🎵 Bot de Discord para Música - Guía de Configuración

📋 Requisitos Previos

- **Node.js** versión 16 o superior
- **FFmpeg** instalado en tu sistema
- **Cuenta de Discord Developer**
- **Cuenta de Spotify Developer** (opcional pero recomendado)

🚀 Configuración Paso a Paso

1. Crear el Bot en Discord

1. Ve a Discord Developer Portal
2. Haz clic en "New Application"
3. Nombra tu aplicación (ej: "Music Bot")
4. Ve a la pestaña "Bot"
5. Haz clic en "Add Bot"
6. Copia el **Token** (lo necesitarás para el archivo .env)
7. En la sección "Privileged Gateway Intents", activa:
   - Message Content Intent
   - Server Members Intent (opcional)

2. Obtener IDs Necesarios

Client ID:

- En Discord Developer Portal, ve a "General Information"
- Copia el "Application ID"

Guild ID (opcional):

- En Discord, activa el "Modo Desarrollador" en Configuración > Avanzado
- Haz clic derecho en tu servidor y selecciona "Copiar ID"

3. Configurar Permisos del Bot

1. Ve a "OAuth2" > "URL Generator"
2. Selecciona scopes:
   - `bot`
   - `applications.commands`

3. Selecciona permisos:
   - `Send Messages`
   - `Use Slash Commands`
   - `Connect`
   - `Speak`
   - `Use Voice Activity`
   - `Embed Links`
   - `Read Message History`
4. Copia la URL generada e invita el bot a tu servidor

4. Configurar Spotify API (Opcional)

1. Ve a Spotify Developer Dashboard
2. Haz clic en "Create an App"
3. Completa el formulario:
   - App name: "Discord Music Bot"
   - App description: "Bot para reproducir música"
   - Redirect URI: `http://localhost:3000/callback`
4. Copia el **Client ID** y **Client Secret**

5. Instalar FFmpeg

Windows:

1. Descarga desde FFmpeg oficial
2. Extrae el archivo y agrega la carpeta `bin` al PATH del sistema

macOS:

```bash
brew install ffmpeg
```

Linux (Ubuntu/Debian):

```bash
sudo apt update
sudo apt install ffmpeg
```

6. Configurar el Proyecto

1. **Clonar/Descargar el código:**

bash

```
mkdir discord-music-bot
cd discord-music-bot
```

2. **Instalar dependencias:**

bash

```
npm install
```

3. **Crear archivo .env:**

bash

```
cp .env.example .env
```

4. **Editar el archivo .env:**

env

```
DISCORD_TOKEN=tu_token_de_discord_aqui
CLIENT_ID=tu_client_id_aqui
GUILD_ID=tu_guild_id_aqui_opcional
SPOTIFY_CLIENT_ID=tu_spotify_client_id_aqui
SPOTIFY_CLIENT_SECRET=tu_spotify_client_secret_aqui
```

7. Ejecutar el Bot

Desarrollo:

bash

```
npm run dev
```

Producción:

bash

```
npm start
```

🎮 Comandos Disponibles

Comando	Descripción
`/play <query>`	Reproduce música de YouTube/Spotify
`/skip`	Salta la canción actual
`/queue`	Muestra la cola de reproducción
`/stop`	Detiene la música y limpia la cola
`/pause`	Pausa la reproducción
`/resume`	Reanuda la reproducción
`/nowplaying`	Muestra la canción actual
`/clear`	Limpia la cola de reproducción
`/leave`	Desconecta el bot del canal

🔧 Características del Bot

- ✅ Reproducción de YouTube (URLs y búsqueda)
- ✅ Integración con Spotify (convierte a YouTube)
- ✅ Sistema de cola de reproducción
- ✅ Comandos slash interactivos
- ✅ Embeds informativos
- ✅ Manejo de errores robusto
- ✅ Reconexión automática

🛠️ Solución de Problemas

Error: "Cannot find module 'node:events'"

- Asegúrate de usar Node.js 16 o superior

Error: "FFMPEG not found"

- Instala FFmpeg y asegúrate de que esté en el PATH

Error: "Invalid token"

- Verifica que el token en .env sea correcto
- Asegúrate de que no haya espacios extra

Error: "Missing permissions"

- Verifica los permisos del bot en Discord
- Asegúrate de que tenga acceso al canal de voz

El bot no responde a comandos

- Verifica que los comandos slash estén registrados
- Reinicia el bot si es necesario

📦 Dependencias Principales

- **discord.js**: Librería principal para Discord
- **@discordjs/voice**: Para funcionalidad de audio
- **ytdl-core**: Para descargar audio de YouTube
- **yt-search**: Para buscar videos en YouTube
- **spotify-web-api-node**: Para integración con Spotify
- **ffmpeg-static**: Codificador de audio

🌟 Funcionalidades Adicionales

Para añadir más funcionalidades:

1. **Filtros de audio** (bass boost, nightcore, etc.)
2. **Playlists guardadas**
3. **Sistema de votación para skip**
4. **Lyrics display**
5. **Radio streaming**
6. **Controles de reproducción con botones**

🚀 Despliegue en Producción

Usando PM2:

bash

```
npm install -g pm2
pm2 start index.js --name "music-bot"
pm2 startup
pm2 save
```

Usando Docker:

dockerfile

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
CMD ["node", "index.js"]
```

📝 Notas Importantes

- El bot requiere estar en un canal de voz para funcionar
- Las URLs de Spotify se convierten automáticamente a YouTube
- El bot mantiene una cola por servidor
- Los comandos slash pueden tardar unos minutos en aparecer después del primer inicio
- FFmpeg es obligatorio para el procesamiento de audio

🔐 Seguridad y Mejores Prácticas

- **Nunca compartas tu token de Discord**
- Usa variables de entorno para datos sensibles
- Implementa rate limiting para evitar spam
- Considera usar un proceso manager como PM2 para producción
- Mantén las dependencias actualizadas

🎯 Optimizaciones de Rendimiento

Para mejorar el rendimiento:

1. **Caché de búsquedas:**

javascript

```
const searchCache = new Map();
// Implementar caché para búsquedas frecuentes
```

2. **Conexiones persistentes:**

javascript

```
// Mantener conexiones de voz activas por más tiempo
```

3. **Streaming optimizado:**

javascript

```javascript
// Configurar ytdl para mejor calidad de streaming
const streamOptions = {
    filter: 'audioonly',
    quality: 'highestaudio',
    highWaterMark: 1 << 25
};
```

🐛 Debugging

Activar logs detallados:

```javascript
// Agregar al inicio del archivo
const DEBUG = process.env.NODE_ENV === 'development';

if (DEBUG) {
    console.log('Modo debug activado');
}
```

Logs útiles para debugging:

```javascript
// En cada función importante
console.log(`[${new Date().toISOString()}] ${message}`);
```

🔄 Actualizaciones y Mantenimiento

Mantener el bot actualizado:

```bash
# Verificar actualizaciones
npm outdated

# Actualizar dependencias
npm update

# Actualizar discord.js específicamente
npm install discord.js@latest
```

Backup de configuración:

- Haz backup regular de tu archivo .env
- Documenta cualquier configuración personalizada

- Mantén un registro de cambios (changelog)

📊 Monitoreo y Métricas

Métricas importantes a monitorear:

- Uptime del bot
- Número de servidores conectados
- Canciones reproducidas por día
- Errores de reproducción
- Uso de memoria y CPU

Implementar logging básico:

javascript

```javascript
const fs = require('fs');
const logFile = 'bot-activity.log';

function log(message) {
    const timestamp = new Date().toISOString();
    const logEntry = `[${timestamp}] ${message}\n`;
    fs.appendFileSync(logFile, logEntry);
}
```

🌐 Extensiones Avanzadas

1. Sistema de Playlists

javascript

```javascript
// Estructura para playlists guardadas
const playlists = new Map();

// Comando para crear playlist
new SlashCommandBuilder()
    .setName('playlist')
    .setDescription('Gestiona playlists')
    .addSubcommand(subcommand =>
        subcommand
            .setName('create')
            .setDescription('Crea una nueva playlist')
            .addStringOption(option =>
                option.setName('name')
                    .setDescription('Nombre de la playlist')
                    .setRequired(true)
            )
    )
```

2. Sistema de Votación

javascript

```javascript
// Para skip con votación
const skipVotes = new Map();

async function handleSkipVote(interaction) {
    const guildId = interaction.guildId;
    const userId = interaction.user.id;
    const voiceChannel = interaction.member.voice.channel;

    if (!voiceChannel) {
        return interaction.reply('❌ Debes estar en el canal de voz para votar.');
    }

    const membersInVoice = voiceChannel.members.size - 1; // -1 para el bot
    const requiredVotes = Math.ceil(membersInVoice / 2);

    // Lógica de votación...
}
```

3. Filtros de Audio

javascript

```
// Filtros usando FFmpeg
const audioFilters = {
    bass: 'bass=g=10',
    nightcore: 'aresample=48000,asetrate=48000*1.25',
    '8d': 'apulsator=hz=0.125'
};
```

🎛️ Interfaz de Usuario Mejorada

Botones interactivos:

javascript

```
const { ButtonBuilder, ButtonStyle, ActionRowBuilder } = require('discord.js');

const playButton = new ButtonBuilder()
    .setCustomId('play_pause')
    .setLabel('⏯️')
    .setStyle(ButtonStyle.Primary);

const skipButton = new ButtonBuilder()
    .setCustomId('skip')
    .setLabel('⏭️')
    .setStyle(ButtonStyle.Secondary);

const row = new ActionRowBuilder()
    .addComponents(playButton, skipButton);
```

Menús desplegables:

javascript

```
const { SelectMenuBuilder } = require('discord.js');

const select = new SelectMenuBuilder()
    .setCustomId('queue_select')
    .setPlaceholder('Selecciona una canción de la cola')
    .addOptions(
        queueSongs.map((song, index) => ({
            label: song.title,
            description: song.artist,
            value: index.toString()
        }))
    );
```

🚀 Despliegue en Diferentes Plataformas

Heroku:

bash

```
# Instalar Heroku CLI
npm install -g heroku

# Crear aplicación
heroku create tu-music-bot

# Configurar variables de entorno
heroku config:set DISCORD_TOKEN=tu_token
heroku config:set SPOTIFY_CLIENT_ID=tu_client_id

# Agregar buildpack para FFmpeg
heroku buildpacks:add https://github.com/jonathanong/heroku-buildpack-ffmpeg-latest.git

# Desplegar
git push heroku main
```

Railway:

bash

```
# Instalar Railway CLI
npm install -g @railway/cli

# Inicializar proyecto
railway init

# Configurar variables
railway variables:set DISCORD_TOKEN=tu_token

# Desplegar
railway deploy
```

VPS/Servidor Dedicado:

bash

```
# Configurar como servicio systemd
sudo nano /etc/systemd/system/music-bot.service

[Unit]
Description=Discord Music Bot
After=network.target

[Service]
Type=simple
User=your-user
WorkingDirectory=/path/to/your/bot
ExecStart=/usr/bin/node index.js
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target

# Activar servicio
sudo systemctl enable music-bot.service
sudo systemctl start music-bot.service
```

🔧 Configuración Avanzada

Configuración de ytdl-core:

javascript

```
const ytdlOptions = {
    filter: 'audioonly',
    quality: 'highestaudio',
    highWaterMark: 1 << 25,
    dlChunkSize: 0,
    requestOptions: {
        headers: {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36'
        }
    }
};
```

Configuración de Discord.js:

javascript

```
const client = new Client({
    intents: [
        GatewayIntentBits.Guilds,
        GatewayIntentBits.GuildMessages,
        GatewayIntentBits.GuildVoiceStates,
        GatewayIntentBits.MessageContent
    ],
    presence: {
        activities: [{
            name: 'música 🎶',
            type: ActivityType.Listening
        }],
        status: 'online'
    }
});
```

🎶 Formatos de Audio Soportados

El bot puede reproducir:

- **YouTube**: Videos y streams en vivo
- **Spotify**: Tracks individuales (convertidos a YouTube)
- **SoundCloud**: Con configuración adicional
- **Archivos directos**: MP3, WAV, OGG (con URL directa)

💡 Consejos Adicionales

1. **Hosting gratuito**: Usa Heroku, Railway, o Glitch para pruebas
2. **Paid hosting**: Considera VPS para mejor rendimiento
3. **Sharding**: Para bots en muchos servidores (500+)
4. **Database**: Usa SQLite/PostgreSQL para persistencia
5. **Caching**: Implementa Redis para mejor rendimiento

🏆 Mejores Prácticas de Desarrollo

- Usa TypeScript para mejor tipado
- Implementa tests unitarios
- Usa linting (ESLint) para código consistente
- Documenta tu código con JSDoc
- Usa git hooks para automatizar tareas
- Implementa CI/CD pipeline

🤝 Contribución y Comunidad

- Crea issues para bugs y sugerencias
- Sigue las convenciones de código
- Prueba exhaustivamente antes de PR
- Mantén compatibilidad hacia atrás
- Documenta cambios significativos

¡Tu bot de música ya está listo para usar! 🎉