Tesi - Payment Request API

Daniele Rigon - 857319

26 giugno 2018

Indice

1 Overview Payment Request

Fare acquisti sul web, in particolare sui dispositivi mobili, può essere un'esperienza frustrante. Ogni sito Web ha il proprio flusso e la maggior parte richiede agli utenti di digitare manualmente gli stessi indirizzi, le informazioni di contatto e le credenziali di pagamento ancora e ancora. Ciò può portare all'abbandono del carrello della spesa e alla perdita della fedeltà dei clienti. Allo stesso modo, è difficile e richiede molto tempo agli sviluppatori di creare e gestire pagine di checkout che supportano vari metodi di pagamento. L' API di richiesta di pagamento (e le specifiche di supporto) consentono ai commercianti di creare esperienze di pagamento semplificate. Piuttosto che ridigitare gli indirizzi di spedizione, le informazioni di contatto, le credenziali di pagamento e altre informazioni ancora e ancora sul Web, gli utenti possono memorizzare e riutilizzare le informazioni e completare più rapidamente e accuratamente le transazioni online. Non è un nuovo metodo di pagamento; piuttosto, è un condotto dal metodo di pagamento preferito dall'utente a un commerciante.

1.1 Concetti e utilizzo della richiesta di pagamento

Molti problemi legati all'abbandono degli acquisti online possono essere ricondotti ai moduli di checkout, difficili da usare, lenti da caricare e aggiornare e richiedono più passaggi da completare. L' API di richiesta di pagamento è un sistema che ha lo scopo di eliminare i moduli di checkout. Migliora notevolmente il flusso di lavoro degli utenti durante il processo di acquisto, offrendo un'esperienza utente più coerente e consentendo ai commercianti di sfruttare facilmente diversi metodi di pagamento.

1.2 Vantaggi nell'utilizzo di questa API

- Esperienza di acquisto rapida: gli utenti immettono i propri dati una volta nel browser e sono quindi pronti a pagare beni e servizi sul Web. Non è più necessario compilare ripetutamente gli stessi dettagli su siti diversi.
- Esperienza coerente su ogni sito (che supporta l'API): poiché il foglio di pagamento è controllato dal browser, può personalizzare l'esperienza per l'utente. Ciò può includere la localizzazione dell'interfaccia utente nella lingua preferita dell'utente.
- Accessibilità: poiché il browser controlla gli elementi di input del foglio di pagamento, è in grado di assicurare accessibilità coerente per tastiera e screen reader su ogni sito Web, senza che gli sviluppatori debbano fare nulla. Un browser può anche regolare la dimensione del carattere o il contrasto cromatico del foglio di pagamento, rendendo più comodo all'utente effettuare un pagamento.
- Gestione delle credenziali: gli utenti possono gestire le loro carte di credito e gli indirizzi di spedizione direttamente nel browser. Un browser può anche sincronizzare queste "credenziali" tra dispositivi, rendendo più semplice per gli utenti passare dal desktop al cellulare e viceversa quando si acquistano oggetti.
- Gestione coerente degli errori: il browser può controllare la validità dei numeri delle carte e può comunicare all'utente se una carta è scaduta (o sta per scadere). Il browser può suggerire automaticamente quale carta utilizzare in base ai modelli di utilizzo passati o alle restrizioni del commerciante, o consentire all'utente di dire quale è la loro carta predefinita / preferita.
- Esperienza utente migliorata in vari modi, tra cui: meno tipizzazione, coerenza tra i siti Web, coerenza tra browser e sistemi operativi e nuove funzionalità del browser per semplificare il checkout.
- Conversione aumentata (ovvero: abbandono del carrello inferiore). Anche se non disponiamo ancora di dati significativi per confermare questa aspettativa, Google ha segnalato che il completamento automatico da solo può aumentare le conversioni di 25; questa API intende migliorare sulla funzione di completamento automatico.
- Sviluppo più facile delle pagine di pagamento per i commercianti o i loro fornitori di servizi.
- Miglioramento della sicurezza. L'API di richiesta di pagamento ha il potenziale per ridurre le opportunità di frode e può facilitare l'adozione di metodi di pagamento più sicuri.

- Responsabilità inferiore In passato, per creare un'esperienza utente semplificata, i commercianti dovevano memorizzare le credenziali di pagamento degli utenti. Questo non è più necessario, il che può aiutare a ridurre la responsabilità del commerciante.
- Supporta l'innovazione del metodo di pagamento. Riducendo il costo dell'integrazione di nuovi metodi di pagamento, l'industria dei pagamenti può creare e adottare più facilmente nuovi metodi di pagamento, ad esempio per migliorare la sicurezza.

Per richiedere un pagamento, una pagina Web crea un oggetto Payment request in risposta a un'azione dell'utente che avvia un pagamento, ad esempio facendo clic sul pulsante "Acquista". Il Payment request permette alla pagina web di scambiare informazioni con l'agente utente mentre l'utente fornisce l'input per completare la transazione.

1.3 Come funziona

L'API di richiesta di pagamento consente a un utente di completare una transazione più facilmente riutilizzando le informazioni memorizzate nel browser o in app di pagamento di terze parti. Quando l'utente preme un pulsante in una pagina di checkout collegata all'API, il commerciante utilizza l'API per richiedere il pagamento. Il commerciante fornisce informazioni su prezzo, valuta e un elenco di metodi di pagamento accettati. Il commerciante può inoltre richiedere al browser di creare un'interfaccia utente semplificata per raccogliere l'indirizzo di spedizione, le informazioni di contatto e un numero limitato di elementi aggiuntivi dall'utente. Il browser determina quale delle "app di pagamento" dell'utente sa di supportare i metodi di pagamento accettati dai commercianti. Il browser mostra le app di pagamento corrispondenti all'utente. Se il commerciante ha chiesto al browser di aiutare a raccogliere un indirizzo di spedizione o altri dati, il browser presenta i dati memorizzati per la selezione dell'utente (o richiede all'utente di inserire nuovi dati). L'utente seleziona un'app di pagamento da pagare. L'app di pagamento può comportare ulteriori interazioni con l'utente (ad esempio, per l'autenticazione). Al completamento, l'app di pagamento restituisce i dati tramite l'API al commerciante.

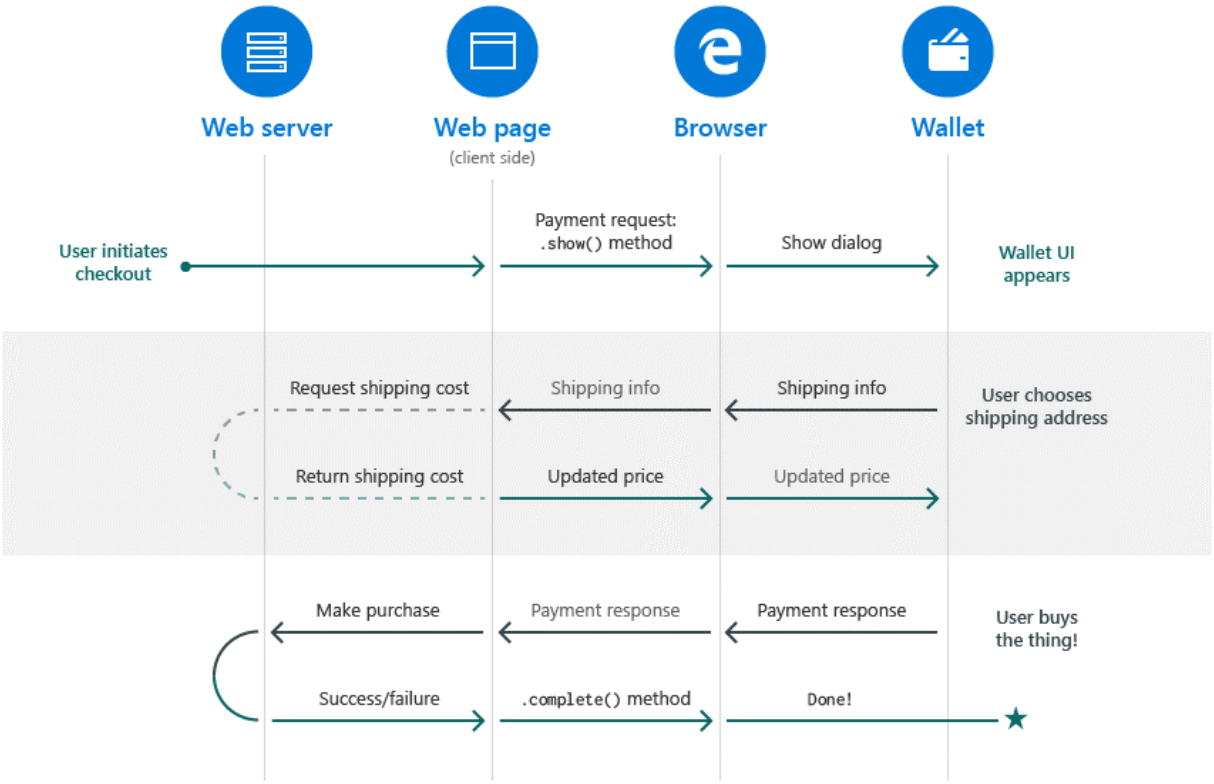

Figura 1: Schema Payment Request API

1.4 Rischi

L'API di richiesta di pagamento consente ai commercianti di "consegnare" la raccolta di credenziali di pagamento alle app di pagamento. Ciò riduce l'onere per i commercianti di creare un'esperienza di pagamento e l'esperienza utente migliorata dovrebbe aumentare le conversioni. Ciò implica, tuttavia, che quando lo fanno, i commercianti non controllano alcuni aspetti dell'esperienza utente.

1.5 Sicurezza API

L'API di richiesta di pagamento aumenta la sicurezza poichè:

- I commercianti possono ottenere un checkout semplificato senza memorizzare le credenziali dell'utente (dato che gli utenti memorizzano le proprie credenziali). Ciò rende i commercianti meno vulnerabili agli attacchi.
- La selezione delle app di pagamento da chrome browser nativo dovrebbe eliminare alcuni comportamenti dannosi che potrebbero influire sulle pagine Web.
- L'API di richiesta di pagamento dovrebbe facilitare l'introduzione di metodi di pagamento più sicuri sul Web, come i pagamenti con carta tokenizzata.
- I proprietari dei metodi di pagamento disporranno di meccanismi standard per autorizzare software specifici a implementare il loro metodo di pagamento, che il browser può verificare (ovvero, una firma digitale).

I browser utilizzano una varietà di meccanismi per archiviare informazioni sensibili dell'utente, inclusi cookie, IndexedDB, memoria locale e funzionalità del sistema operativo in modo sicuro. W3C sta inoltre sviluppando nuove tecnologie correlate come l'autenticazione Web e l'API di gestione delle credenziali. Per quanto riguarda la meorizzazione delle informazioni delle app di pagamento in modo sicuro, questo è un dettaglio di implementazione per ogni app di pagamento. Lo storage nel back-end è interamente a carico del provider di pagamenti.

Inoltre l'API di richiesta di pagamento è richiamata in un contesto sicuro, come indicato dal WebIDL nelle specifiche. Nota: le implementazioni del browser possono variare in ciò che supportano. Ad esempio, Chrome considera i seguenti contesti protetti: "https: //", "file: //" e URL localhost.

1.6 Uso API

1.6.1 Ruolo dell'utente

Gli utenti beneficiano del riutilizzo delle credenziali (come informazioni di pagamento, indirizzo di spedizione e informazioni di contatto) che hanno accettato di archiviare nel browser o nelle app di pagamento. Ciò implica che gli utenti hanno bisogno di:

- Un browser o altro agente utente che implementa l'API di richiesta di pagamento;
- Per i metodi di pagamento diversi dalla carta base, l'utente avrà bisogno di una o più app di pagamento;
- L'utente quindi memorizzerà le credenziali nel proprio browser o in altre app di pagamento

Quindi, quando si visita un sito Web che sfrutta l'API di richiesta di pagamento, gli utenti avranno l'opportunità di sfruttare il riutilizzo semplificato delle credenziali archiviate.

1.6.2 Ruolo del commerciante

Ad un livello elevato, l'API influisce sul front end (checkout), non sul back-end. Pertanto, il commerciante non dovrebbe dover apportare modifiche all'elaborazione back-end di vari metodi di pagamento. Chiunque fornisca la pagina di pagamento del venditore sostituirà i moduli Web con le chiamate all'API di richiesta di pagamento. Un commerciante puo accettare o meno un particolare metodo di pagamento in base al metodo di pagamento Ad esempio, se oggi il commerciante raccoglie i dati della carta di credito tramite un modulo Web, il commerciante (o il fornitore di servizi di pagamento) riceverà gli stessi dati

tramite l'API di richiesta di pagamento e dovrà comunque gestirlo, tuttavia ciò avviene oggi. Altri metodi di pagamento possono comportare l'integrazione di software aggiuntivo, accordi legali, verifica di commercianti, ecc. Ad esempio, un'organizzazione che possiede un metodo di pagamento proprietario potrebbe richiedere l'integrazione del software oltre a ciò che fa l'API di richiesta di pagamento.

È richiesto il gesto dell'utente per attivare l'API di richiesta di pagamento attraverso PaymentRequest.show (), il che significa che deve essere attivato PaymentRequest.show (), ad esempio, tramite un gestore di eventi del pulsante.

L'API non include l'autenticazione dell'utente Esistono (almeno) due tipi di autenticazione utente, ma entrambi esulano dall'ambito delle specifiche dell'API della richiesta di pagamento:

- Autenticazione del commerciante dell'utente (ad esempio, l'utente accede al proprio account con il commerciante).
- Autenticazione delle app di pagamento dell'utente (ad esempio, fornendo un codice CVV, tramite nome / password o tramite autenticazione a più fattori).

1.6.3 Ruolo fornitore app di pagamento

Ad un livello elevato, un'app di pagamento deve registrarsi con un browser per dichiarare quale metodo di pagamento supporta. Le app native di pagamento mobile registrano e comunicano con i browser attraverso meccanismi specifici della piattaforma che sono fuori portata per W3C. Il gruppo di lavoro Web Payments sta lavorando a un meccanismo per consentire ai siti Web di registrarsi per gestire i pagamenti per conto dell'utente; questo è un lavoro in corso.

1.6.4 Ruolo del browser

I browser svolgono due ruoli nell'API. Il primo riguarda queste attività:

- Calcola l'intersezione dei metodi di pagamento accettati dal commerciante e registrati dall'utente.
- Visualizza l'interfaccia utente (nativa) che consente all'utente di fornire informazioni di contatto, indirizzi di spedizione e corrispondenti app di pagamento.
- Fungere da canale per i dati da e verso il commerciante e da e verso l'utente (tramite app di pagamento).

È probabile che anche i browser svolgano un secondo ruolo, agendo come app di pagamento per un piccolo numero di metodi di pagamento. Al momento della stesura di questo documento, i produttori di browser prevedono di supportare i pagamenti della Carta base. Ciò significa che gli utenti saranno in grado di restituire rapidamente le informazioni della carta ai commercianti che sono stati precedentemente memorizzati nel browser.

1.6.5 Metodi di pagamento

L'API di richiesta di pagamento è progettata per funzionare con un gran numero di metodi di pagamento, inclusi i metodi pull (come i pagamenti con carta) e i metodi push (come i bonifici). I metodi di pagamento vengono identificti attraverso due strade:

- I metodi di pagamento definiti da W3C sono identificati da stringhe corte come "basic-card". Ogni specifica del metodo di pagamento di W3C include la stringa breve per l'identificazione di quel metodo di pagamento.
- I metodi di pagamento definiti da altre parti sono identificati dagli URL. Ciò consente a qualsiasi parte di pubblicare un metodo di pagamento a cui possono fare riferimento i commercianti.

Per creare un metodo di pagamento che può essere utilizzato con questa API, sono richieste le seguenti informazioni:

- Un identificatore per il metodo di pagamento; vedere la specifica dell'identificatore del metodo di pagamento .
- Una descrizione dei dati forniti dal commerciante all'app di pagamento ("dati di richiesta") e i dati restituiti dall'app di pagamento al commerciante tramite l'agente utente ("dati di risposta").

1.6.6 App di pagamento

In base alla progettazione,le app di pagamento sono app native per dispositivi mobili e app Web. Il gruppo di lavoro Web Payments sta lavorando su una specifica per come le applicazioni Web si registrano per gestire le richieste di pagamento. Il gruppo di lavoro Web Payments non discute l'integrazione di app di pagamento native su piattaforme proprietarie; che viene gestito dai proprietari di quelle piattaforme. Il gruppo di lavoro Web Payments sta sviluppando una specifica che definisce il modo in cui il browser possa sapere quali app di pagamento ha l'utente. L'obiettivo del gruppo di lavoro Web Payments è stabilire un meccanismo di registrazione unico per i programmi utente su qualsiasi sistema. Per le app di pagamento create con tecnologia nativa (proprietaria), il browser o il sistema operativo sottostante determinano il meccanismo di registrazione e possono variare da sistema a sistema. L'API di richiesta di pagamento, ovvero il browser, determina se un'app di pagamento "corrisponde" a una determinata transazione definendo un algoritmo che considera:

- Se l'app di pagamento supporta uno qualsiasi dei metodi di pagamento accettati dal commerciante. Il commerciante dichiara quali metodi di pagamento supporta attraverso un elenco di identificativi del metodo di pagamento passati attraverso l'API.
- Inoltre, alcuni metodi di pagamento consentono ai commercianti e alle app di pagamento di descrivere più dettagliatamente le condizioni in base alle quali accettano o supportano tali metodi di pagamento. Questa informazione di "capacità" viene anche utilizzata per determinare se un'app di pagamento corrisponde a una determinata transazione.

Al fine di proteggere la privacy degli utenti, i commercianti hanno accesso a informazioni molto limitate sull'ambiente dell'utente. L'API di richiesta di pagamento supporta un meccanismo di query limitato per consentire al commerciante di determinare se l'utente ha "qualcosa" che può essere utilizzato con l'API di richiesta di pagamento, ma che non fornisce informazioni su software specifico. Questa informazione consente ai commercianti di rilevare il supporto per l'API di richiesta di pagamento e quindi di mostrare una pagina di checkout che utilizza l'API, oppure di creare una pagina di fallback se l'utente non è pronto a pagare con qualche app di pagamento. I commercianti possono semplificare le pagine di pagamento in diversi modi, tra cui:

- Non avranno bisogno di usare la pagina di checkout per raccogliere la spedizione e altri dati comuni; il browser può fornirlo più rapidamente se l'utente lo ha già archiviato.
- Non avranno bisogno di chiedere agli utenti di scegliere un metodo di pagamento. Invece, possono fornire un singolo pulsante che consente al browser di visualizzare app di pagamento rilevanti.

In che modo l'API di richiesta di pagamento influisce sul flusso dei metodi di pagamento che già supporta? Oggi, il flusso per gli utenti di solito implica qualcosa del genere:

- Scansione di un elenco di metodi di pagamento accettati (non di cui l'utente si preoccupa)
- Scegline uno e continua
- Per i metodi di pagamento che prevedono il lancio di un'app o la visita a un sito Web, inviare l'utente a quell'app o sito, quindi inviarli di nuovo dopo che il pagamento è stato completato.

L'API di richiesta di pagamento consente un flusso migliorato:

- L'utente preme un pulsante di acquisto singolo (non è richiesta alcuna scansione)
- Il browser visualizza le app di pagamento dell'utente che possono essere utilizzate per la transazione. È probabile che i browser supportino le preferenze dell'utente in modo che, ad esempio, un'app di pagamento venga avviata automaticamente su un determinato sito Web. Ciò semplificherà ulteriormente il checkout.
- Per i metodi di pagamento che prevedono il lancio di un'app o la visita a un sito Web, inviare l'utente a quell'app o sito, quindi inviarli di nuovo dopo che il pagamento è stato completato. Il gruppo di lavoro Web Payments sta discutendo, tuttavia, come possiamo migliorare l'esperienza dell'utente creando un senso più forte di "essere ancora nel contesto commerciale" quando si utilizza un'app o un sito Web da pagare. Più lavoro deve essere fatto su questo argomento.

1.6.7 Differenze tra metodo di pagamento e app di pagamento

Nell'ecosistema dell'API della richiesta di pagamento, disaccoppiamo i dati necessari per effettuare un pagamento dal software utilizzato per raccogliere tali dati e avviare l'elaborazione dei pagamenti. Un metodo di pagamento è caratterizzato dai dati che il commerciante fornisce al pagatore e riceve dal pagatore per essere pagato.

- Esempio: per un metodo di pagamento con carta di base, il commerciante non fornisce dati al pagatore e riceve dal numero di carta del pagatore, dalla data di scadenza, ecc.
- Esempio: per un metodo di pagamento con trasferimento di credito, il commerciante fornisce informazioni sull'account commerciante, ecc. E riceve dalle informazioni del pagatore sulla data di elaborazione, i codici di transazione, ecc.

Un'app di pagamento è il software che l'utente utilizza per pagare. Un'app di pagamento può supportare uno o più metodi di pagamento. Le app di pagamento possono essere implementate utilizzando una varietà di tecnologie, incluse quelle di sistemi operativi nativi o tecnologie Web o di un ibrido. I browser possono anche fungere da app di pagamento, memorizzando le credenziali dell'utente come le informazioni sulla carta. L'API di richiesta di pagamento e l'API di pagamento non risolvono il funzionamento interno delle app di pagamento, ma solo il modo in cui comunicano con il browser. In generale, più app di pagamento possono implementare lo stesso metodo di pagamento. Tuttavia, vi sono casi importanti in cui è disponibile una sola app di pagamento autorizzata a supportare un metodo di pagamento (proprietario). Nei casi in cui più app di pagamento possono servire un determinato metodo di pagamento (ad es. "Carta base"), il commerciante non deve preoccuparsi dell'integrazione software. Il commerciante richiede " dati" attraverso l'API di richiesta di pagamento. Il commerciante (o il suo fornitore di servizi di pagamento) deve gestire tali dati sul back-end, ma non è necessaria alcuna integrazione software aggiuntiva. Allo stesso modo, il disaccoppiamento libera l'utente a scegliere la sua app di pagamento preferita.

2 Specifiche

2.1 Metodi

Per utilizzare l'API, lo sviluppatore deve fornire e tenere traccia di una serie di informazioni chiave. Questi bit di informazioni vengono passati al costruttore PaymentRequest come argomenti e successivamente utilizzati per aggiornare la richiesta di pagamento visualizzata all'utente. Vale a dire, questi bit di informazione sono:

- Il methodData: Una sequenza di PaymentMethodData che rappresenta i metodi di pagamento che il sito supporta;
- I details: i dettagli della transazione, come un PaymentDetailsInit. Ciò include il costo totale e facoltativamente un elenco di beni o servizi acquistati, beni materiali e opzioni di spedizione. Inoltre, può facoltativamente includere "modificatori" su come vengono effettuati i pagamenti. Ad esempio, "se paghi con una carta di credito di tipo X, incorre in una tassa di elaborazione di 3,00".
- Le options: un elenco di PaymentOptions, in quanto il sito ha bisogno di consegnare il bene o il servizio (ad esempio, per i beni fisici, il commerciante di solito avrà bisogno di un indirizzo fisico da spedire; per i beni digitali, di solito un'e-mail è sufficiente). Una volta che il PaymentRequest è stato costruito, viene presentato all'utente finale tramite il show()metodo. I show() ritornano una promessa che, una volta che l'utente conferma la richiesta di pagamento, si traduce in una PaymentResponse.
- payment request: uno sviluppatore crea una PaymentRequest per effettuare una richiesta di pagamento. Questo è in genere associato all'utente che avvia un processo di pagamento. La PaymentRequest consente agli sviluppatori di scambiare informazioni con l' user agent mentre l'utente sta fornendo dati in input. Poiché la visualizzazione simultanea di più interfacce PaymentRequest potrebbe confondere l'utente, questa specifica limita lo user agent a visualizzarne uno alla volta tramite il metodo show().

- payment methodData: un PaymentMethodData viene utilizzato per indicare un insieme di metodi di pagamento supportati e qualsiasi dato specifico del metodo di pagamento associato per tali metodi.
- payment detailsInit: Il PaymentDetailsInit viene utilizzato nella costruzione della richiesta di pagamento.
- payment options: Il PaymentOptions viene passato al costruttore PaymentRequest e fornisce informazioni sulle opzioni desiderate per la richiesta di pagamento.
- payment response: un PaymentResponse viene restituito quando un utente ha selezionato un metodo di pagamento e approvato una richiesta di pagamento.

2.1.1 MethodData

La sequenza methodData contiene PaymentMethodData contenenti gli identificativi del metodo di pagamento per i metodi di pagamento accettati dal sito Web e qualsiasi dato specifico del metodo di pagamento associato.

```
1     const methodData = [
2       {
3         supportedMethods: "basic-card",
4         data: {
5           supportedNetworks: ["visa", "mastercard"],
6           supportedTypes: ["debit", "credit"],
7         },
8       },
9       {
10        supportedMethods: "https://example.com/bobpay",
11        data: {
12          merchantIdentifier: "XXXX",
13          bobPaySpecificField: true,
14        },
15      },
16    ];
17
```

2.1.2 Details

I details contengono informazioni sulla transazione che l'utente è invitato a completare.

```
    const details = {
      id: "super-store-order-123-12312",
      displayItems: [
      {
        label: "Sub-total",
        amount: { currency: "USD", value: "55.00" },
      },
      {
        label: "Sales Tax",
        amount: { currency: "USD", value: "5.00" },
        type: "tax"
      },
      ],
      total: {
        label: "Total due",
        // The total is USD$65.00 here because we need to
        // add shipping (below). The selected shipping
        // costs USD$5.00.
        amount: { currency: "USD", value: "65.00" },
      },
    };

```

2.1.3 Opzioni di spedizione

Qui vediamo un esempio di come aggiungere due opzioni di spedizione ai details.

```
    const shippingOptions = [
      {
        id: "standard",
        label: "Ground Shipping (2 days)",
        amount: { currency: "USD", value: "5.00" },
        selected: true,
      },
      {
        id: "drone",
        label: " Drone Express (2 hours)",
        amount: { currency: "USD", value: "25.00" }
      },
    ];
    Object.assign(details, { shippingOptions });

```

2.1.4 Modifiche condizionali alla richiesta di pagamento

Qui vediamo come aggiungere una tassa di elaborazione per l'utilizzo di una carta di credito. Si noti che richiede il ricalcolo del totale.

```
    // Credit card incurs a $3.00 processing fee.
    const creditCardFee = {
      label: "Credit card processing fee",
      amount: { currency: "USD", value: "3.00" },
    };

    // Modifiers apply when the user chooses to pay with
    // a credit card.
    const modifiers = [
      {
        additionalDisplayItems: [creditCardFee],
        supportedMethods: "basic-card",
        total:
        {
          label: "Total due",
          amount: { currency: "USD", value: "68.00" },
        },
        data:
        {
          supportedTypes: "credit",
        },
      },
    ];
    Object.assign(details, { modifiers });

```

2.1.5 Options

Options contiene informazioni che lo sviluppatore ha bisogno dall'utente per eseguire il pagamento.

```
    const options = {
      requestPayerEmail: false,
      requestPayerName: true,
      requestPayerPhone: false,
      requestShipping: true,
    }

```

2.1.6 PaymentRequest

Dopo aver raccolto tutti i bit di informazioni prerequisite, ora possiamo costruirne uno PaymentRequest e richiedere che il browser lo presenti all'utente

```
async function doPaymentRequest()
{
  try
  {
    const request = new PaymentRequest(methodData, details, options);
    // See below for a detailed example of handling these events
    request.onshippingaddresschange = ev => ev.updateWith(details);
    request.onshippingoptionchange = ev => ev.updateWith(details);
    const response = await request.show();
    await validateResponse(response);
  } catch (err)
    {
      // AbortError, SecurityError
      console.error(err);
    }
}
async function validateResponse(response)
{
  try {
    if (await checkAllValuesAreGood(response))
    {
      await response.complete("success");
    }
    else {
      await response.complete("fail");
    }
  } catch (err)
    {
      // Something went wrong...
      await response.complete("fail");
    }
}
doPaymentRequest();

```

2.1.7 Gestione degli eventi e aggiornamento della richiesta di pagamento

Prima che l'utente accetti di effettuare il pagamento, al sito viene data l'opportunità di aggiornare la richiesta di pagamento in risposta all'input dell'utente. Ciò può includere, ad esempio, la fornitura di ulteriori opzioni di spedizione (o la modifica dei costi), la rimozione di articoli che non possono essere spediti a un determinato indirizzo, ecc.

```
    const request = new PaymentRequest(methodData, details, options);
    // Async update to details
    request.onshippingaddresschange = ev => {
      ev.updateWith(checkShipping(request));
    };
    // Sync update to the total
    request.onshippingoptionchange = ev => {
      const shippingOption = shippingOptions.find(
      option => option.id === request.id
    );
    const newTotal = {
      currency: "USD",
      label: "Total due",
      value: calculateNewTotal(shippingOption),
    };
    ev.updateWith({ ...details, total: newTotal });
    };
    async function checkShipping(request) {
      try {
        const json = request.shippingAddress.toJSON();
        await ensureCanShipTo(json);
        const { shippingOptions, total } = await calculateShipping(json);
        return { ...details, shippingOptions, total };
      } catch (err) {
        return { ...details, error: 'Sorry! we can't ship to your address.' };
      }
    }

```

2.1.8 Segnalazione di errori a grana fine

Uno sviluppatore può utilizzare il shippingAddressErrors, membro del PaymentDetailsUpdate, per indicare che ci sono errori di convalida con attributi specifici di a PaymentAddress. Il shippingAddressErrors è un AddressErrorFields, i cui membri delimitano in modo specifico i campi di un indirizzo fisico errati e forniscono anche messaggi di errore utili per la visualizzazione all'utente finale.

```
    request.onshippingaddresschange = ev => {
      ev.updateWith(validateAddress(request.shippingAddress));
    };
    function validateAddress(shippingAddress) {
      const error = "Can't ship to this address.";
      const shippingAddressErrors = {
        cityError: "FarmVille is not a real place.",
        postalCodeError: "Unknown postal code for your country.",
      };
      // Empty shippingOptions implies that we can't ship
      // to this address.
      const shippingOptions = [];
      return { error, shippingAddressErrors, shippingOptions };
    }

```

2.1.9 Postare la risposta di pagamento su un server

È previsto che i dati in a PaymentResponse verranno reindirizzati a un server per l'elaborazione. Per rendere questo più semplice possibile, PaymentResponse fornisce un toJSON()metodo che serializza l'oggetto direttamente in JSON. Ciò rende banale il POST del JSON risultante su un server che utilizza l' API Fetch.

```
    async function doPaymentRequest() {
      const payRequest = new PaymentRequest(methodData, details, options);
      const payResponse = await payRequest.show();
      let result = "";
      try {
        const httpResponse = await fetch("/process-payment", {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: payResponse.toJSON(),
        });
        result = httpResponse.ok ? "success" : "fail";
      } catch (err) {
          console.error(err);
          result = "fail";
        }
      await payResponse.complete(result);
    }
    doPaymentRequest();

```

3 Codice d'esempio per pagina demo

Implementazione PaymentRequest API su una pagina d'esempio.

3.0.1 Costruttore

Iniziamo con il costruttore. L'oggetto PaymentRequest è costruito passando i seguenti parametri:

- methodData: una serie di dizionari che contiene gli identificativi del metodo di pagamento e tutti i dati pertinenti; un identificativo del metodo di pagamento è una stringa che identifica un metodo di pagamento supportato
- dettagli: informazioni sulla transazione, come gli elementi pubblicitari in un ordine
- opzioni: informazioni aggiuntive che il Wallet potrebbe dover raccogliere

Nel seguente snippet, stiamo consentendo agli utenti di pagare con qualsiasi carta di debito o di credito appartenente alle reti Visa, MasterCard o Amex. L'oggetto dettagli contiene l'importo totale parziale, l'imposta sulle vendite e il totale dovuto. Questi dettagli verranno mostrati all'utente nel portafoglio. Bisogna tenere presente che l'API non aggiunge elementi o calcola l'imposta sulle vendite: spetta al commerciante fornire le informazioni corrette. In questo esempio, stiamo vendendo un bene fisico, quindi chiediamo al portafoglio di ritirare l'indirizzo di spedizione del cliente.

```
var methodData = [
  {
    supportedMethods: ['basic-card'],
    data: {
      supportedNetworks: ['visa', 'mastercard', 'amex'],
      supportedTypes: ['credit']
    }
  }
];

var details =  {
  displayItems: [
    {
      label: "Sub-total",
      amount: { currency: "USD", value : "100.00" } // US$100.00
    },
    {
      label: "Sales Tax",
      amount: { currency: "USD", value : "9.00" } // US$9.00
    }
  ],
  total:  {
    label: "Total due",
    amount: { currency: "USD", value : "109.00" } // US$109.00
  }
};

var options = {
  requestShipping: true
};

var payment = new PaymentRequest(methodData, details, options);

```

3.0.2 Visualizzazione dell'interfaccia utente, elaborazione del pagamento e visualizzazione dei risultati

Una volta creato l'oggetto PaymentRequest, è possibile attivare il browser per visualizzare il wallet con request.show ().

I clienti possono quindi selezionare le informazioni di pagamento, l'indirizzo di spedizione e altri campi appropriati, quindi fare clic su Paga quando è pronto. A questo punto, gli utenti dovranno verificare la loro identità. In caso di esito positivo, ciò soddisferà la promessa request.show () e restituirà al sito Web tutte le informazioni che il cliente ha fornito al Portafoglio. Per il metodo di pagamento con carta di base, l'oggetto risultato conterrà il nome del titolare della carta, il numero della carta, il mese di scadenza e altri campi pertinenti. Il commerciante può quindi utilizzare queste informazioni per elaborare la transazione sul back-end. Dopo che la risposta è tornata dal server, è possibile utilizzare result.complete ('successo') per visualizzare la schermata di successo nel Portafoglio e result.complete ('fail') per indicare una transazione fallita.

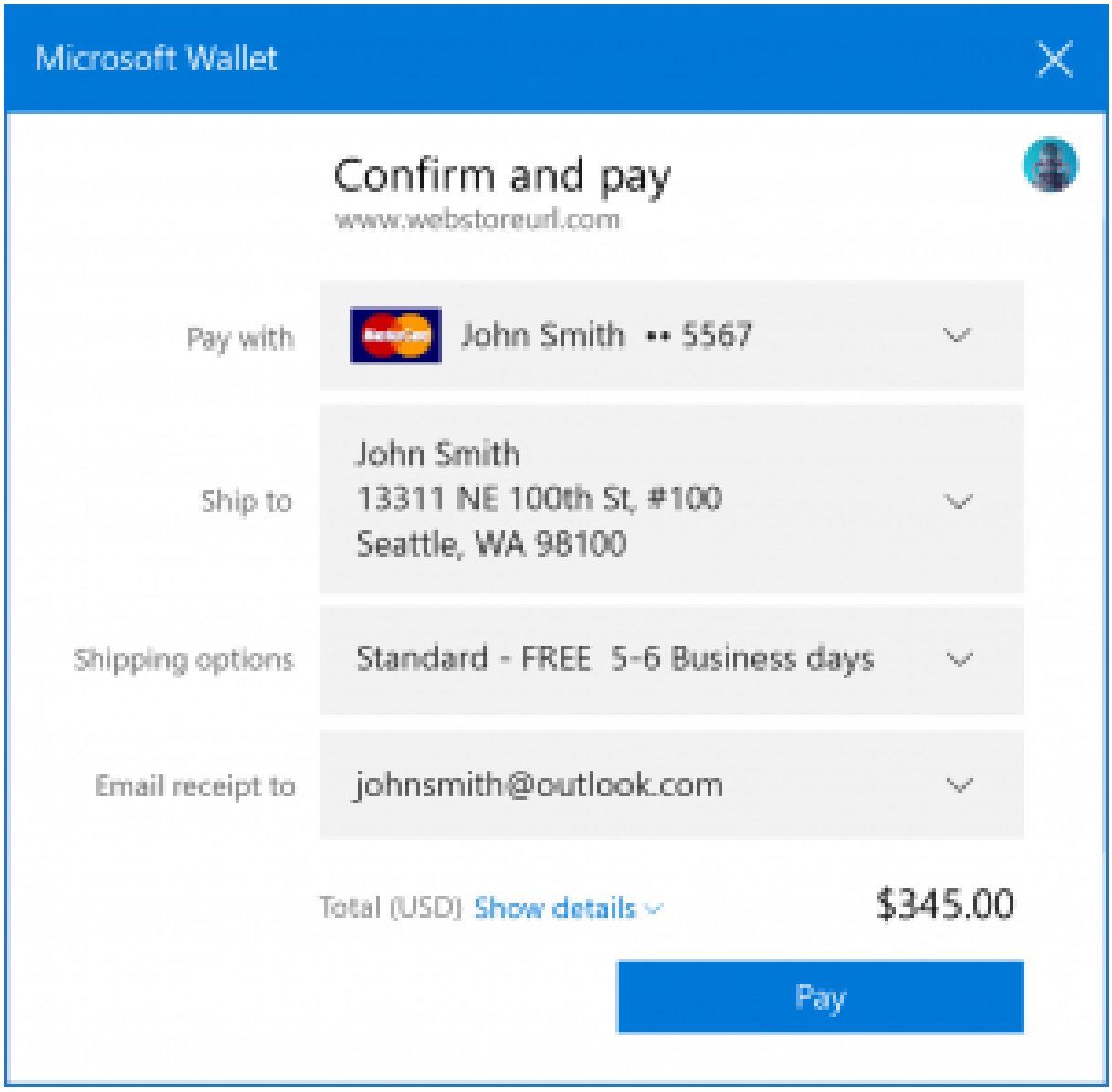

Figura 2: Wallet dopo la chiamata request.show()

```
  // Mostra l'interfaccia utente nativa
  payment.show()
  // Quando la promessa e soddisfatta, passa i risultati al tuo server per l'elaborazione
  .then(result => {
    return process(result).then(response => {
      if (response.status === 200) {
        // Mostra che la transazione ha avuto successo nell'interfaccia utente
        return result.complete('success');
      } else {
          // Mostra nell'interfaccia utente nativa che la transazione ha avuto esito negativo
          return result.complete('fail');
        }
      }).catch((err) => {
      console.error('User rejected request', err.message)
    });
  });

```

3.0.3 Ascoltando gli eventi

Il prezzo potrebbe cambiare in base all'indirizzo di spedizione e alle opzioni di spedizione selezionate dal cliente. È possibile ascoltare tali modifiche con gli eventi shippingaddresschange e shippingoptionchange per ricalcolare di conseguenza i prezzi.

```
    payment.addEventListener("shippingaddresschange", function (changeEvent) {
          // Elabora la modifica dell'indirizzo di spedizione
    });

    payment.addEventListener("shippingoptionchange", function (changeEvent) {
      // Modifica delle opzioni di spedizione del processo (ad esempio "spedizione in giornata")
    });

```

3.0.4 Rilevamento di funzionalità

I siti possono essere rilevati per l'API di richiesta di pagamento, inoltrare l'utente a un'esperienza legacy, basata su moduli, se non è disponibile.

```
      // Inoltra utente alla verifica basata su modulo se PaymentRequest non e disponibile
      if (!window.PaymentRequest) {
        window.location.href = '/form-based-checkout';
        return;
      }

```

Ecco un esempio di implementazione minima di questo codice:

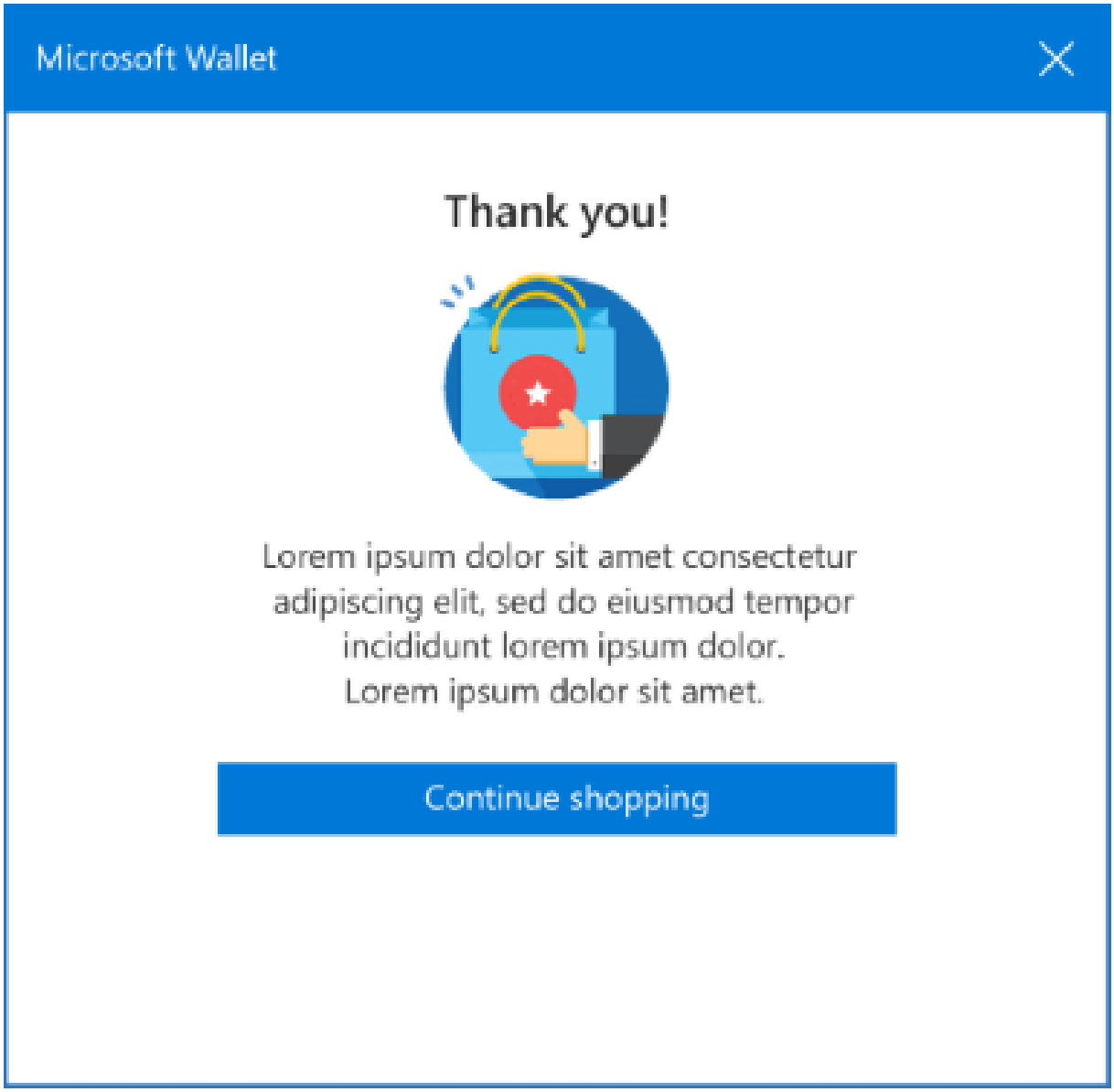

Figura 3: Wallet in caso di successo

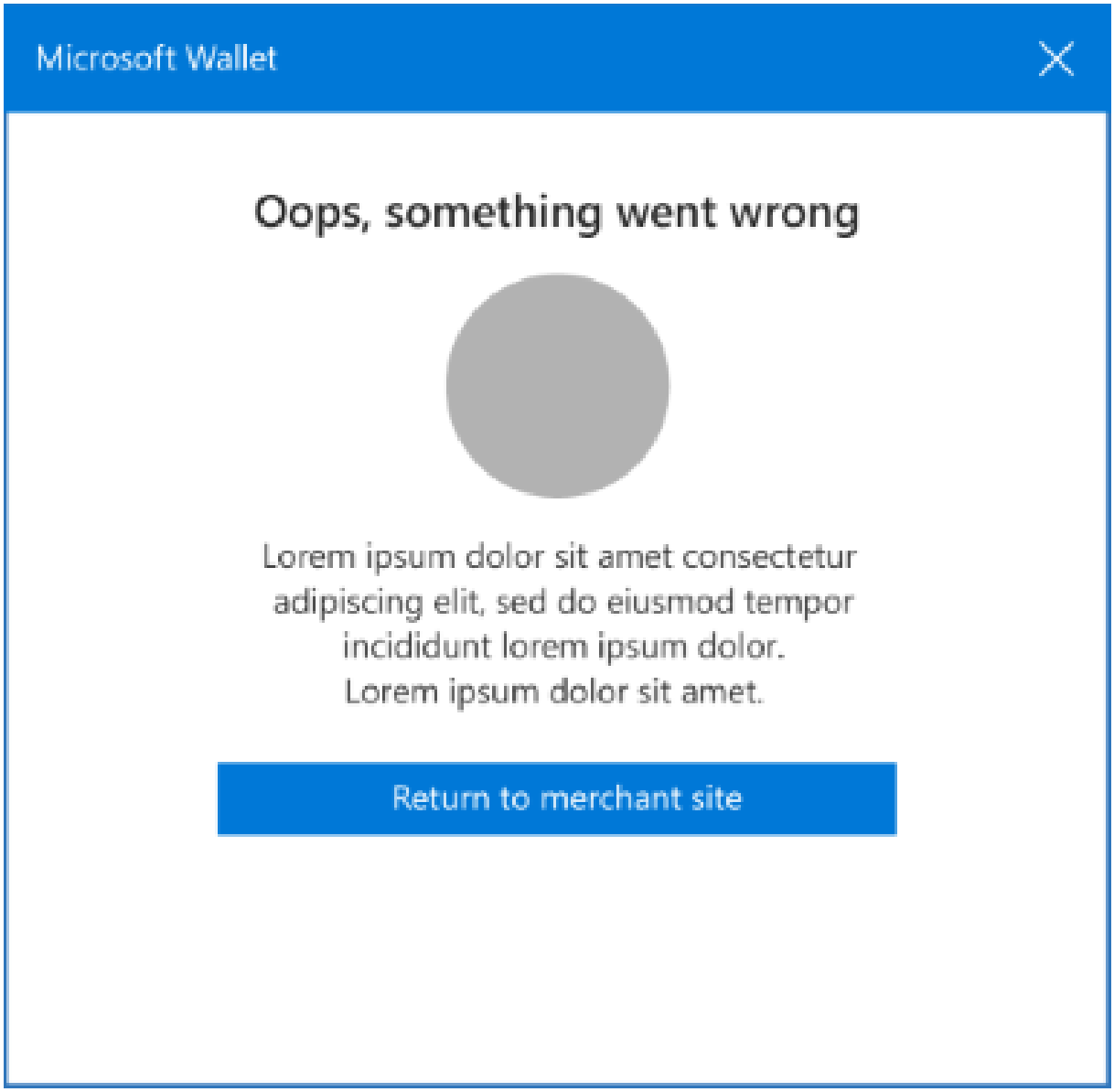

Figura 4: Wallet in caso di fail

```
function startPaymentRequestCheckout() {
  var methodData = [
    {
      supportedMethods: ['basic-card'],
      data: {
        supportedNetworks: ['visa', 'mastercard', 'amex'],
        supportedTypes: ['credit']
      }
    }
  ];

  var details =  {
      displayItems: [
        {
          label: "Sub-total",
          amount: { currency: "USD", value : "100.00" }, // US$100.00
        },
        {
          label: "Sales Tax",
          amount: { currency: "USD", value : "9.00" }, // US$9.00
        }
      ],
      total:  {
        label: "Total due",
        amount: { currency: "USD", value : "109.00" }, // US$109.00
      }
  };

  var options = {
    requestShipping: true
  };

  var request = new PaymentRequest(methodData, details, options);

  //Show the Native UI
  request.show()
  //When the promise is fulfilled, pass the results to your server for processing
  .then(result => {
    return process(result).then(response => {
      if (response.status === 200) {
        //Show that the transaction was successful in the UI
        return result.complete('success');
      } else {
        //Show in the Native UI that the transaction failed
        return result.complete('fail');
      }
    })
  });

  request.addEventListener("shippingaddresschange", function (changeEvent) {
    // Process shipping address change
  });

  request.addEventListener("onshippingoptionchange", function (changeEvent) {
    // Process shipping option change (e.g. "one day shipping")
  });
}
document.getElementById('#checkout').addEventListener('click',
  startPaymentRequestCheckout)

```

4 Compatibilità web

Desktop Mobile

Feature	Chrome	Edge	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	61	(Yes)	No support[1]	?	No support	?

Figura 5: Compatibilità desktop

Desktop **Mobile**

Feature	Android Webview	Chrome for Android	Edge	Firefox Mobile (Gecko)	IE Mobile	Opera Mobile	Safari Mobile
Basic support	No support	51	(Yes)	No support[1]	?	No support	?

Figura 6: Compatibilità mobile

5 Conclusioni

L'API di richiesta di pagamento fornisce un potente strumento per i commercianti per migliorare la conversione del checkout sul Web e per offrire ai clienti un'esperienza di acquisto più piacevole e conveniente. Questa API è un ottimo esempio della potenza e della flessibilità della piattaforma web, ed è sulla strada per un'ampia interoperabilità, con Chrome per Android che supporta l'API a partire da Chrome 54.