# Guitar Catalog

## Manual del Administrador

**Realizado por:**

**Alejandro Valle Rodríguez**

# Índice

# 1. INTRODUCCIÓN

Guitar Catalog es un proyecto que se basa en ofrecer un listado actualizado diariamente con guitarras de las marcas más populares del marcado y algunas marcas menos conocidas.

La necesidad de creación de este proyecto se basa en la poca variedad de páginas web que ofrezcan este tipo de servicio, puesto que la mayoría de páginas web son tiendas y no hay ninguna que ofrezca toda la variedad disponible, por lo que Guitar Catalog sería una propuesta perfecta para el mundo de la música.

# 2. OBJETIVOS DEL PROYECTO

El objetivo principal de Guitar Catalog es ofrecer un listado actualizado de una gran variedad de marcas de guitarras, habiendo guitarras disponibles en el mercado actual y guitarras descatalogadas. Para ello, se necesitarán las siguientes funcionalidades:

- Base de datos:
  - Una base de datos mySql donde almacenar guitarras y marcas.
- API .NET Core:
  - API web encargada de ofrecer métodos para listar en formato Json las distintas guitarras almacenadas.
- Scrapings:
  - Programas de Python encargados de recopilar la información más relevante de cada guitarra encontrada en las páginas oficiales de cada marca. Además, un programa principal encargado de llamar a cada scraping.
- Aplicación Android:
  - Aplicación encargada llamar a la API previamente creada, obtener el listado y mostrarlo al usuario pudiendo filtrar y ordenar los resultados, guardar las guitarras favoritas y hacer búsquedas personalizadas.

# 3. PLANIFICACIÓN DEL PROYECTO

Este proyecto tiene una estimación de 160 horas de trabajo además de todo el soporte que necesitará en el futuro y posibles extensiones.

La estimación de coste se podría dividir en secciones, siendo estas:

- Base de datos: 15h aproximadamente
- API .NET Core: 25 aproximadamente
- Scrapings: 50h aproximadamente
- Aplicación Android: 70h aproximadamente

# 4. ANÁLISIS Y DISEÑO DEL SISTEMA

Los datos se almacenan en una base de datos MySql, en la que se encuentra un procedimiento almacenado para insertar guitarras solo si no existía antes y dos tablas, guitarra y marca.

La tabla guitarra es donde se guardan los datos de las guitarras y marca es donde se guardan los datos necesarios para hacer el scraping de cada marca.

| guitarra | | |
|---|---|---|
| id | int | PK |
| idExterno | VARCHAR(70) | |
| marca | VARCHAR(15) | |
| modelo | VARCHAR(70) | |
| forma | VARCHAR(15) | |
| madera_cuerpo | VARCHAR(50) | |
| madera_mastil | VARCHAR(50) | |
| madera_diapason | VARCHAR(50) | |
| tapa | VARCHAR(15) | |
| configuracion | VARCHAR(7) | |
| pastillas | VARCHAR(200) | |
| trastes | VARCHAR(3) | |
| nCuerdas | VARCHAR(2) | |
| tremolo | BOOL | |
| precio | FLOAT | |
| urlImagen | VARCHAR(255) | |
| url | VARCHAR(255) | |
| isActive | BOOL | |
| rotacion | FLOAT | |

| marca | | |
|---|---|---|
| id | int | PK |
| nombre | VARCHAR(20) | |
| seccion | VARCHAR(20) | |
| url_base | VARCHAR(255) | |
| url | VARCHAR(255) | |
| url_siguiente_pagina | VARCHAR(255) | |

# Script para crear la Base de Datos MySql

```
DROP DATABASE IF EXISTS guitar_catalog;
Create database guitar_catalog;
use guitar_catalog;

CREATE TABLE guitarra (
    id INT AUTO_INCREMENT PRIMARY KEY,
    idExterno VARCHAR(70),
    marca VARCHAR(15),
    modelo VARCHAR(70),
    forma VARCHAR(15),
    madera_cuerpo VARCHAR(50),
    madera_mastil VARCHAR(50),
    madera_diapason VARCHAR(50),
    tapa VARCHAR(15),
    configuracion VARCHAR(7),
    pastillas VARCHAR(200),
    trastes VARCHAR(3),
    nCuerdas VARCHAR(2),
    tremolo BOOL,
    precio FLOAT,
    urlImagen VARCHAR(255),
    url VARCHAR(255),
    isActive BOOL,
    rotacion FLOAT
);
CREATE TABLE marca(
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(20),
    seccion VARCHAR(20),
    url_base VARCHAR(255),
    url VARCHAR(255),
    url_siguiente_pagina VARCHAR(255)
);

DELIMITER $$

CREATE PROCEDURE InsertGuitarIfNotExists(
    IN p_idExterno VARCHAR(70),
    IN p_marca VARCHAR(15),
    IN p_modelo VARCHAR(70),
    IN p_forma VARCHAR(15),
    IN p_madera_cuerpo VARCHAR(50),
    IN p_madera_mastil VARCHAR(50),
    IN p_madera_diapason VARCHAR(50),
    IN p_tapa VARCHAR(15),
    IN p_configuracion VARCHAR(7),
    IN p_pastillas VARCHAR(200),
    IN p_trastes VARCHAR(3),
    IN p_nCuerdas VARCHAR(2),
    IN p_tremolo BOOL,
    IN p_precio FLOAT,
    IN p_urlImagen VARCHAR(255),
    IN p_url VARCHAR(255),
    IN p_rotacion FLOAT
)
BEGIN
    DECLARE guitarExists INT;

    SELECT COUNT(*) INTO guitarExists
    FROM guitarra
    WHERE idExterno = p_idExterno
      AND marca = p_marca
      AND modelo = p_modelo;

    IF guitarExists = 0 THEN
        INSERT INTO guitarra (
            idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
            configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion
        ) VALUES (
            p_idExterno, p_marca, p_modelo, p_forma, p_madera_cuerpo, p_madera_mastil, p_madera_diapason, p_tapa,
            p_configuracion, p_pastillas, p_trastes, p_nCuerdas, p_tremolo, p_precio, p_urlImagen, p_url,
p_rotacion
        );
    END IF;
END $$
DELIMITER ;

CREATE USER 'scraper'@'localhost' IDENTIFIED BY '1234';
CREATE USER 'api'@'localhost' IDENTIFIED BY '1234';
GRANT SELECT ON guitar_catalog.guitarra TO 'api'@'localhost';
GRANT SELECT ON *.* TO 'scraper'@'localhost';
GRANT EXECUTE ON PROCEDURE guitar_catalog.InsertGuitarIfNotExists TO 'scraper'@'localhost';

INSERT INTO marca(nombre, seccion, url_base, url, url_siguiente_pagina) values
("Strandberg","all","https://strandbergguitars.com/eu/","https://strandbergguitars.com/eu/product-category/
guitar/", null);
INSERT INTO marca(nombre, seccion, url_base, url) values('Schecter', 'Demon', 'https://www.schecterguitars.com/',
'https://www.schecterguitars.com/api/items?c=765602&commercecategoryurl=%2Fguitars
%2Fdemon&country=US&currency=USD&fieldset=search&include=facets&language=en&limit=24&n=2&offset=0&pricelevel=5&so
rt=custitem1%3Aasc');
INSERT INTO marca(nombre, seccion, url_base, url) values('Schecter', 'Apocalypse',
'https://www.schecterguitars.com/', 'https://www.schecterguitars.com/api/items?c=765602&commercecategoryurl=
```

```
%2Fguitars
%2Fapocalypse&country=US&currency=USD&fieldset=search&include=facets&language=en&limit=24&n=2&offset=0&pricelevel
=5&sort=custitem1%3Aasc');
INSERT INTO marca(nombre, seccion, url_base, url) values('Schecter', 'Reaper Elite',
'https://www.schecterguitars.com/', 'https://www.schecterguitars.com/api/items?c=765602&commercecategoryurl=
%2Fguitars%2Freaper-
elite&country=US&currency=USD&fieldset=search&include=facets&language=en&limit=24&n=2&offset=0&pricelevel=5&sort=
custitem1%3Aasc');

INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('DCR ML ET FL
SBG', 'Dean Guitars', 'SELECT FLUENCE BLACK SATIN', 'ML', 'Eastern Mahogany', '3 Piece Eastern Mahogany',
'Ebony', null, 'H-H', 'Fishman Fluence Modern', '22', '6', false, 1799,
'https://armadillo.sirv.com/dg-wp/2023/04/ML10.png', 'https://www.deanguitars.com/product-mlselflbks/', -90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('ML SEL FM TBZ',
'Dean Guitars', 'SELECT FLAME TOP TRANS BRAZILIA', 'ML', 'Eastern Mahogany', '3 Piece Eastern Mahogany', 'Ebony',
null, 'H-H', 'Seymour Duncan TB-5 Custom Zebra / Seymour Duncan APH-1 Custom Zebra', '22', '6', false, 1199.00,
'https://www.deanguitars.com/wp-content/uploads/2024/03/ML-SEL-FM-TBZ-Header.png',
'https://www.deanguitars.com/product-mlselfmtbz/', -90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('ML 79 BBF', 'Dean
Guitars', '79 BLACK BLUE FADE', 'ML', 'Eastern Mahogany', '3 Piece Eastern Mahogany', 'Indian Rosewood', null,
'H-H', 'DMT Series Time Capsule BK/CR / DMT Series Time Capsule BK/CR', '22', '6', false, 569.00,
'https://www.deanguitars.com/wp-content/uploads/2024/03/ML-79-BBF-Header.png',
'https://www.deanguitars.com/product-ml79bbf/', -90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('USAANNIHILATOR',
'Dean Guitars', 'USA DOYLE WOLFGANG VON FRANKENSTEIN SIGNATURE ANNIHILATOR', 'Annihilator', 'Mahogany',
'Mahogany', 'Richlite', null, 'H', 'Von Frankestein Monster', '27', '6', true, 9999.00,
'https://armadillo.sirv.com/dg-wp/2023/04/annihilator.png?
w=2048&h=715&scale.option=fill&cw=2048&ch=715&cx=center&cy=center', 'https://www.deanguitars.com/annihilator/', -
90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion)
values('A2.7CANIBALISMO+', 'Solar Guitars', 'A2.7CANIBALISMO+', 'Stratocaster', 'Sungkai', 'Sungkai', 'Ebony',
null, 'H-H', 'Duncan Solar+', '24', '7', false, 849.00,
'https://cdn.solar-guitars.com/wp-content/uploads/2022/04/20220428155634-A2.7Canibalismo20-20FRONT20HORIZONTAL-
2048x672.png', 'https://www.solar-guitars.com/product/a2-7canibalismo/', -90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('A2.7C', 'Solar
Guitars', 'A2.7C - CARBON BLACK MATTE', 'Stratocaster', 'Mahogany', 'Maple', 'Ebony', null, 'H-H', 'Duncan
Solar', '24', '7', false, 799.00, 'https://cdn.solar-guitars.com/wp-content/uploads/2019/02/A2.7C-FRONT-
HORIZONTAL-1.png', 'https://www.solar-guitars.com/product/a2-7c-carbon-black-matte/', -90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('EC-1000', 'ESP',
'EC-1000 Vintage Black', 'Single Cut', 'Mahogany', '3Pc Mahogany', 'Macassar Ebony', null, 'H-H', 'EMG 81 / EMG
60', '24', '6', false, 1099.00, 'https://cdn.connectsites.net/user_files/esp/product_images/000/027/299/
original.png?1566681442', 'https://www.espguitars.com/products/9520-ec-1000-vb', -90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('TE-201', 'ESP',
'TE-201 Black Satin', 'Telecaster', 'Mahogany', '3Pc Maple', 'Roasted Jatoba', null, 'H', 'ESP LH-150B', '24',
'6', false, 499.00, 'https://cdn.connectsites.net/user_files/esp/product_images/000/032/201/original.png?
1642176679', 'https://www.espguitars.com/products/27450-te-201', -90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('TE-1000', 'ESP',
'TE-1000 Snow White', 'Telecaster', 'Mahogany', '3Pc Maple', 'Macassar Ebony', null, 'H-H', 'Fishman Fluence
Modern Humbucker Ceramic / Fiishman Fluence Modern Humbucker Alnic', '24', '6', false, 1299.00,
'https://cdn.connectsites.net/user_files/esp/product_images/000/032/195/original.png?1642176671',
'https://www.espguitars.com/products/27448-te-1000', -90.0);
INSERT INTO guitarra (idExterno, marca, modelo, forma, madera_cuerpo, madera_mastil, madera_diapason, tapa,
configuracion, pastillas, trastes, nCuerdas, tremolo, precio, urlImagen, url, rotacion) values('VIPER-1000
BARITONE', 'ESP', 'VIPER-1000 Baritone Black Satin', 'Double Cut', 'Mahogany', '3Pc Mahogany', 'Macassar Ebony',
null, 'H-H', 'EMG 81 / EMG 60', '24', '6', false, 1199.00,
'https://cdn.connectsites.net/user_files/esp/product_images/000/032/210/original.png?1642176694',
'https://www.espguitars.com/products/27453-viper-1000-baritone', -90.0);
```

Para poder visualizarlo mejor, se puede acceder también al archivo adjunto.

# 5. ESPECIFICACIONES DEL SISTEMA

## Tecnologías

Las tecnologías utilizadas en este proyecto son:

- Mysql para la base de datos.
- Python para los scraping.
- .NET Core 8 junto a C# para la API web.
- Android Studio.

## librerías

Las librerías utilizadas en este proyecto son:

- selenium para Python.
- undetected_chromedriver para Python.
- setuptools para Python.
- disutils para Python.
- glide para Android Studio.
- Gson para Android Studio.
- room para Android Studio
- lottie para Android Studio

## Configuración

Para configurar es proyecto, es necesario instanciar la API web en un servidor web, por ejemplo IIS e instalar los *conjuntos de hospedaje de .NET Core*.

Para usar los programas de scraping es necesario instalar Python junto a las librerias necesarias.

Para que la API pueda llamar a la base de datos y los programas de scraping puedan guardar datos en la base de datos, es necesario crearla con el script adjunto.

## Requisitos de hardware

El servidor sería el encargado de la base de datos, los scraping y la API, por lo que se requeriría un equipo windows 10 con acceso a internet de alta velocidad y como mínimo los siguientes componentes: Intel Core i7 8700, 16GB RAM., 500GB de almacenamiento.

# 6. ESPECIFICACIONES DEL SOFTWARE

## Operaciones

- Operaciones de recopilación de datos.
  - Estas operaciones se realizan en los scraping de Python.
- Operaciones de inserción de datos en la base de datos.
  - Estas operaciones se realizan en la clase principal de Python.
- Operaciones de facilitación de datos.
  - Estas operaciones se realizan en la API web.
- Operaciones de obtención de datos.
  - Estas operaciones se realizan en la aplicación Android.

## Interfaces

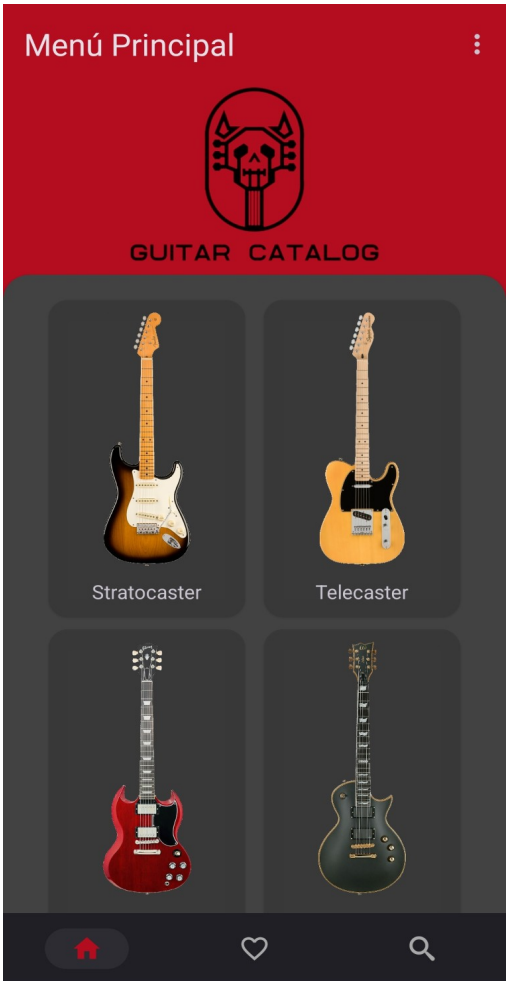
### Menú principal

Al iniciar la aplicación saldrá esta pantalla, donde se podrá elegir el tipo de forma deseada para listar.

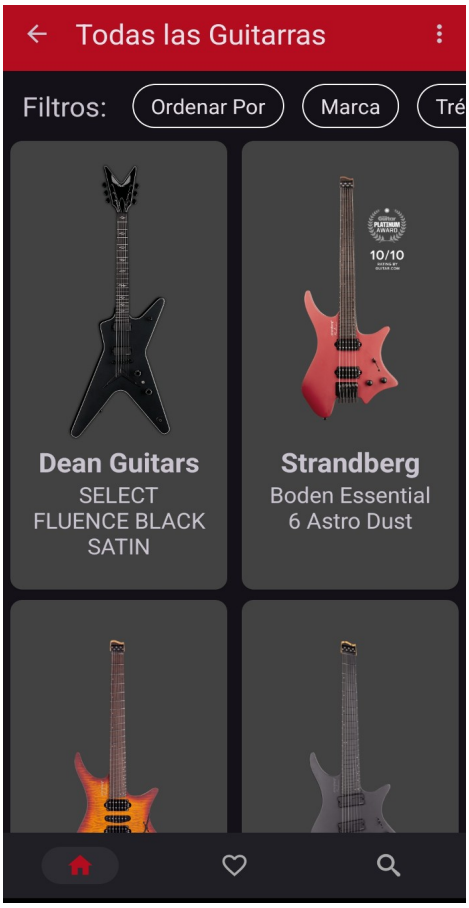
Hay un total de 6 formas a elegir:

- Stratocaster
- Telecaster
- Double Cut
- Single Cut
- Otras Formas
- Todas las Guitarras

Desde cualquier interfaz se podrá acceder a las opciones pulsando en los 3 botones de arriba a la derecha.



## Listado de guitarras




Una vez seleccionada la forma deseada, se llamará a la API para cargar las guitarras y aparecerá un listado con las guitarras cuya forma se corresponda con la deseada.

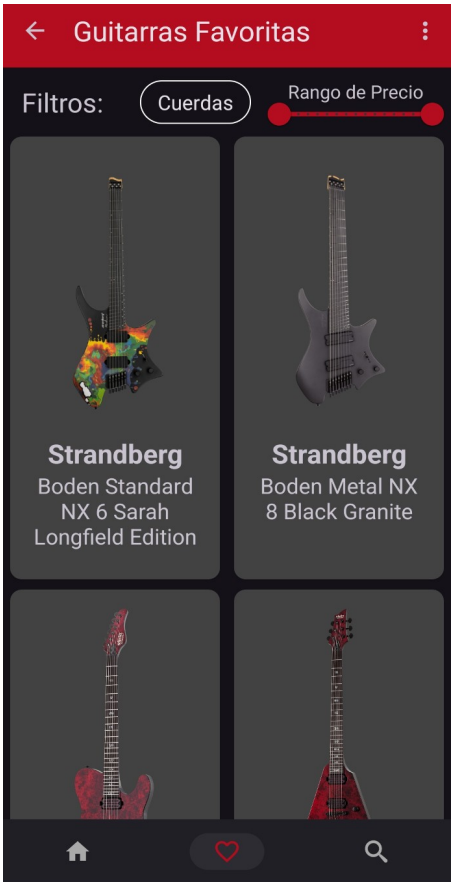
Se puede filtrar el listado por marca, trémolo, número de cuerdas, precio y ordenar por marca y precio.

Al pulsar sobre una guitarra, se podrá acceder a los detalles de la misma.

Esta interfaz es la misma que se usa para el listado de favoritos y la interfaz de búsqueda.

Si no existen guitarras con los filtros introducidos aparecerá un mensaje indicándolo y si no hay conexión a internet mostrará un mensaje de error distinto.

## Guitarras Favoritas

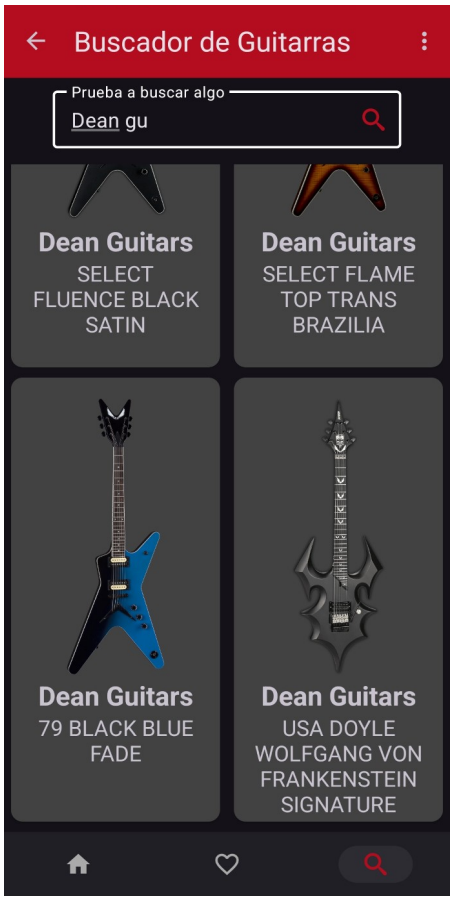



Al entrar a esta interfaz, se llamará a la base de datos interna del móvil para sacar un listado de ids de guitarra guardados y se llamará a la API para sacar los datos de las guitarras que se correspondan con esos ids.

Si no hay guitarras guardadas como favoritos aparecerá un mensaje indicándolo y si no hay conexión a internet mostrará un mensaje de error distinto.

## Buscador de guitarras

Se podrá introducir la marca o modelo en el buscador y aparecerán todas las coincidencias

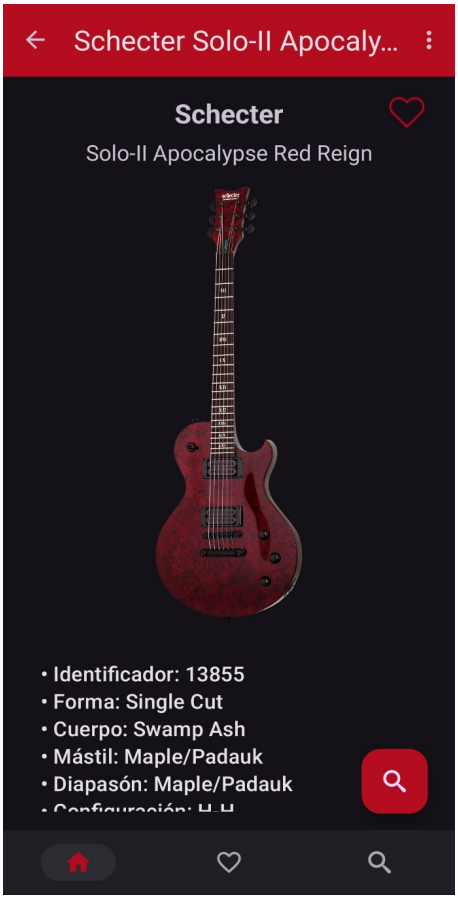



## Detalles

Tras pulsar el cualquier guitarras del listado, se accederán a los detalles de la guitarra, pudiendo hacer scroll en los detalles.

En la parte superior derecha se puede añadir a favoritos cualquier guitarra.

Si se pulsa en la lupa que aparece en la parte inferior derecha se podrá abrir la página web oficial de la guitarra.



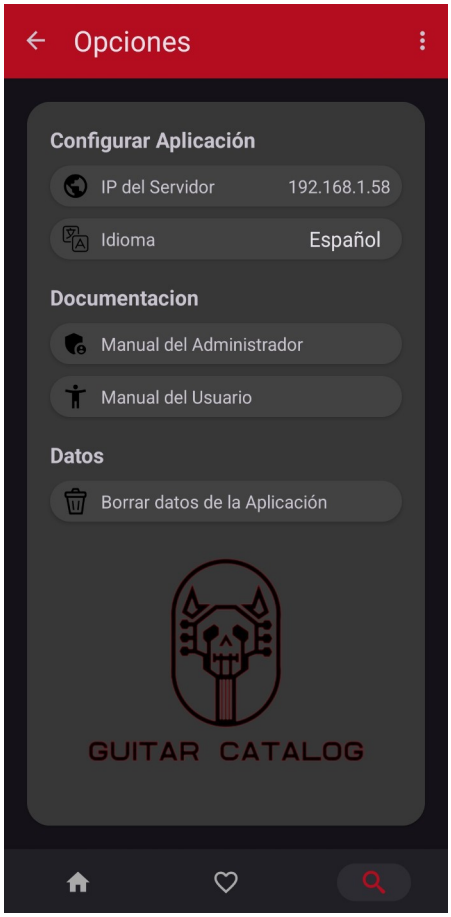
## Opciones

Se puede establecer en qué ip se encuentra el servidor para llamarlo.

Se puede configurar el idioma de la aplicación, pudiendo elegir entre español, inglés y alemán.

Se puede descargar el manual de administrador y el manual de usuario desde sus correspondientes botones.

Finalmente se pueden borrar todos los datos de la aplicación dejándola en su versión de fábrica.



# 7. CÓDIGO FUENTE RELEVANTE

Método para obtener el listado de guitarras:

```
suspend fun getGuitars(ip:String, ids: String?, shape:String?, brand:String?, tremolo:String?, strings:String?,
price:String?, patternToSearch: String?): Flow<List<Guitar>> = callbackFlow {
            var thereIsParams = false
            if (!isHostAccessible(ip))
                trySend(emptyList())
            try {
                var urlTmp = "https://$ip/list"
                if(patternToSearch != null)
                    urlTmp += "?patternToSearch=$patternToSearch"
                else {
                    if (ids != null) {
                        urlTmp += "?favs=$ids"
                        thereIsParams = true
                    }
                    if (shape != null) {
                        urlTmp += "?shape=$shape"
                        thereIsParams = true
                    }
                    if (brand != null) {
                        urlTmp += if (thereIsParams)
                            "&brand=$brand"
                        else
                            "?brand=$brand"
                        thereIsParams = true
                    }
                    if (tremolo != null) {
                        urlTmp += if (thereIsParams)
                            "&tremolo=${tremolo.lowercase().replace("si", "yes")}"
                        else
                            "?tremolo=${tremolo.lowercase().replace("si", "yes")}"
                        thereIsParams = true
                    }
                    if (strings != null) {
                        urlTmp += if (thereIsParams)
                            "&strings=$strings"
                        else
                            "?strings=$strings"
                        thereIsParams = true
                    }
                    if (price != null) {
                        urlTmp += if (thereIsParams)
                            "&price=$price"
                        else
                            "?price=$price"
                        thereIsParams = true
                    }
                }
                val url = URL(urlTmp)
                val connection = url.openConnection() as HttpURLConnection

                if (connection is HttpsURLConnection) {
                    val trustAllCerts = arrayOf<TrustManager>(object : X509TrustManager {
                        override fun checkClientTrusted(
                            chain: Array<out java.security.cert.X509Certificate>?,
                            authType: String?
                        ) {
                        }
                        override fun checkServerTrusted(
                            chain: Array<out java.security.cert.X509Certificate>?,
                            authType: String?
                        ) {
                        }
                        override fun getAcceptedIssuers(): Array<java.security.cert.X509Certificate>? =
                            null
                    })
                    val sslContext = SSLContext.getInstance("TLS")
                    sslContext.init(null, trustAllCerts, java.security.SecureRandom())
                    connection.sslSocketFactory = sslContext.socketFactory
                    connection.hostnameVerifier = javax.net.ssl.HostnameVerifier { _, _ -> true }
                }
                CoroutineScope(Dispatchers.IO).launch {
                    try {
                        connection.requestMethod = "GET"
                        val responseCode = connection.responseCode
                        if (responseCode == HttpURLConnection.HTTP_OK) {
                            val bufferedReader =
                                BufferedReader(InputStreamReader(connection.inputStream))
                            val response = StringBuffer()
                            var inputLine: String?
                            while (bufferedReader.readLine().also { inputLine = it } != null) {
                                response.append(inputLine)
                            }
                            bufferedReader.close()
                            val gson = Gson()
```

```
                            val guitars =
                                gson.fromJson(response.toString(), Array<Guitar>::class.java)
                                    .toList()
                            trySend(guitars).isSuccess
                        } else {
                            trySend(emptyList()).isSuccess
                        }
                    } catch (e: Exception) {
                        trySend(emptyList()).isSuccess
                        close(e)
                    } finally {
                        connection.disconnect()
                    }
                }
            } catch (e: Exception) {
                trySend(emptyList()).isSuccess
            }
            awaitClose()
        }
```

Método para cargar las marcas de guitarra para el spinner de marcas:

```
suspend fun getBrands(ip:String): List<String>? {

            if (!isHostAccessible(ip)){
                return null
            }
            val url = "https://$ip/brand"

            return withContext(Dispatchers.IO) {
                val url = URL(url)
                val connection = url.openConnection() as HttpURLConnection

                if (connection is HttpsURLConnection) {
                    val trustAllCerts = arrayOf<TrustManager>(object : X509TrustManager {
                        override fun checkClientTrusted(
                            chain: Array<out java.security.cert.X509Certificate>?,
                            authType: String?
                        ) {
                        }
                        override fun checkServerTrusted(
                            chain: Array<out java.security.cert.X509Certificate>?,
                            authType: String?
                        ) {
                        }
                        override fun getAcceptedIssuers(): Array<java.security.cert.X509Certificate>? =
                            null
                    })
                    val sslContext = SSLContext.getInstance("TLS")
                    sslContext.init(null, trustAllCerts, java.security.SecureRandom())
                    connection.sslSocketFactory = sslContext.socketFactory
                    connection.hostnameVerifier = javax.net.ssl.HostnameVerifier { _, _ -> true }
                }
                try {
                    connection.requestMethod = "GET"
                    val responseCode = connection.responseCode
                    if (responseCode == HttpURLConnection.HTTP_OK) {
                        val bufferedReader =
                            BufferedReader(InputStreamReader(connection.inputStream))
                        val response = StringBuffer()
                        var inputLine: String?
                        while (bufferedReader.readLine().also { inputLine = it } != null) {
                            response.append(inputLine)
                        }
                        bufferedReader.close()
                        val gson = Gson()
                        var myType: Type = object : TypeToken<List<String>>() {}.type
                        val brands = gson.fromJson<List<String>>(response.toString(), myType)
                        brands
                    } else
                        null
                } finally {
                    connection.disconnect()
                }
            }
        }
```

Método para cargar los precios de las guitarras para el slider de filtros:

```
suspend fun getPriceRange(ip: String, ids: String?, shape:String?, brand:String?, tremolo:String?,
strings:String?): List<Float>? {
            var thereIsParams = false

            if (!isHostAccessible(ip)){
                return null
            }
            val url = "https://$ip/price"

            var urlTmp = url
            if (ids != null){
                urlTmp += "?favs=$ids"
                thereIsParams = true
            }
            if(shape != null){
                urlTmp += "?shape=$shape"
                thereIsParams = true
            }
            if(brand != null){
                urlTmp += if(thereIsParams)
                    "&brand=$brand"
                else
                    "?brand=$brand"
                thereIsParams = true
            }
            if(tremolo != null){
                urlTmp += if(thereIsParams)
                    "&tremolo=${tremolo.lowercase().replace("si", "yes")}"
                else
                    "?tremolo=${tremolo.lowercase().replace("si", "yes")}"
                thereIsParams = true
            }
            if(strings != null){
                urlTmp += if(thereIsParams)
                    "&strings=$strings"
                else
                    "?strings=$strings"
                thereIsParams = true
            }

            return withContext(Dispatchers.IO) {
                val url = URL(urlTmp)
                val connection = url.openConnection() as HttpURLConnection

                if (connection is HttpsURLConnection) {
                    val trustAllCerts = arrayOf<TrustManager>(object : X509TrustManager {
                        override fun checkClientTrusted(
                            chain: Array<out java.security.cert.X509Certificate>?,
                            authType: String?
                        ) {
                        }
                        override fun checkServerTrusted(
                            chain: Array<out java.security.cert.X509Certificate>?,
                            authType: String?
                        ) {
                        }
                        override fun getAcceptedIssuers(): Array<java.security.cert.X509Certificate>? =
                            null
                    })
                    val sslContext = SSLContext.getInstance("TLS")
                    sslContext.init(null, trustAllCerts, java.security.SecureRandom())
                    connection.sslSocketFactory = sslContext.socketFactory
                    connection.hostnameVerifier = javax.net.ssl.HostnameVerifier { _, _ -> true }
                }
                try {
                    connection.requestMethod = "GET"
                    val responseCode = connection.responseCode
                    if (responseCode == HttpURLConnection.HTTP_OK) {
                        val bufferedReader =
                            BufferedReader(InputStreamReader(connection.inputStream))
                        val response = StringBuffer()
                        var inputLine: String?
                        while (bufferedReader.readLine().also { inputLine = it } != null) {
                            response.append(inputLine)
                        }
                        bufferedReader.close()
                        val gson = Gson()
                        var myType: Type = object : TypeToken<List<Float>>() {}.type
                        val priceRange = gson.fromJson<List<Float>>(response.toString(), myType)
                        priceRange
                    } else
                        null
                } finally {
                    connection.disconnect()
                }
            }
        }
```

Método para obtener los datos de una guitarra:

```
suspend fun getGuitar(ip:String, id: Int): Guitar? {
            if (!isHostAccessible(ip))
                return null

            val urlGet = "https://$ip/get?id=$id"

            return withContext(Dispatchers.IO) {
                val url = URL(urlGet)
                val connection = url.openConnection() as HttpURLConnection

                if (connection is HttpsURLConnection) {
                    val trustAllCerts = arrayOf<TrustManager>(object : X509TrustManager {
                        override fun checkClientTrusted(
                            chain: Array<out java.security.cert.X509Certificate>?,
                            authType: String?
                        ) {
                        }
                        override fun checkServerTrusted(
                            chain: Array<out java.security.cert.X509Certificate>?,
                            authType: String?
                        ) {
                        }
                        override fun getAcceptedIssuers(): Array<java.security.cert.X509Certificate>? =
                            null
                    })
                    val sslContext = SSLContext.getInstance("TLS")
                    sslContext.init(null, trustAllCerts, java.security.SecureRandom())
                    connection.sslSocketFactory = sslContext.socketFactory
                    connection.hostnameVerifier = javax.net.ssl.HostnameVerifier { _, _ -> true }
                }
                try {
                    connection.requestMethod = "GET"
                    val responseCode = connection.responseCode
                    if (responseCode == HttpURLConnection.HTTP_OK) {
                        val bufferedReader =
                            BufferedReader(InputStreamReader(connection.inputStream))
                        val response = StringBuffer()
                        var inputLine: String?
                        while (bufferedReader.readLine().also { inputLine = it } != null) {
                            response.append(inputLine)
                        }
                        bufferedReader.close()
                        val gson = Gson()
                        val guitar = gson.fromJson(response.toString(), Guitar::class.java)
                        guitar
                    } else
                        null

                } finally {
                    connection.disconnect()
                }
            }
        }
```

Método para saber si un host es accesible:

```
suspend fun isHostAccessible(host: String): Boolean {
            return withContext(Dispatchers.IO) {
                try {
                    val address = InetAddress.getByName(host)
                    address.isReachable(5000)
                } catch (e: Exception) {
                    e.printStackTrace()
                    false
                }
            }
        }
```

Clase para cargar fotos desde una url:

```
class ImageLoader (private val context: Context, private val imageView: ImageView, private val rotation:Float?) {

    fun load(imageUrl: String) {
        if (rotation == null) {
            Glide.with(context)
                .load(imageUrl)
                .into(imageView)
        } else {
            val rotationTransformation = object : BitmapTransformation() {
                override fun transform(pool: BitmapPool, toTransform: Bitmap, outWidth: Int, outHeight: Int):
Bitmap {
                    return rotateImage(toTransform, rotation)
                }
                override fun updateDiskCacheKey(messageDigest: MessageDigest) {}
```

```
            }
            Glide.with(context)
                .load(imageUrl)
                .transform(rotationTransformation)
                .into(imageView)
        }
    }

    private fun rotateImage(source: Bitmap, angle: Float): Bitmap {
        val matrix = android.graphics.Matrix()
        matrix.postRotate(angle)
        return Bitmap.createBitmap(source, 0, 0, source.width, source.height, matrix, true)
    }
}
```

Método para devolver listado de guitarras:

```
[HttpGet]
public ActionResult Get(string? favs, string? shape, string? brand, string? tremolo, string? strings, string?
price, string? patternToSearch)
{
    string connectionString = "Server=localhost;Database=guitar_catalog;Uid=api;Pwd=1234;";
    try
    {
        List<Dictionary<string, object>> guitars = new List<Dictionary<string, object>>();
        using (MySqlConnection connection = new MySqlConnection(connectionString))
        {
            connection.Open();
            string sql;
            if (patternToSearch != null)
                sql = "SELECT * from guitarra WHERE CONCAT(marca, ' ', modelo) like '%" + patternToSearch + "%'";
            else
            {
                if (favs != null)
                {
                    sql = "SELECT * FROM guitarra where id in (" + favs + ")";
                }
                else if (shape != null)
                {
                    string shapes = "('Stratocaster', 'Telecaster', 'Double Cut', 'Single Cut')";
                    if (shapes.Contains(shape))
                        sql = "SELECT * FROM guitarra where forma = '" + shape + "'";
                    else
                        sql = "SELECT * FROM guitarra where forma not in " + shapes;
                }
                else
                    sql = "SELECT * FROM guitarra where forma <> 'notAShape'";
                if (!string.IsNullOrEmpty(brand))
                    sql += " AND marca = '" + brand + "'";
                if (!string.IsNullOrEmpty(tremolo))
                    sql += " AND tremolo = " + tremolo.Replace("yes", "1").Replace("no", "0");
                if (!string.IsNullOrEmpty(strings))
                    sql += " AND nCuerdas = '" + strings + "'";
                if (!string.IsNullOrEmpty(price))
                    sql += " AND precio BETWEEN " + price;
            }
            using (MySqlCommand command = new MySqlCommand(sql, connection))
            {
                using (MySqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        Dictionary<string, object> guitar = new Dictionary<string, object>();

                        for (int i = 0; i < reader.FieldCount; i++)
                        {
                            guitar[reader.GetName(i)] = reader.GetValue(i);
                        }

                        guitars.Add(guitar);
                    }
                }
            }
        }
        string json = JsonConvert.SerializeObject(guitars);
        return Ok(json);
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Error: {ex.Message}");
    }
}
```

Método para devolver datos de una guitarra:

```
[HttpGet]
public ActionResult Get(int id)
{
    try
    {
        string connectionString = "Server=localhost;Database=guitar_catalog;Uid=api;Pwd=1234;";
        Dictionary<string, object> guitar = new Dictionary<string, object>();
        using (MySqlConnection connection = new MySqlConnection(connectionString))
        {
            connection.Open();
            string sql = $"SELECT * FROM guitarra where id = {id}";
            using (MySqlCommand command = new MySqlCommand(sql, connection))
            {
                using (MySqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        for (int i = 0; i < reader.FieldCount; i++)
                        {
                            guitar[reader.GetName(i)] = reader.GetValue(i);
                        }

                    }
                }
            }
        }
        string json = JsonConvert.SerializeObject(guitar);
        return Ok(json);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}
```

Método para devolver el listado de marcas:

```
HttpGet]
public ActionResult Get()
{
    string connectionString = "Server=localhost;Database=guitar_catalog;Uid=api;Pwd=1234;";
    try
    {
        List<string> brands = new List<string>();

        using (MySqlConnection connection = new MySqlConnection(connectionString))
        {
            connection.Open();

            string sql = "Select distinct marca from guitarra";
            using (MySqlCommand command = new MySqlCommand(sql, connection))
            {
                using (MySqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                            string brand  = reader.GetString(0);

                        brands.Add(brand);
                    }
                }
            }
        }
        string json = JsonConvert.SerializeObject(brands);
        return Ok(json);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}
```

Método para devolver el listado de precios:

```
[HttpGet]
public ActionResult Get(string? favs, string? shape, string? brand, string? tremolo, string? strings, string?
price)
{
    string connectionString = "Server=localhost;Database=guitar_catalog;Uid=api;Pwd=1234;";
    try
    {
        List<int> prices = new List<int>();
        using (MySqlConnection connection = new MySqlConnection(connectionString))
        {
            connection.Open();
            string sql = "";
            if (shape != null)
            {
                string shapes = "('Stratocaster', 'Telecaster', 'Double Cut', 'Single Cut')";
                if (shapes.Contains(shape))
                    sql = "SELECT min(precio) as min, max(precio) as max from guitarra where forma = '" + shape +
"'";
                else
                    sql = "SELECT min(precio) as min, max(precio) as max from guitarra where forma not in " +
shapes;
            }
            else
                sql = "SELECT min(precio) as min, max(precio) as max from guitarra where forma <> 'notAShape'";
            if (!string.IsNullOrEmpty(brand))
                sql += " AND marca = '" + brand + "'";
            if (!string.IsNullOrEmpty(tremolo))
                sql += " AND tremolo = " + tremolo.Replace("yes", "1").Replace("no", "0");
            if (!string.IsNullOrEmpty(strings))
                sql += " AND nCuerdas = '" + strings + "'";
            using (MySqlCommand command = new MySqlCommand(sql, connection))
            {
                using (MySqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        prices.Add(reader.IsDBNull(reader.GetOrdinal("min")) ? 0 : reader.GetInt32("min"));
                        prices.Add(reader.IsDBNull(reader.GetOrdinal("max")) ? 3000 : reader.GetInt32("max"));
                    }
                }
            }
        }
        string json = JsonConvert.SerializeObject(prices);
        return Ok(json);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}
```

Forma de insertar datos desde Python:

```
sql_insert = """
    CALL InsertGuitarIfNotExists(
        %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s
    )
"""
datos_insert = (
    guitar[1], guitar[2], guitar[3], guitar[4], guitar[5], guitar[6],
    guitar[7], guitar[8], guitar[9], guitar[10], guitar[11], guitar[12],
    guitar[13], guitar[14], guitar[15], guitar[16], guitar[17]
)
```

# 8. CONCLUSIONES DEL PROYECTO

Este proyecto se extenderá con más scraps en Python, más mejoras y más servicios de instrumentos musicales, incluyendo bajos eléctricos, banjos y otros instrumentos.

# 9. APÉNDICES

## Formato del Json obtenido en la API


```
[
   {
      "id": 1,
      "idExterno": "DCR ML ET FL SBG",
      "marca": "Dean Guitars",
      "modelo": "SELECT FLUENCE BLACK SATIN",
      "forma": "ML",
      "madera_cuerpo": "Eastern Mahogany",
      "madera_mastil": "3 Piece Eastern Mahogany",
      "madera_diapason": "Ebony",
      "tapa": null,
      "configuracion": "H-H",
      "pastillas": "Fishman Fluence Modern",
      "trastes": "22",
      "nCuerdas": "6",
      "tremolo": false,
      "precio": 1799.0,
      "urlImagen": "https://armadillo.sirv.com/dg-wp/2023/04/ML10.png",
      "url": "https://www.deanguitars.com/product-mlselflbks/",
      "isActive": null,
      "rotacion": -90.0
   }
]
```


# 10.BIBLIOGRAFÍA


Instalar Asp Net Core para IIS

Llamar a una API desde Kotlin

Llamar a una API desde Kotlin

Cargar imágenes con Glide

Usar MediaPlayer

Iconos usados

GitHub del Proyecto