de pagos es un servidor externo. El usuario interactúa con el sitio web haciendo clic en el botón "comprar (me un café)", esto desencadena una cadena de llamadas a la API que fluyen de derecha a izquierda.

El "servidor backend" es responsable de calcular la cantidad que el usuario debe pagar. Se calcula en el servidor, no en el cliente, para evitar que usuarios malintencionados manipulen el cálculo del precio (ya que la tecnología frontend se puede manipular a través del navegador web).

Se prepara una sesión de pago informando al proveedor de pagos que el usuario realizará un pago por una cantidad determinada. Este proceso utiliza una clave API secreta que permite al proveedor de pagos saber a qué cuenta bancaria deben ir los pagos. El proveedor de pagos devuelve una URL de pago, que se utiliza para redirigir al usuario a la página de pago real.

Diseñando la taza de café usando Picta Graphic para iPad

El proveedor de pagos que utilizo es Mollie, una empresa con sede en Ámsterdam con una fantástica plataforma de pago. Sus API están bien documentadas y proporcionan paquetes para implementar fácilmente su sistema de pago en su servidor backend.

```
const payment = await mollieClient.payments.create({
     amount: {
        currency: 'EUR',
        value:    '10.00',
      },
      method:      'creditcard',
      description: 'My first payment',
      redirectUrl: 'https://shop.org/order/12345/',
      webhookUrl:  'https://shop.org/payments/webhook/',
    });

// Redirect the consumer to `payment.getPaymentUrl()`.
payment.getPaymentUrl()
```

	Amount	Status	Details
	€ 10,00	PAID	10022

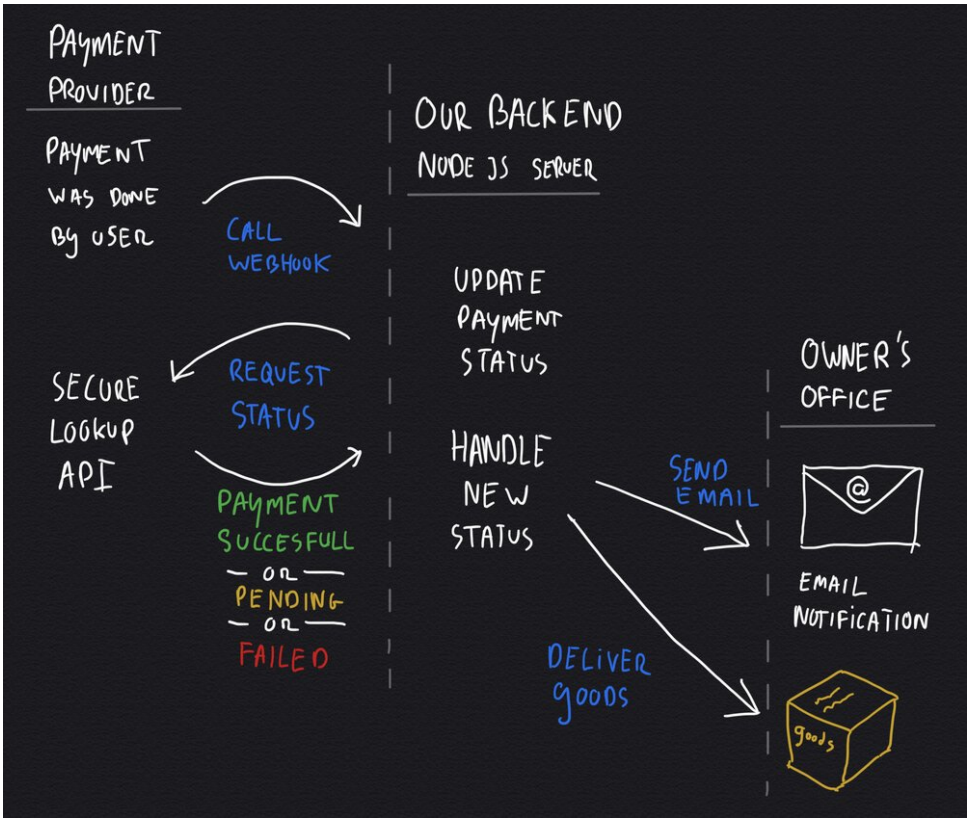
[image]

En armonía con mis otros servidores backend, elijo implementar el servidor backend en NodeJS. Se ejecuta en prácticamente cualquier cosa (desde una pequeña Raspberry PI hasta grandes nubes informáticas). Alternativamente, puedes usar PHP, Ruby, Python o cualquiera de los otros paquetes disponibles. La referencia de la API de Mollie ofrece información detallada sobre los valores que puedes esperar.

Actualizaciones asíncronas del estado del pago

Pocas personas se dan cuenta de que recuperar el estado del pago es un proceso asíncrono que puede tener diferentes estados. En un mundo perfecto, se sabe instantáneamente si un pago tiene éxito o falla. Pero debido a la naturaleza distribuida de los pagos en línea, donde los backends se comunican con diferentes servidores (bancos, tarjetas de crédito, etc.), nunca se sabe con precisión cuándo se completará un pago.

Piensa en situaciones en las que un usuario cierra la página de pago, deja su dispositivo desatendido o cuando un banco tiene una interrupción con su sistema de banca en línea. Sucede, y es tu tarea diseñar el servidor backend para manejar las actualizaciones del estado del pago por separado del proceso principal de interacción del usuario.



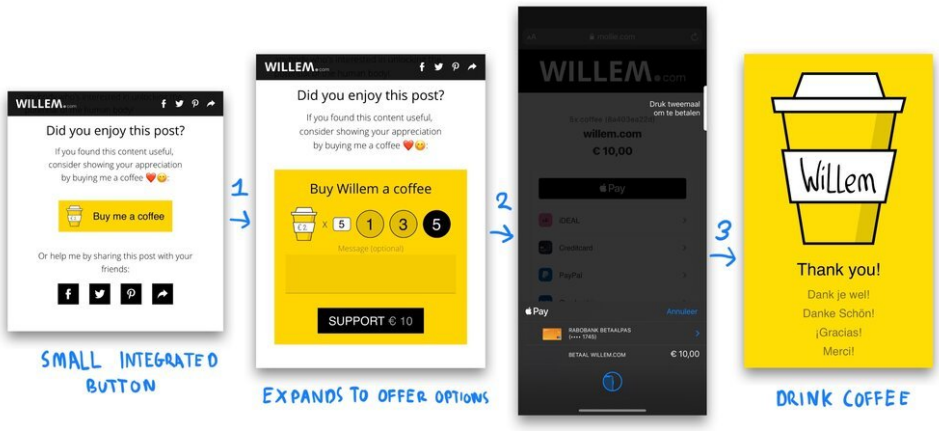
Desarrollando la interfaz de usuario usando Blink e Inspect Browser en iPad

El proveedor de pagos llama a un webhook en nuestro servidor backend. Entrega un mensaje como "Mira, algo cambió en el pago XYZ". Este mensaje solo dice que *algo* cambió, no nos dice *qué* es el nuevo estado. Esto es por razones de seguridad: de lo contrario, los usuarios malintencionados podrían imitar la llamada a nuestro webhook. En cambio, nuestro servidor backend solicita el estado de pago real a través de un canal seguro utilizando nuestra clave API secreta. El proveedor de pagos devuelve el estado del pago, que luego es manejado por nuestro servidor backend. Dependiendo del estado del pago, se toman las medidas apropiadas, como enviar notificaciones por correo electrónico, iniciar la entrega de bienes.

Fullstack: Combinando frontend y backend

El último paso es conectar el frontend con la tecnología backend, haciendo que todo el proceso funcione. Esto es lo que la gente llama "desarrollo fullstack", ya que estás trabajando en todo el sistema. El desarrollo fullstack es un tipo diferente de arte que pocos desarrolladores realmente dominan, requiere que trabajes (depures y desarrolles) en múltiples extremos simultáneamente; ¡a menudo en diferentes lenguajes de programación a la vez!

En la práctica, esto significa que estarás monitoreando diferentes partes de la programación, para ver si funcionan bien juntas. Observando los registros del servidor mientras intentas realizar un pago utilizando la tecnología frontend recién creada. Es difícil, pero también es muy gratificante, ya que da como resultado un producto funcional.



[image]

El botón ”invítame un café” resultante es un proceso engañosamente simple de 3 pasos para el usuario:

- **paso 1**: haz clic en un simple botón ”invítame un café”, sin formularios de pedido complejos, sin irritantes códigos captcha: solo un botón
- **paso 2**: la interfaz se expande en su lugar, preservando el contexto y ofreciendo la opción de dejar un mensaje o comprarme varias tazas de café. Además, otros elementos de la interfaz, como los botones de redes sociales, se ocultan automáticamente para mejorar el enfoque en la interfaz del café.
- **paso 3**: la pantalla del proveedor de pagos permite al usuario realizar el pago utilizando cualquier método de pago disponible, como Apple Pay. Cuando el pago se realiza correctamente, ¡es hora de tomar café!

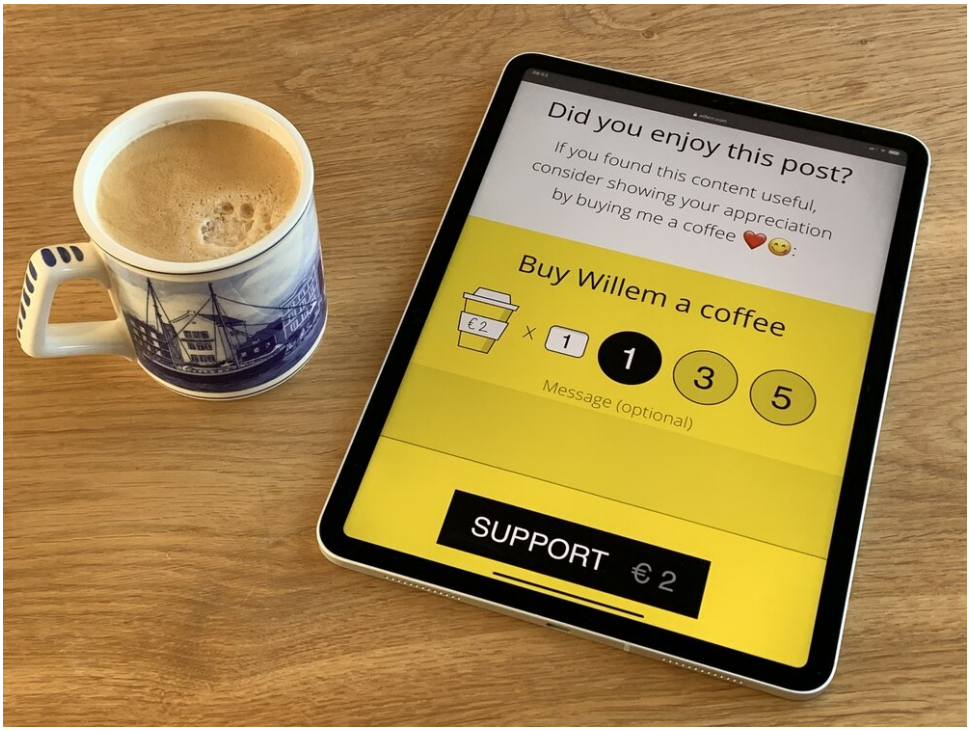


Diagrama de secuencia dibujado del proceso de pago

Conclusión

Diseñar e implementar un sistema de pago no es tan difícil una vez que sabes lo que necesitas hacer. Sin embargo, implica muchos pasos para lograr esa comprensión. Si bien mi botón "invítame un café" parece simple, requirió resolver muchos desafíos diferentes de diseño y programación.

Es difícil exagerar la importancia de la simplicidad, pero a menudo es muy difícil de lograr porque requiere una comprensión de todos los aspectos del desafío. Para citar a Einstein: *"La definición de genio es tomar lo complejo y hacerlo simple."* Ahora, no dudes en probar el botón tú mismo: está justo aquí