



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programador Web Avanzado

Clase N° 12: Spotify – Mercado Pago

Profesor: Ing. Leandro Rodolfo Gil Carrano

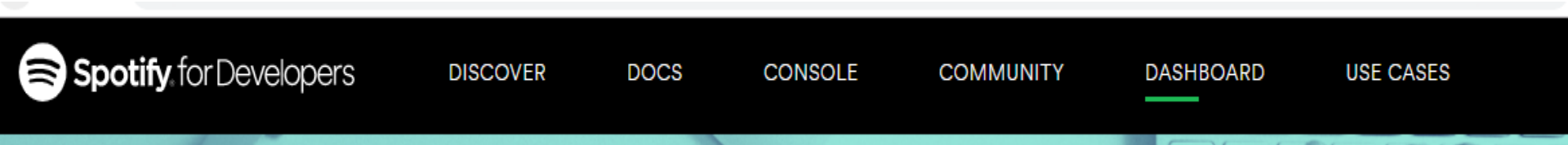
Email: leangilutn@gmail.com

Spotify

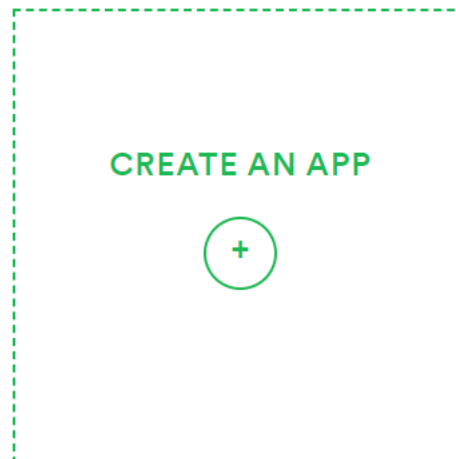
Spotify

Ingresar a <https://developer.spotify.com/>

Dirigirse a “Dashboard”



Loguearse en la plataforma y luego hacer click en “Crear Aplicación”



Spotify

Colocar nombre y descripción de la aplicación

CREATE AN APP OR HARDWARE INTEGRATION

Step 1/3

App or Hardware Name *

Test2

App or Hardware Description *

Test2

Spotify

Seleccionar tipo de aplicación a desarrollar

What are you building? *

- ☐ I don't know
- ☐ Mobile App
- ☒ Desktop App
- ☐ Website
- ☐ Speakers
- ☐ Voice - Cortana
- ☐ Voice - Other
- ☐ TV
- ☐ Gaming Consoles
- ☐ Wearables

Check all that apply

NEXT

Spotify

Colocar si la aplicación a desarrollar tendrá fines comerciales o no

CREATE AN APP OR HARDWARE INTEGRATION

Step 2/3

Are you developing a commercial integration?

Generally speaking, you have a commercial integration if you are incorporated and/or plan to monetize your app.

What does that mean?

- **You monetize if...** your app or product charges a fee to purchase, promotes using ads or utilizes a freemium model for upsells.
- **You are incorporated if...** you are legally protected as a corporate entity or company.

Note: Please note that using Spotify's tools are always subject to our [Developer Terms of Service](#) and final approvals for commercial usage will always be left up to the discretion of Spotify and its partners.

NO

YES

Spotify

Aceptar términos y condiciones

CREATE A NON-COMMERCIAL APP

Step 3/3

- ☐ I understand that this app is **not** for commercial use
- ☐ I understand that I cannot migrate my app from non-commercial to commercial without permission
- ☐ I understand and agree with Spotify's [Developer Terms of Service](#), [Branding Guidelines](#), and [Privacy Policy](#)

SUBMIT

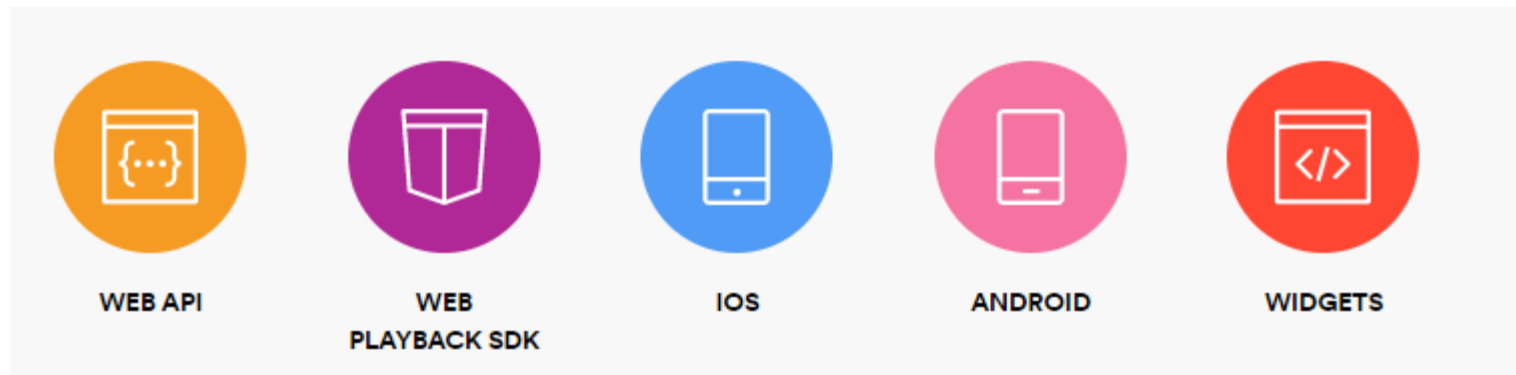
CANCEL

Spotify

En “Docs” podremos visualizar la documentación disponible

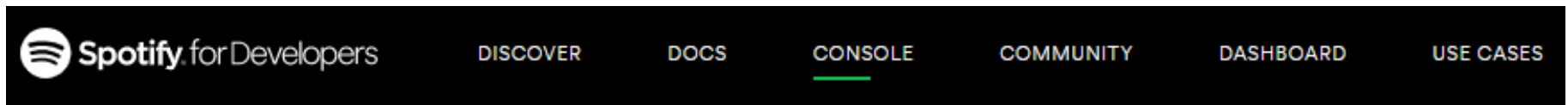


La documentación esta clasificada según plataforma

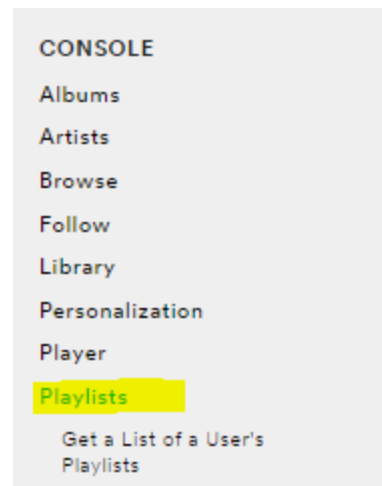


Spotify

En “Console” se podrán ejecutar los request a la api de spotify



Ejemplo consultar las playlist



Psst! Check out [our brand-new Web API Reference in beta!](#)
And be sure to tweet us your feedback at [@SpotifyPlatform on Twitter!](#)

Playlists

| METHOD | ENDPOINT | USAGE |
|--------|---|--|
| DELETE | <code>/v1/playlists/{playlist_id}/tracks</code> | Remove Tracks from a Playlist |
| GET | <code>/v1/me/playlists</code> | Get a List of Current User's Playlists |
| GET | <code>/v1/playlists/{playlist_id}/tracks</code> | Get a Playlist's Tracks |



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Spotify

Obtener “token” con “getToken”

Playlists

Get a List of Current User's Playlists

| | |
|-------------|---|
| Description | Get a List of Current User's Playlists DOCS |
| Endpoint | <code>https://api.spotify.com/v1/me/playlists</code> |
| HTTP Method | GET |
| OAuth | Required |

limit

offset

OAuth Token

GET TOKEN



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Spotify

Luego hacer click en “TryIt”

TRY IT

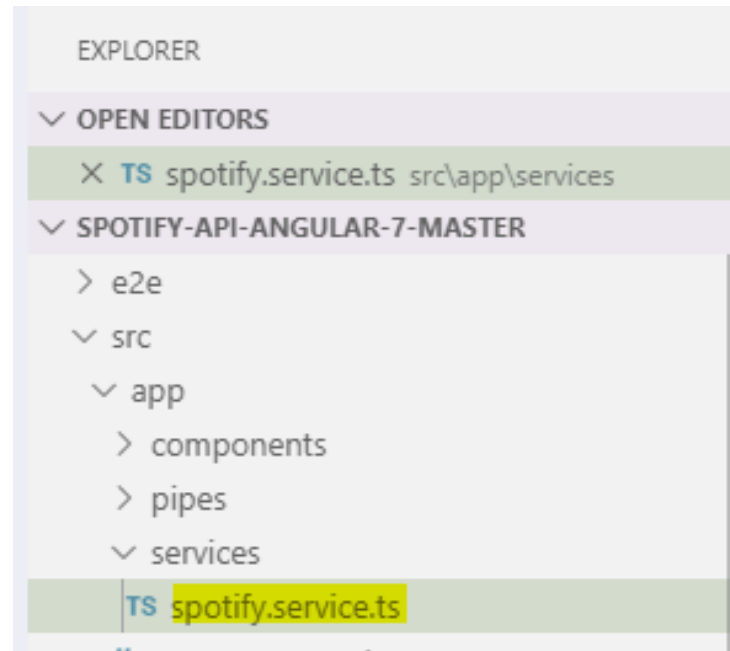
FILL SAMPLE DATA

```
{
  "href": "https://api.spotify.com/v1/users/leangilcarrano/playlists?
offset=0&limit=20",
  "items": [
    {
      "collaborative": false,
      "external_urls": {
        "spotify": "https://open.spotify.com/playlist/7qtWvXp0FcG1o9CwDGzBcy"
      },
      "href": "https://api.spotify.com/v1/playlists/7qtWvXp0FcG1o9CwDGzBcy",
      "id": "7qtWvXp0FcG1o9CwDGzBcy",
      "images": [
        /
```

Spotify

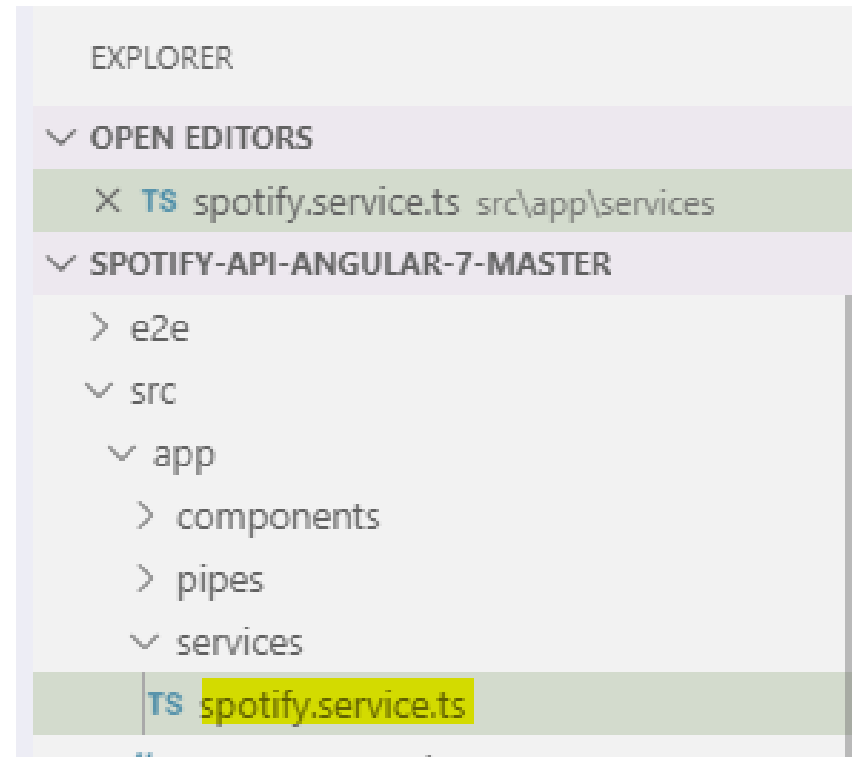
Ejemplo en Angular

Ejecutar **npm install** dentro del directorio de la aplicación



Spotify

Ubicar “spotify.service.ts”



Spotify

Modificar token, por uno actualizado

```
// ... headers que me da spotify ...  
const headers = new HttpHeaders({  
  Authorization:  
    "Bearer BQCItPNevyVExFrIcz2Lu36MiTNW7A-6pmsZDcfd302RoQhiK97POPBWecTOu4B38DARuVmJjvBPzfI-Mvo"  
});
```

Probar aplicación que tiene los reques a la api de spotify aplicados

Mercado Pago

Mercado Pago

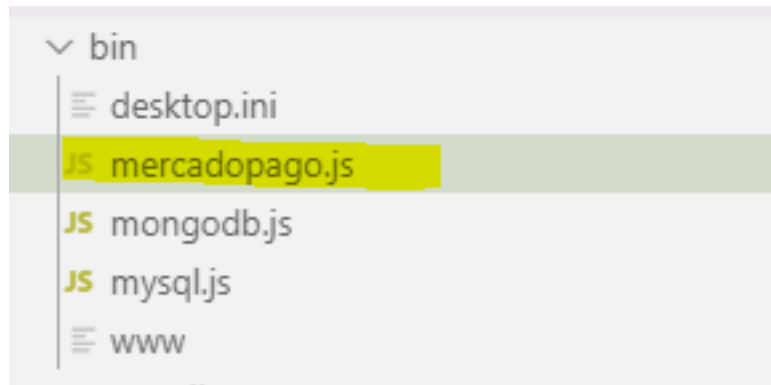
Acceder a <https://www.mercadopago.com.ar/developers>

En la aplicación de express ejecutar
npm install mercadopago

```
npm install mercadopago
```


Mercado Pago

Crear un nuevo archivo en directorio “bin”



En ese archivo colocamos la configuración con el sdk de mercado pago

```
const mp = require('mercadopago');

mp.configure({
  sandbox : 'true',
  access_token : 'TEST-3707140219808292-101422-106f590e953337fe6c614c2caded5b29-45517724'
})
```

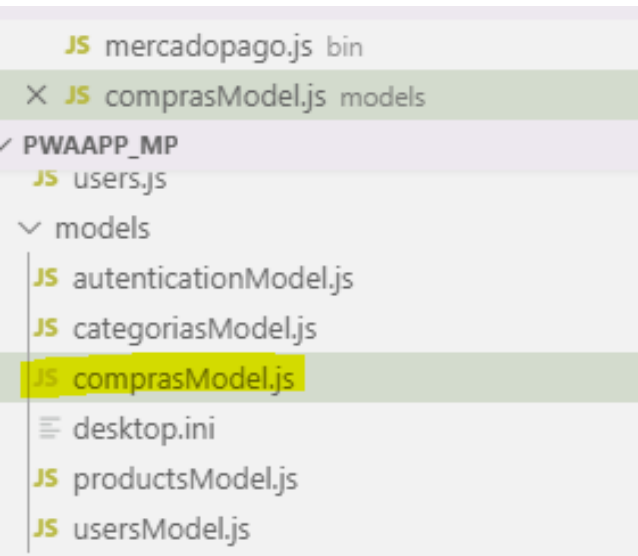
Mercado Pago

En el mismo archivo colocamos el llamado al método “create” para realizar una compra

```
async function comprar(preference) {  
  try {  
    return await mp.preferences.create(preference);  
  } catch(error) {  
    throw error;  
  }  
}  
  
module.exports = {comprar}
```

Mercado Pago

Generar el modelo “comprasModel”



```
1  const mongoose = require('../bin/mongodb');
2
3  //Define a schema
4  const Schema = mongoose.Schema;
5  const CategoriaSchema = new Schema({
6    importe: {
7      type: Number,
8      required: true,
9    },
10   producto: {type:Schema.ObjectId, ref:"products"},
11 });
12 module.exports = mongoose.model('compras', CategoriaSchema);
13
```

Mercado Pago

Luego incluimos este modelo en el controller y el modulo generado para mercadopago



The screenshot shows a VS Code interface. On the left, the 'OPEN EDITORS' panel lists 'mercadopago.js bin', 'comprasModel.js models', and 'compras.js controllers'. Below it, the 'PWAAPP_MP' project structure is visible, including 'www' and 'controllers'. The 'controllers' folder is expanded, showing 'autentication.js' and 'compras.js'. The 'compras.js' file is selected and highlighted in yellow. On the right, the code editor shows the content of 'compras.js'. The code includes two 'require' statements at the top, followed by a 'module.exports' object containing a 'save' property. The 'save' property is an async function that uses 'comprasModel' to create a new 'compra' object with 'importe' and 'producto' fields.

```
controllers > JS compras.js > ...  
1  var comprasModel = require("../models/comprasModel")  
2  var mp = require("../bin/mercadopago");  
3  
4  module.exports = {  
5    save: async function(req, res, next) {  
6      try{  
7  
8        var compra = new comprasModel({  
9          importe: 1000,  
10         producto: req.body.product_id
```

Mercado Pago

Generamos método **save** para generar la compra

```
module.exports = {  
  save: async function(req, res, next) {  
    try{  
  
      var compra = new comprasModel({  
        importe: 1000,  
        producto: req.body.product_id  
      });  
      var result = await compra.save();  
      console.log(result["_id"]);  
      let preference = {  
        items : [  
          {  
            id : result["_id"],  
            title : 'Compra Carrito',  
            quantity : 1,  
            currency_id : 'ARS',  
            unit_price : result["importe"]  
          }  
        ],  
        payer : {  
          email : 'leangilutn@gmail.com'  
        },  
        notification_url : 'http://miurl.com/'  
      }  
    }  
  }  
}
```

En este método generamos un nuevo documento en la base de mongodb para realizar la compra.

En base al documento generado en la colección de “compras” generamos el pago en Mercado pago

Es importante que el id de “preference” refleje el id de compra.

“notification_url”: IPN de mercadopago, es una url nuestra la cual será invocada por MP ante cada cambio de estado en el pago

Mercado Pago

```
let dato_return = await mp.comprar(preference);  
//console.log(dato_return);  
//return dato_return.body.sandbox_init_point;  
res.status(200).json({status: "success", message: "Compra added successfully!!!", data: dato_return})  
}catch(error) {  
  next(error)  
}  
}  
}
```

Una vez generada la compra en mongodb y definido el documento para generar la compra en MP, se invoca el método “comprar” del modulo “mercadopago” (ubicado en /bin)

Mercado Pago

Generar las rutas para llamar al controlador

✓ OPEN EDITORS

- JS mercadopago.js bin
- JS comprasModel.js models
- JS compras.js controllers
- ✕ JS compras.js routes

✓ PWAAPP_MP

- > bin
- > controllers
- > models
- > node_modules
- > public
- ✓ routes
 - JS authentication.js
 - JS compras.js

routes > JS compras.js > ...

```
1 var express = require('express');
2 var router = express.Router();
3 var compras = require("../controllers/compras")
4
5 /* GET home page. */
6 router.post('/', compras.save);
7 module.exports = router;
8
9
```

Mercado Pago

```
var usersRouter = require('./routes/users');  
var productosRouter = require('./routes/productos');  
var comprasRouter = require('./routes/compras');  
var autenticacionRouter = require('./routes/authentication');
```

```
app.use('/users', usersRouter);  
app.use('/authentication', autenticacionRouter);  
app.use('/products', validateUser, productosRouter);  
app.use('/compras', /*validateUser,*/ comprasRouter);
```


Mercado Pago

Desde angular debemos generar un servicio de compras



The image shows a screenshot of a code editor (VS Code) with two panels. The left panel is the file explorer, showing a project structure with a folder 'APLICACION_PARTE2' containing several TypeScript files. The file 'compras.service.ts' is selected and highlighted. The right panel shows the code for 'compras.service.ts'. The code imports 'Injectable' and 'HttpClient' from '@angular/core' and '@angular/common/http' respectively. It defines an '@Injectable' class 'ComprasService' with a 'providedIn: 'root'' property. The class has a constructor that takes a private 'http: HttpClient' parameter. It also has a method 'prueba(product_id)' that returns 'this.http.post' with a URL 'http://localhost:3000/compras/' and a body object containing 'product_id'.

```
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http'
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class ComprasService {
7
8   constructor(private http:HttpClient) { }
9
10  prueba(product_id){
11    return this.http.post('http://localhost:3000/compras/',{
12      | "product_id":product_id
13    })
14  }
15 }
```

En este servicio realizamos el request al servicio brindado por la api rest realizada en node js



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Mercado Pago

Instanciamos este servicio en el controlador de “Producto”

```
export class ProductoComponent implements OnInit {  
  @Input()  
  data  
  @Output()  
  propagar = new EventEmitter<String>()  
  constructor(public compras:ComprasService, public route:Router) { }  
  onPropagar(){  
    this.propagar.emit("dasdasd")  
  }  
  comprar(){  
    this.compras.prueba(this.data["_id"]).subscribe(data=>{  
      console.log("Compra ",data["data"]["body"]["init_point"]);  
      window.open(data["data"]["body"]["init_point"], "_blank");  
    })  
  }  
  ngOnInit() {  
    console.log(this.data)  
  }  
}
```

El init_point tendrá la url brindada por MP para realizar el pago, debemos redirigir a la misma para que el usuario complete la compra

Mercado Pago

Desde la vista colocamos un button que llame al método comprar

```
<div>
  <mat-card>
    <div >{{data.name}}</div>
    <div >{{data.sku}}</div>
    <div >{{data.price}}</div>
    <button mat-raised-button color="primary" routerLink="/productos/{{data._id}}">Ver detalle</button>
    <button mat-raised-button color="primary" (click)="comprar()">Comprar</button>
  </mat-card>
</div>
```

Ver documentación en

[https://www.mercadopago.com.ar/developers/es/plugins_sdks/sdks/official/nodejs/#bookmark configuraci%C3%B3n](https://www.mercadopago.com.ar/developers/es/plugins_sdks/sdks/official/nodejs/#bookmark_configuraci%C3%B3n)