SPRING BOOT + ANGULAR +STRIPE

# PAGOS CON STRIPE EN SPRING BOOT Y ANGULAR 8. CAPÍTULO 7: FORMULARIO DE PAGO

Llega el momento de implementar el pago desde el cliente. Aquí vamos a recoger los datos de la tarjeta y del artículo a comprar para enviarlo al servidor, que es el encargado de enviar las credenciales y los datos a Stripe.

**Modal:**

Tenemos un modal que mostrará los datos del pago.  Antes de nada necesitamos un servicio para los métodos cancelar y confirmar del backend pues el modal será el encargado de invocarlos. payment.service.ts queda así:

|  |  |
| --- | --- |
|  | import { Injectable } from '@angular/core'; |
|  | import { HttpClient, HttpHeaders } from '@angular/common/http'; |
|  | import { Observable } from 'rxjs'; |
|  |  |
|  | const cabecera = {headers: new HttpHeaders({'Content-Type': 'application/json'})}; |
|  |  |
|  | @Injectable({ |
|  | providedIn: 'root' |
|  | }) |
|  | export class PaymentService { |
|  |  |
|  | stripeURL = 'http://localhost:8080/stripe/'; |
|  |  |
|  | constructor(private httpClient: HttpClient) { } |
|  |  |
|  | public pagar(paymentIntentDto: PaymentIntentDto): Observable<string> { |
|  | return this.httpClient.post<string>(this.stripeURL + 'paymentintent', paymentIntentDto, cabecera); |
|  | } |
|  |  |
|  | public confirmar(id: string): Observable<string> { |
|  | return this.httpClient.post<string>(this.stripeURL + `confirm/${id}`, {}, cabecera); |
|  | } |
|  |  |
|  | public cancelar(id: string): Observable<string> { |
|  | return this.httpClient.post<string>(this.stripeURL + `cancel/${id}`, {}, cabecera); |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/52aa45f1d7be4abe460eba2300b3e7b6/raw/fdf7096d4a647c721befae2bbbec860fac8d1944/payment.service.ts)[**payment.service.ts**](https://gist.github.com/cavanosa/52aa45f1d7be4abe460eba2300b3e7b6#file-payment-service-ts) hosted with  by [**GitHub**](https://github.com/)

Vamos a continuación a modal.component.ts:

|  |  |
| --- | --- |
|  | import { Component, OnInit, Input } from '@angular/core'; |
|  | import { NgbActiveModal } from '@ng-bootstrap/ng-bootstrap'; |
|  |  |
|  | @Component({ |
|  | selector: 'app-modal', |
|  | templateUrl: './modal.component.html', |
|  | styleUrls: ['./modal.component.css'] |
|  | }) |
|  | export class ModalComponent implements OnInit { |
|  |  |
|  | @Input() id; |
|  | @Input() nombre; |
|  | @Input() descripcion; |
|  | @Input() precio; |
|  |  |
|  | constructor(public activeModal: NgbActiveModal) { } |
|  |  |
|  | ngOnInit() { |
|  | } |
|  |  |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/51e7267cd60f401fd521100d73880ff7/raw/6e8b462b2497a5b949090b7974c1b3a2fb368b1d/modal.component.ts)[**modal.component.ts**](https://gist.github.com/cavanosa/51e7267cd60f401fd521100d73880ff7#file-modal-component-ts) hosted with  by [**GitHub**](https://github.com/)

Habrá tres campos que mostrará por pantalla. El componente padre, que será el encargado de abrir el modal, también se tendrá que encargar de enviárselos, por eso utilizamos el decorador @Input.

Y la vista del modal muestra esos datos:

|  |  |
| --- | --- |
|  | <div class="modal-header"> |
|  | <h4 class="modal-title">¿Deseas confirmar el pago?</h4> |
|  | <button type="button" class="close" aria-label="Close" (click)="activeModal.dismiss('Cross click')"> |
|  | <span aria-hidden="true">&times;</span> |
|  | </button> |
|  | </div> |
|  | <div class="modal-body"> |
|  | <h5>Datos del pago:</h5> |
|  | <p><b>ID:</b> {{id}}</p> |
|  | <p><b>Nombre:</b> {{nombre}}</p> |
|  | <p><b>Descripción:</b> {{descripcion}}</p> |
|  | <p><b>Precio:</b> {{precio / 100}}€</p> |
|  | </div> |
|  | <div class="modal-footer d-flex justify-content-between"> |
|  | <button type="button" class="btn btn-outline-success">Confirmar</button> |
|  | <button type="button" class="btn btn-outline-danger">Cancelar</button> |
|  | </div> |

[**view raw**](https://gist.github.com/cavanosa/170287c6f79a662d152c590fa14a30a8/raw/577053c6432aec459644cce3eefa09dc39555411/modal.component.html)[**modal.component.html**](https://gist.github.com/cavanosa/170287c6f79a662d152c590fa14a30a8#file-modal-component-html) hosted with  by [**GitHub**](https://github.com/)

**Payment:**

Llegamos a la parte más complicada del front. Vamos a payment.component.ts para modificarlo. Lo que vamos a hacer es crear el formulario de pago, llamar al servicio de Stripe y mostrar el resultado del modal.

Tenemos dos formas de crear el formulario de la tarjeta. La primera, que es la que vamos a utilizar, con Stripe Elements; y la segunda, que es un poco más sencilla, con el componente StripeCardComponent. Todo esto viene bien explicado en la [documentación](https://www.npmjs.com/package/ngx-stripe).

El código:

|  |  |
| --- | --- |
|  | import { Component, OnInit, Input } from '@angular/core'; |
|  | import { NgbModal } from '@ng-bootstrap/ng-bootstrap'; |
|  | import { ModalComponent } from '../modal/modal.component'; |
|  | import { StripeService, Elements, Element as StripeElement, ElementsOptions } from 'ngx-stripe'; |
|  | import { FormGroup, FormControl, Validators } from '@angular/forms'; |
|  | import { PaymentService } from './payment.service'; |
|  | import { Router } from '@angular/router'; |
|  | import { PaymentIntentDto } from '../model/payment-intent-dto'; |
|  |  |
|  | @Component({ |
|  | selector: 'app-payment', |
|  | templateUrl: './payment.component.html', |
|  | styleUrls: ['./payment.component.css'] |
|  | }) |
|  | export class PaymentComponent implements OnInit { |
|  |  |
|  | elements: Elements; |
|  | card: StripeElement; |
|  |  |
|  | elementsOptions: ElementsOptions = { |
|  | locale: 'es' |
|  | }; |
|  |  |
|  | @Input() precio; |
|  | @Input() descripcion; |
|  | @Input() nombre; |
|  |  |
|  | error: any; |
|  |  |
|  | constructor( |
|  | private stripeService: StripeService, |
|  | private modalService: NgbModal, |
|  | private paymentService: PaymentService, |
|  | private router: Router |
|  | ) { } |
|  |  |
|  | public stripeForm = new FormGroup({ |
|  | name: new FormControl('', Validators.required), |
|  | }); |
|  |  |
|  | ngOnInit() { |
|  | this.stripeService.elements(this.elementsOptions).subscribe( |
|  | elements => { |
|  | this.elements = elements; |
|  | if (!this.card) { |
|  | this.card = this.elements.create('card', { |
|  | style: { |
|  | base: { |
|  | iconColor: '#666EE8', |
|  | color: '#31325F', |
|  | lineHeight: '40px', |
|  | fontWeight: 300, |
|  | fontFamily: '"Helvetica Neue", Helvetica, sans-serif', |
|  | fontSize: '18px', |
|  | '::placeholder': { |
|  | color: '#CFD7E0' |
|  | } |
|  | } |
|  | } |
|  | }); |
|  | this.card.mount('#card-element'); |
|  | } |
|  | } |
|  | ); |
|  | } |
|  |  |
|  | buy() { |
|  | const name = this.stripeForm.get('name').value; |
|  | this.stripeService |
|  | .createToken(this.card, { name }) |
|  | .subscribe(result => { |
|  | if (result.token) { |
|  | const paymentIntentDto: PaymentIntentDto = { |
|  | token: result.token.id, |
|  | amount: this.precio, |
|  | currency: 'EUR', |
|  | description: this.descripcion |
|  | }; |
|  | this.paymentService.pagar(paymentIntentDto).subscribe( |
|  | data => { |
|  | this.abrirModal(data[`id`], this.nombre, data[`description`], data[`amount`]); |
|  | this.router.navigate(['/']); |
|  | } |
|  | ); |
|  | this.error = undefined; |
|  | } else if (result.error) { |
|  | this.error = result.error.message; |
|  | } |
|  | }); |
|  | } |
|  |  |
|  | abrirModal(id: string, nombre: string, descripcion: string, precio: number) { |
|  | const modalRef = this.modalService.open(ModalComponent); |
|  | modalRef.componentInstance.id = id; |
|  | modalRef.componentInstance.nombre = nombre; |
|  | modalRef.componentInstance.descripcion = descripcion; |
|  | modalRef.componentInstance.precio = precio; |
|  | } |
|  |  |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/38d1adc1b893d9ef7769a2bdd9f822ed/raw/558186e8324fe6e0fb2cfba876cb5aa5be4d5aa8/payment.component.ts)[**payment.component.ts**](https://gist.github.com/cavanosa/38d1adc1b893d9ef7769a2bdd9f822ed#file-payment-component-ts) hosted with  by [**GitHub**](https://github.com/)

Aquí ya hay más cosas que explicar. Lo primero que hacemos es crear dos objetos, elements y card. En el import le ponemos un alias a la tarjeta:

import { Elements, Element as StripeElement, ElementsOptions, StripeService } from 'ngx-stripe';

Para que los campos aparezcan en castellano utilizamos elementsOptions con locale = ‘es’.

Como este componente irá incrustado dentro del componente detalle (el padre es detalle y el hijo payment) obtendremos el precio, la descripción y el nombre del artículo a partir del padre, por lo que utilizamos el decorador @Input.

La variable error es la que mostrará el mensaje de que los datos de la tarjeta están vacíos o son erróneos.

Añadimos el campo name al formulario con FormGroup, que se utilizará junto a card para crear el token.

A continuación creamos el formulario de pago con tarjeta en el método ngInit.

El método clave es buy, donde primero obtenemos el token y creamos un objeto que será enviado al backend. Una vez hecho, se ejecuta el método post del cliente http que ejecuta el paymentintent. El httpClient devuelve un observable al que nos suscribimos y, si no ha habido errores, se abre el modal. Para ello se llama al método abrirModal. Dicho método lleva varios parámetros (el id, el nombre del producto, su precio y descripción) y abre el modal, que mostrará los valores de dichos parámetros.

En la vista payment-component.html ponemos un formulario:

|  |  |
| --- | --- |
|  | <form novalidate (ngSubmit)="buy()" [formGroup]="stripeForm" #f="ngForm"> |
|  | <div class="form-group"> |
|  | <input class="form-control" type="text" id="name" formControlName="name" placeholder="tu nombre" required> |
|  | </div> |
|  |  |
|  | <div id="card-element" class="field"></div> |
|  | <div \*ngIf="error" class="alert alert-danger"> |
|  | {{error}} |
|  | </div> |
|  |  |
|  | <button class="btn btn-outline-success" type="submit" [disabled]="!f.valid"> |
|  | Pagar |
|  | </button> |
|  | </form> |

[**view raw**](https://gist.github.com/cavanosa/f424854385b5c2dd9323c5fc0769c643/raw/d71f099aea4c269eeac826a7b413eaba681431b7/payment.component.html)[**payment.component.html**](https://gist.github.com/cavanosa/f424854385b5c2dd9323c5fc0769c643#file-payment-component-html) hosted with  by [**GitHub**](https://github.com/)

Y ahora tan sólo nos queda pasar los valores al componente payment desde el padre, detalle-articulo.componenet.html. Lo dejamos así:

|  |  |
| --- | --- |
|  | <div class="container mt-3"> |
|  | <div class="card p-3" style="width: 28rem;" \*ngIf="articulo"> |
|  | <img src="{{articulo.imagenURL}}" class="card-img-top mx-auto" alt="..." style="max-height: 200px; width: auto"> |
|  | <div class="card-body mx-auto"> |
|  | <h5 class="card-title">{{articulo.nombre}}</h5> |
|  | <p class="card-text">{{articulo.descripcion}}</p> |
|  | <p><b>Precio: </b>{{articulo.precio / 100}}€</p> |
|  | </div> |
|  | <div class="card-body"> |
|  | // desde el componente padre le pasamos los valores con el decorador @Input |
|  | <app-payment nombre="{{articulo.nombre}}" precio="{{articulo.precio}}" descripcion="{{articulo.descripcion}}"> |
|  | </app-payment> |
|  | </div> |
|  | </div> |
|  | </div> |

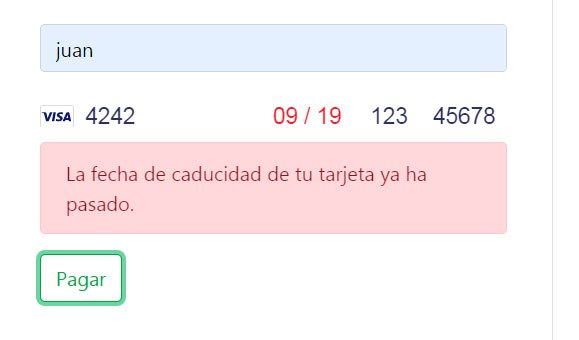
[**view raw**](https://gist.github.com/cavanosa/0dfb662fd5719cc1064583ab7404ba44/raw/9fe91740ea3c988c6eef7989fbf67de506f1329e/detalle-articulo.component.html)[**detalle-articulo.component.html**](https://gist.github.com/cavanosa/0dfb662fd5719cc1064583ab7404ba44#file-detalle-articulo-component-html) hosted with  by [**GitHub**](https://github.com/)

Probando la app:

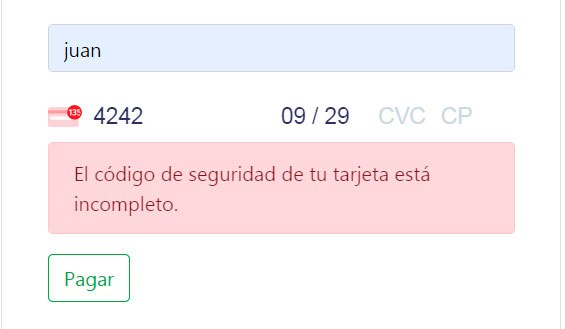
Y llegó la hora de la verdad. Nos vamos al formulario, ponemos el nombre y pulsamos el botón “Pagar”. Nos mostrará un error:



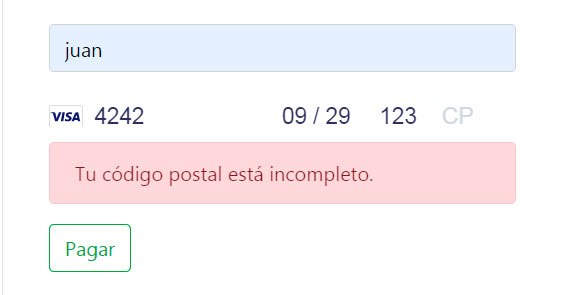
Para utilizar una tarjeta de prueba ponemos 4242 4242 4242 4242. El mes y año deben ser iguales o posteriores a la fecha actual, pues si ponemos un mes anterior al actual los números aparecerán en rojo.



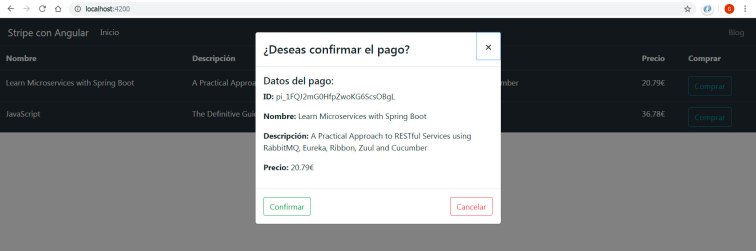
CVC debe ser una cadena de tres números y CP de cinco. Si los dejamos vacíos también hay error:



Y lo mismo con el código postal:

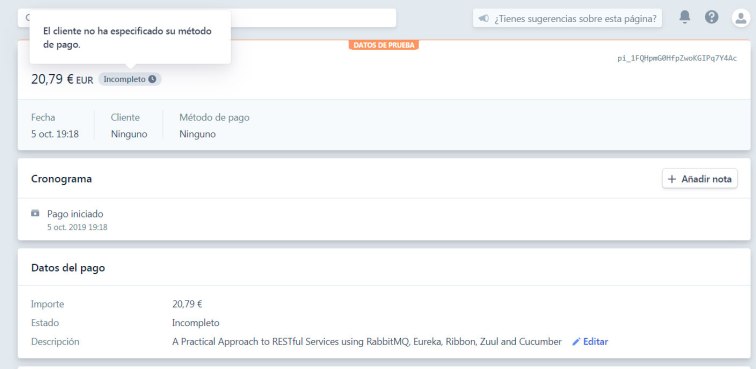


Ponemos los valores correctos y pulsamos pagar. Si todo va ok nos mostrará el modal que nos pregunta si deseamos confirmar e pago. De momento no hemos implementado las operaciones de confirmar y cancelar, así que cerramos el modal con el botón x.



**Comprobar la cuenta de Stripe:**

Si vamos a nuestra cuenta de Stripe, veremos que el pago se ha realizado pero está incompleto (indica que no se ha especificado el método de pago).



No os asustéis, es normal, pues de este modo Stripe nos permite cancelar el pago o confirmarlo.  Por defecto, el status del PaymentIntent será “requires\_payment\_method” hasta que no se especifique el método de pago. Podéis ver todo esto con más detalle en la [documentación](https://stripe.com/docs/api/payment_intents/confirm).

**Confirmar y cancelar:**

Nos queda solamente implementar estos dos métodos. Modificamos el archivo modal.component.ts, inyectando los servicios payment y toastr y creando dos métodos, confirmar y cancelar:

|  |  |
| --- | --- |
|  | import { Component, OnInit, Input } from '@angular/core'; |
|  | import { NgbActiveModal } from '@ng-bootstrap/ng-bootstrap'; |
|  | import { PaymentService } from '../payment/payment.service'; |
|  | import { ToastrService } from 'ngx-toastr'; |
|  |  |
|  | @Component({ |
|  | selector: 'app-modal', |
|  | templateUrl: './modal.component.html', |
|  | styleUrls: ['./modal.component.css'] |
|  | }) |
|  | export class ModalComponent implements OnInit { |
|  |  |
|  | @Input() id; |
|  | @Input() nombre; |
|  | @Input() descripcion; |
|  | @Input() precio; |
|  |  |
|  | constructor( |
|  | public activeModal: NgbActiveModal, |
|  | private paymentService: PaymentService, |
|  | private toastrService: ToastrService |
|  | ) { } |
|  |  |
|  | ngOnInit() { |
|  | } |
|  |  |
|  | confirmar(id: string): void { |
|  | this.paymentService.confirmar(id).subscribe( |
|  | data => { |
|  | // console.log(data); |
|  | this.toastrService.success |
|  | ('pago confirmado', 'se ha confirmado el pago con id ' + data[`id`], {positionClass: 'toast-top-center', timeOut: 3000}); |
|  | this.activeModal.close(); |
|  | }, |
|  | err => { |
|  | console.log(err); |
|  | this.activeModal.close(); |
|  | } |
|  | ); |
|  | } |
|  |  |
|  | cancelar(id: string): void { |
|  | this.paymentService.cancelar(id).subscribe( |
|  | data => { |
|  | // console.log(data); |
|  | this.toastrService.success |
|  | ('pago cancelado', 'se ha cancelado el pago con id ' + data[`id`], {positionClass: 'toast-top-center', timeOut: 3000}); |
|  | this.activeModal.close(); |
|  | }, |
|  | err => { |
|  | console.log(err); |
|  | this.activeModal.close(); |
|  | } |
|  | ); |
|  | } |
|  |  |
|  | } |

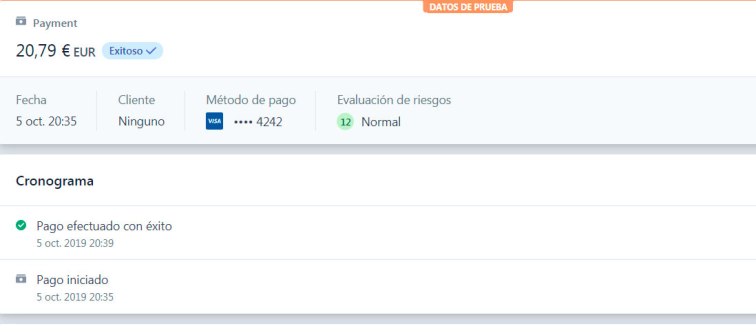
[**view raw**](https://gist.github.com/cavanosa/1ec5f4456b43fe637dd557206dd136b6/raw/1f94febc196ffe88ed6a29621d9a3363f050886e/modal.component.ts)[**modal.component.ts**](https://gist.github.com/cavanosa/1ec5f4456b43fe637dd557206dd136b6#file-modal-component-ts) hosted with  by [**GitHub**](https://github.com/)

Y en la vista hay que ejecutar dichos métodos conlos botones a partir del evento click:

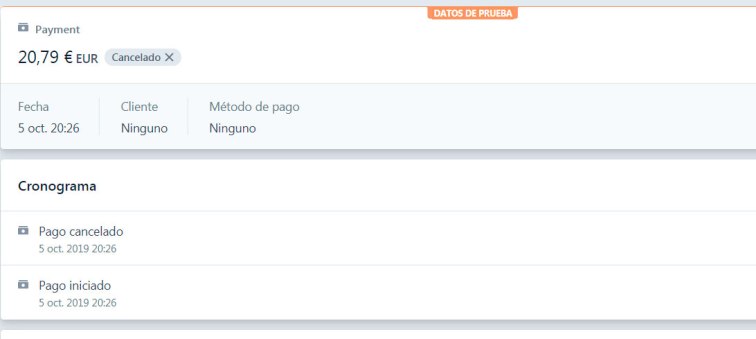
|  |  |
| --- | --- |
|  | <div class="modal-header"> |
|  | <h4 class="modal-title">¿Deseas confirmar el pago?</h4> |
|  | <button type="button" class="close" aria-label="Close" (click)="activeModal.dismiss('Cross click')"> |
|  | <span aria-hidden="true">&times;</span> |
|  | </button> |
|  | </div> |
|  | <div class="modal-body"> |
|  | <h5>Datos del pago:</h5> |
|  | <p><b>ID:</b> {{id}}</p> |
|  | <p><b>Nombre:</b> {{nombre}}</p> |
|  | <p><b>Descripción:</b> {{descripcion}}</p> |
|  | <p><b>Precio:</b> {{precio / 100}}€</p> |
|  | </div> |
|  | <div class="modal-footer d-flex justify-content-between"> |
|  | <button type="button" class="btn btn-outline-success" (click)="confirmar(id)">Confirmar</button> |
|  | <button type="button" class="btn btn-outline-danger" (click)="cancelar(id)">Cancelar</button> |
|  | </div> |

[**view raw**](https://gist.github.com/cavanosa/e3671bedffb24fc22bc5a182f463dd40/raw/523541ecc47fda2041467a82437147dd4d3c6499/modal.component.html)[**modal.component.html**](https://gist.github.com/cavanosa/e3671bedffb24fc22bc5a182f463dd40#file-modal-component-html) hosted with  by [**GitHub**](https://github.com/)

Y al fin hemos terminado el proyecto. Si probamos los botones y vamos a la cuenta de Stripe podremos comprobar que los pagos ahora se confirman o se cancelan. En el caso de confirmar:



Y si lo cancelamos:



Y esto ha sido todo. Nos vemos en el próximo tutorial.