

# Tesi - Payment Request API

Daniele Rigon - 857319

17 settembre 2018

# Indice

<b>1</b>	<b>Overview Payment Request</b>	<b>3</b>
1.1	Vantaggi . . . . .	3
1.2	Come funziona . . . . .	3
1.3	Uso API . . . . .	4
1.3.1	Ruolo dell'utente . . . . .	4
1.3.2	Ruolo del commerciante . . . . .	4
1.3.3	Ruolo del browser . . . . .	4
1.3.4	Metodi di pagamento . . . . .	5
1.3.5	App di pagamento . . . . .	5
1.3.6	Differenze tra metodo di pagamento e app di pagamento . . . . .	5
<b>2</b>	<b>Specifiche</b>	<b>6</b>
2.1	Metodi . . . . .	6
2.1.1	PaymentMethodData . . . . .	7
2.1.2	PaymentDetails . . . . .	7
2.1.3	PaymentOptions . . . . .	9
2.1.4	PaymentRequest . . . . .	9
<b>3</b>	<b>Rischi e sicurezza</b>	<b>10</b>
3.1	Esempio di un possibile attacco . . . . .	10
3.1.1	Descrizione dell'attacco . . . . .	10
3.1.2	Come difendersi . . . . .	12
<b>4</b>	<b>Implementazione PaymentRequest API su una pagina d'esempio</b>	<b>13</b>
4.1	Costruttore . . . . .	13
4.2	Visualizzazione dell'interfaccia utente, elaborazione del pagamento e visualizzazione dei risultati . . . . .	14
4.3	Ascoltare gli eventi . . . . .	16
<b>5</b>	<b>Compatibilità web</b>	<b>17</b>
<b>6</b>	<b>Conclusioni</b>	<b>18</b>

# 1 Overview Payment Request

La PaymentRequest API nasce con l'intento di creare esperienze di pagamento semplificate, in quanto ogni sito web ha il proprio sistema di pagamento e molti siti richiedono la ridigitazione manuale delle stesse informazioni più volte, le quali possono essere invece memorizzate e riutilizzate dall'API per completare più rapidamente le transazioni online.

## 1.1 Vantaggi

- **Esperienza di acquisto rapida:** gli utenti immettono i propri dati una sola volta nel browser, e dopo averli inseriti non è più necessario reinserirli su siti diversi;
- **Esperienza coerente su ogni sito che supporta l'API:** poiché la pagina di pagamento è controllata dal browser si può personalizzare l'esperienza utente, ad esempio includendo la localizzazione per impostare automaticamente la lingua preferita dell'utente o altre features;
- **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, 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 sia la carta predefinita/preferita;
- **Esperienza utente migliorata:** meno tipizzazione, coerenza tra i siti Web, tra browser e sistemi operativi e nuove funzionalità del browser per semplificare il checkout, ecc;
- **Miglioramento della sicurezza:** la PaymentRequest API ha il potenziale per ridurre le opportunità di frode e può facilitare l'adozione di metodi di pagamento più sicuri. Purtroppo ci sono dei problemi di sicurezza analizzati al capitolo 4;
- **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 nei confronti del cliente.

## 1.2 Come funziona

La PaymentRequest API 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, e può inoltre richiedere al browser di creare un'interfaccia utente semplificata per raccogliere l'indirizzo di spedizione, le informazioni di contatto e altri elementi all'utente. Il browser determina quali metodi di pagamento sono supportati dal commerciante tra le varie "app di pagamento" mostrandole all'utente. L'utente seleziona un'app di pagamento con la quale pagare, la quale 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.

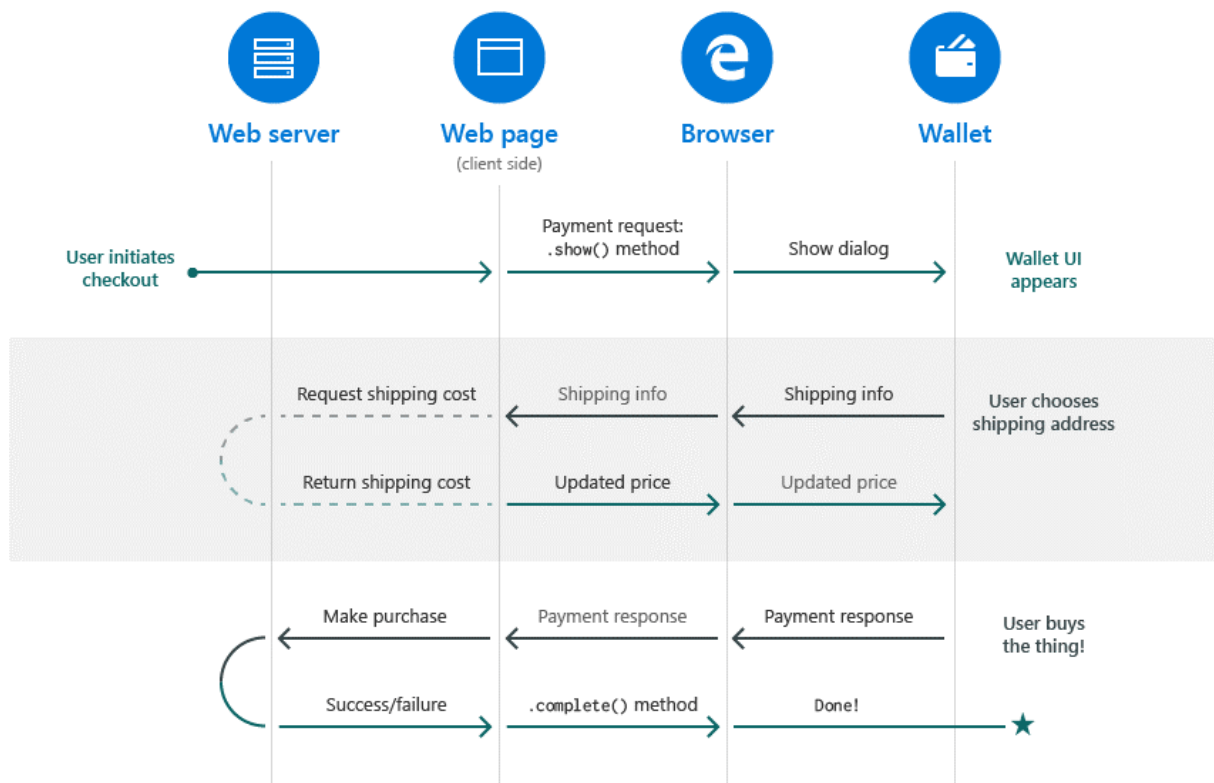


Figura 1: Schema Payment Request API

## 1.3 Uso API

### 1.3.1 Ruolo dell'utente

Gli utenti beneficiano del riutilizzo delle informazioni inserite nel browser o nelle app di pagamento. Quindi, quando si visita un sito Web che sfrutta la PaymentRequest API gli utenti avranno l'opportunità di sfruttare il riutilizzo semplificato delle informazioni archiviate.

### 1.3.2 Ruolo del commerciante

L'API influisce sul front end (l'interfaccia dell'esperienza utente) e non sul back-end, pertanto il commerciante non dovrebbe dover apportare modifiche all'elaborazione back-end dei vari metodi di pagamento; questo sarà compito del fornitore della pagina di pagamento il quale sostituirà i moduli Web con le chiamate alla PaymentRequest API.

### 1.3.3 Ruolo del browser

Il browser svolge diversi ruoli:

- Calcola l'intersezione dei metodi di pagamento accettati dal commerciante e registrati dall'utente;
- Visualizza l'interfaccia utente che consente all'utente di inserire le proprie informazioni;
- Funge da canale per i dati da e verso il commerciante e da e verso l'utente.

### 1.3.4 Metodi di pagamento

La PaymentRequest API è progettata per funzionare con un gran numero di metodi di pagamento, i quali vengono identificati attraverso due strade:

- I metodi di pagamento definiti da W3C sono identificati come "basic-card" e sono composti da stringhe corte;
- I metodi di pagamento definiti da altre parti sono identificati dagli URL.

### 1.3.5 App di pagamento

La PaymentRequest API determina se un'app di pagamento "corrisponde" a una determinata transazione definendo un algoritmo che considera i metodi di pagamento accettati dal commerciante, dichiarati attraverso un elenco di identificativi del metodo di pagamento, passati attraverso l'API. Al fine di proteggere la privacy degli utenti i commercianti hanno accesso a informazioni molto limitate dell'utente.

Vediamo in che modo la PaymentRequest API influisce sul flusso dei metodi di pagamento che già supporta. Il normale flusso per gli utenti di solito implica qualcosa del genere:

- Scansione di un elenco di metodi di pagamento accettati;
- Scelta di un metodo;
- Per i metodi di pagamento che prevedono il lancio di un'app o la visita a un sito Web si invia l'utente a quell'app o sito;
- Pagamento completato.

La PaymentRequest API consente un flusso migliorato:

- L'utente preme un pulsante di acquisto singolo;
- Il browser visualizza le app di pagamento dell'utente che possono essere utilizzate per la transazione, ed è probabile che i browser supportino le preferenze dell'utente in modo che un'app di pagamento venga avviata automaticamente su un determinato sito Web, semplificando 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;
- Pagamento completato;

### 1.3.6 Differenze tra metodo di pagamento e app di pagamento

Un'app di pagamento è il software che l'utente utilizza per pagare, la quale può supportare uno o più metodi di pagamento e può essere implementata utilizzando diverse tecnologie. I browser possono anche fungere da app di pagamento, memorizzando le credenziali dell'utente. In generale più app di pagamento possono implementare lo stesso metodo di pagamento. Vi sono casi importanti in cui è disponibile una sola app di pagamento autorizzata a supportare un metodo di pagamento, mentre ci sono casi in cui più app di pagamento possono servire diversi metodi di pagamento. In questo caso non è il commerciante che deve preoccuparsi dell'integrazione software, ma deve solamente richiedere le informazioni attraverso la PaymentRequest API.

## 2 Specifiche

### 2.1 Metodi

Per utilizzare l'API lo sviluppatore deve fornire e tenere traccia di una serie di informazioni chiave, le quali vengono passate al costruttore `PaymentRequest` come argomenti e successivamente utilizzate per aggiornare la richiesta di pagamento visualizzata all'utente. Queste informazioni sono:

- **PaymentMethodData:** rappresenta i metodi di pagamento che il sito supporta;
- **PaymentDetails:** rappresenta i dettagli della transazione. Ciò include il costo totale e facoltativamente un elenco di beni o servizi acquistati, beni materiali, opzioni di spedizione o "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 tot";
- **PaymentOptions:** il `PaymentOptions` viene passato al costruttore `PaymentRequest` e fornisce informazioni sulla consegna del prodotto: ad esempio per i beni fisici il commerciante avrà bisogno di un indirizzo fisico dove spedire, mentre per i beni digitali è sufficiente un'e-mail. Una volta che il `PaymentRequest` è stato costruito viene presentato all'utente finale tramite il metodo `show()`, il quale ritorna una promise che, una volta che l'utente conferma la richiesta di pagamento, si traduce in una `PaymentResponse`;
- **PaymentRequest:** la `PaymentRequest` serve a effettuare una richiesta di pagamento, in genere associata all'avvio di un processo di pagamento da parte dell'utente. 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()`;
- **PaymentDetailsInit:** Il `PaymentDetailsInit` viene utilizzato nella costruzione della richiesta di pagamento;
- **PaymentResponse:** un `PaymentResponse` viene restituito quando un utente ha selezionato un metodo di pagamento e approvato una richiesta di pagamento.

### 2.1.1 PaymentMethodData

PaymentMethodData contiene gli identificativi dei 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 ];
```

### 2.1.2 PaymentDetails

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

```
1 const details = {  
2   id: "super-store-order-123-12312",  
3   displayItems: [  
4     {  
5       label: "Sub-total",  
6       amount: { currency: "USD", value: "55.00" },  
7     },  
8     {  
9       label: "Sales Tax",  
10      amount: { currency: "USD", value: "5.00" },  
11      type: "tax"  
12    },  
13  ],  
14  total: {  
15    label: "Total due",  
16    /* The total is USD$65.00 here because we need to add shipping (  
17    below). The selected shipping costs USD $5.00.*/  
18    amount: { currency: "USD", value: "65.00" },  
19  },  
20 };
```

## Opzioni di spedizione

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

```
1 const shippingOptions = [  
2   {  
3     id: "standard",  
4     label: "Ground Shipping (2 days)",  
5     amount: { currency: "USD", value: "5.00" },  
6     selected: true,  
7   },  
8   {  
9     id: "drone",  
10    label: " Drone Express (2 hours)",  
11    amount: { currency: "USD", value: "25.00" }  
12  },  
13 ];  
14 Object.assign(details, { shippingOptions });
```

## 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.

```
1 // Credit card incurs a $3.00 processing fee.  
2 const creditCardFee = {  
3   label: "Credit card processing fee",  
4   amount: { currency: "USD", value: "3.00" },  
5 };  
6 // Modifiers apply when the user chooses to pay with a credit card.  
7 const modifiers = [  
8   {  
9     additionalDisplayItems: [creditCardFee],  
10    supportedMethods: "basic-card",  
11    total:  
12    {  
13      label: "Total due",  
14      amount: { currency: "USD", value: "68.00" },  
15    },  
16    data:  
17    {  
18      supportedTypes: "credit",  
19    },  
20  },  
21 ];  
22 Object.assign(details, { modifiers });
```



### 2.1.3 PaymentOptions

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

```
1 const options = {
2   requestPayerEmail: false,
3   requestPayerName: true,
4   requestPayerPhone: false,
5   requestShipping: true,
6 }
```

### 2.1.4 PaymentRequest

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

```
1 async function doPaymentRequest() {
2   try{
3     const request = new PaymentRequest(methodData, details, options)
4     ;
5     // See below for a detailed example of handling these events
6     request.onshippingaddresschange = ev => ev.updateWith(details);
7     request.onshippingoptionchange = ev => ev.updateWith(details);
8     const response = await request.show();
9     await validateResponse(response);
10    } catch (err){
11      // AbortError, SecurityError
12      console.error(err);
13    }
14  }
15  async function validateResponse(response){
16    try {
17      if (await checkAllValuesAreGood(response)){
18        await response.complete("success");
19      }
20      else{
21        await response.complete("fail");
22      }
23    } catch (err){
24      // Something went wrong
25      await response.complete("fail");
26    }
27  }
28  doPaymentRequest();
```

## 3 Rischi e sicurezza

La PaymentRequest API aumenta la sicurezza poichè:

- I commercianti possono ottenere un checkout semplificato senza memorizzare le informazioni dell'utente, in quanto lo fa l'API;
- La PaymentRequest API 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 attraverso una firma digitale.

### 3.1 Esempio di un possibile attacco

#### 3.1.1 Descrizione dell'attacco

L'API apre una finestra dove inserire le informazioni, le quali però possono essere intercettate come verrà mostrato nell'esempio di seguito. L'API salva le informazioni dell'utente ma viene chiesto ogni volta di inserire il codice CVV come forma di sicurezza, quindi per l'attaccante sarà possibile in ogni sessione rubare le informazioni inserite dall'utente. Alla richiesta di avvio di un processo di pagamento viene creato l'oggetto PaymentRequest nel quale vengono inseriti tutti i dati che l'utente inserisce. Una volta che l'utente ha approvato una richiesta di pagamento viene restituito un PaymentResponse per approvare tale richiesta. L'oggetto PaymentRequest, creato in precedenza e contenente i dati dell'utente, passa però attraverso il DOM, e quindi può essere intercettato. Ad esempio nel campo details.cardNumber si può leggere il numero di carta di credito, mentre in details.cardSecurityCode si può leggere il codice CVV; entrambi sono mostrati nell'esempio in Figura 4 (CAMBIARE SE CAMBIA LA FIGURA)

Verifica il pagamento

Riepilogo dell'ordine	Original donation amount	65,00	
	Friends and family discount	-10,00	▶
	Donation	USD 55,00	

---

Pagamento	Visa ****7782 aaa	VISA ▶
-----------	----------------------	--------

Puoi gestire carte e indirizzi nelle [Impostazioni](#).

chrome Paga Annulla

Figura 2: Inserimento informazioni dell'utente

← Inserisci il codice CVC della carta Visa ••••7782

Dopo essere stati confermati, i dati della carta saranno condivisi con questo sito.




Figura 3: Inserimento CVV

Dopo l'elaborazione i numeri di carta di credito e CVV saranno visibili all'attaccante.

#### Live Output

```
{
  "methodName": "basic-card",
  "details": {
    "billingAddress": {
      "addressLine": [
        "aaa"
      ],
      "city": "aaa",
      "country": "IT",
      "dependentLocality": "",
      "languageCode": "it",
      "organization": "aaa",
      "phone": "3333333",
      "postalCode": "aaa",
      "recipient": "aaa",
      "region": "BO",
      "sortingCode": ""
    },
    "cardNumber": "4916629305287782",
    "cardSecurityCode": "111",
    "cardholderName": "aaa",
    "expiryMonth": "06",
    "expiryYear": "2021"
  }
}
```

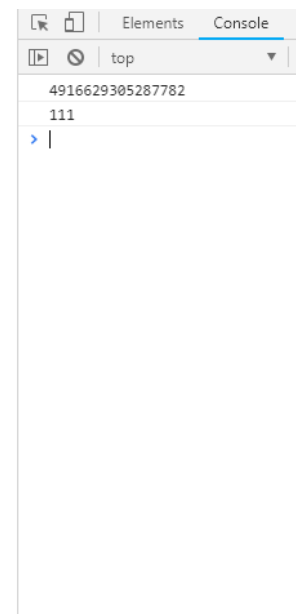


Figura 4: Informazioni rubate

### **3.1.2 Come difendersi**

– E COSI E NON SI PUO FAR NIENTE? –

## 4 Implementazione PaymentRequest API su una pagina d'esempio

### 4.1 Costruttore

L'oggetto PaymentRequest è costruito passando i seguenti parametri:

- **methodData:** una serie di 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;
- **details:** contiene le informazioni sulla transazione, come gli elementi pubblicitari in un ordine;
- **options:** contiene informazioni aggiuntive che il Wallet potrebbe dover raccogliere.

Nel seguente esempio stiamo consentendo agli utenti di pagare con qualsiasi carta di debito o di credito appartenente alle reti Visa, MasterCard o Amex. L'oggetto details 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 l'indirizzo di spedizione del cliente.

```
1 var methodData = [  
2   {  
3     supportedMethods: ['basic-card'],  
4     data: {  
5       supportedNetworks: ['visa', 'mastercard', 'amex'],  
6       supportedTypes: ['credit']  
7     }  
8   }  
9 ];  
10 var details = {  
11   displayItems: [  
12     {  
13       label: "Sub-total",  
14       amount: { currency: "USD", value : "100.00" } // US$100.00  
15     },  
16     {  
17       label: "Sales Tax",  
18       amount: { currency: "USD", value : "9.00" } // US$9.00  
19     }  
20   ],  
21   total: {  
22     label: "Total due",  
23     amount: { currency: "USD", value : "109.00" } // US$109.00  
24   }  
25 };  
26 var options = {  
27   requestShipping: true  
28 };  
29 var payment = new PaymentRequest(methodData, details, options);
```

## 4.2 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()`.

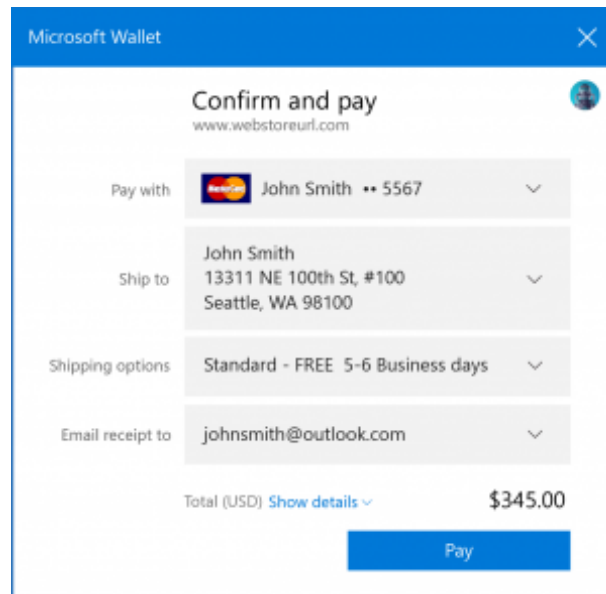


Figura 5: Wallet dopo la chiamata `request.show()`

I clienti possono selezionare le informazioni di pagamento, l'indirizzo di spedizione e altri campi appropriati e cliccare su Paga quando è pronto. A questo punto, gli utenti dovranno verificare la loro identità: in caso di esito positivo verrà soddisfatta la promise `request.show()` e verranno restituite al sito Web tutte le informazioni che il cliente ha fornito. Per il metodo di pagamento con carta di base l'oggetto risultante 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('success')` per visualizzare la schermata di successo o `result.complete('fail')` per indicare una transazione fallita.

```
1 //Quando la promessa è soddisfatta, passa i risultati al server per l'
  elaborazione
2 payment.show().then(result => {
3   return process(result).then(response => {
4     if (response.status === 200) {
5       //La transazione ha avuto successo
6       return result.complete('success');
7     } else {
8       //la transazione ha fallito
9       return result.complete('fail');
10    }
11  }).catch((err) => {
12    console.error('User rejected request', err.message)
13  });
14 });
```

Ed ecco i wallet in caso di successo e di fallimento.

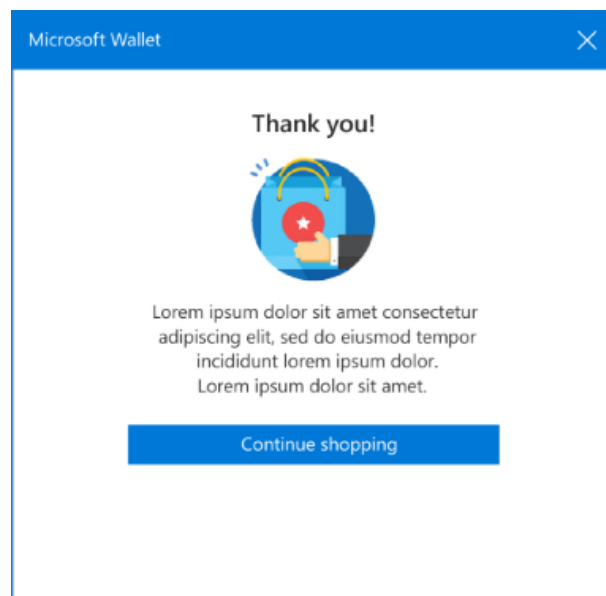


Figura 6: Wallet in caso di successo

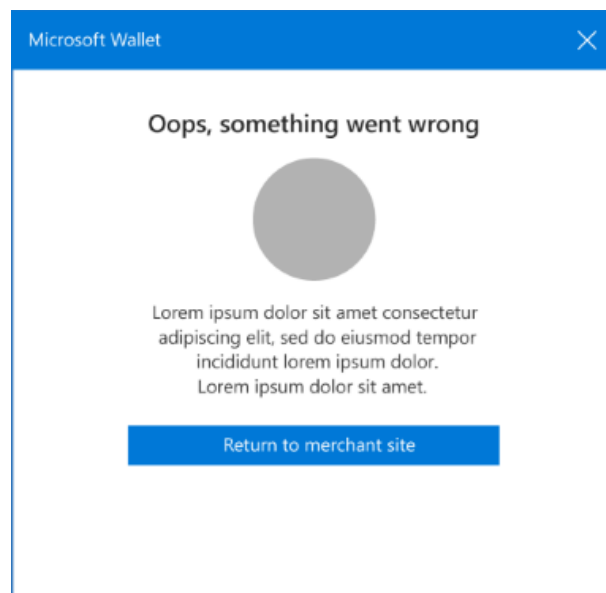


Figura 7: Wallet in caso di fail

### 4.3 Ascoltare 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.

```
1 payment.addEventListener("shippingaddresschange",function (changeEvent){
2     // Elabora la modifica dell'indirizzo di spedizione
3 });
4 payment.addEventListener("shippingoptionchange",function (changeEvent){
5     // Modifica delle opzioni di spedizione del processo
6 });
```



## 5 Compatibilità web

Desktop		Mobile				
Feature	Chrome	Edge	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	61	(Yes)	No support <sup>[1]</sup>	?	No support	?

Figura 8: 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 <sup>[1]</sup>	?	No support	?

Figura 9: Compatibilità mobile

## 6 Conclusioni

La PaymentRequest API è uno strumento per migliorare l'esperienza utente sul Web offrendo ai clienti un'esperienza di acquisto più piacevole, pur avendo delle vulnerabilità.

## Elenco delle figure

1	Schema Payment Request API . . . . .	4
2	Inserimento informazioni dell'utente . . . . .	10
3	Inserimento CVV . . . . .	11
4	Informazioni rubate . . . . .	11
5	Wallet dopo la chiamata request.show() . . . . .	14
6	Wallet in caso di successo . . . . .	15
7	Wallet in caso di fail . . . . .	15
8	Compatibilità desktop . . . . .	17
9	Compatibilità mobile . . . . .	17

## Riferimenti bibliografici

- [1] Sheppy, poshaughnessy, jpmedley, echenley, chrisdavidmills, marcoscaceres, riking, amZotti, andersnorgaard, erikadoyle, agektmr, reaktivo, dgashmdn *Payment Request API*, Mozilla Developer, [https://developer.mozilla.org/en-US/docs/Web/API/Payment\\_Request\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Payment_Request_API)
- [2] W3C (2018), *W3C Payment Request API*, Mozilla Developer, <https://w3c.github.io/payment-request/>
- [3] Natasja Bolton *One more nail in the coffin for iFrames?*, <https://sysnetgs.com/2016/06/one-nail-coffin-iframes/>
- [4] Matt Gaunt *Deep Dive into the Payment Request API*, <https://developers.google.com/web/fundamentals/payments/deep-dive-into-payment-request>
- [5] *What is Payment Request?*, <https://paymentrequest.show/>
- [6] *Introduction to the Payment Request API*, <https://developers.google.com/web/ilt/pwa/introduction-to-the-payment-request-api>
- [7] Eiji Kitamura, *Bringing Easy and Fast Checkout with Payment Request API*, <https://developers.google.com/web/updates/2016/07/payment-request>
- [8] Microsoft, *Payment Request API*, <https://docs.microsoft.com/en-us/microsoft-edge/dev-guide/windows-integration/payment-request-api>
- [9] Microsoft, *Payment Request API samples*, <https://developer.microsoft.com/en-us/microsoft-edge/testdrive/demos/paymentrequest/>
- [10] *Simpler web payments: Introducing the Payment Request API*, <https://blogs.windows.com/msedgedev/2016/12/15/payment-request-api-edge/#UATsm3ejAT9oYtrj.97>
- [11] Google Inc.(2018), <https://googlechrome.github.io/samples/paymentrequest/credit-cards/>