7. **Despliegue Final**
8. **Pruebas y Documentación**

1. Prerrequisitos

- **Herramientas Necesarias**:
  - Node.js v18+ y npm.
  - Docker y Docker Compose.
  - Cuenta en AWS.
  - Cuenta en Spotify for Developers.
  - PostgreSQL o MongoDB instalado localmente (para desarrollo).
  - Git.

2. Configuración Inicial del Proyecto

Estructura de Carpetas

bash Copy

```
music-player/
├── backend/           # Código del backend (Node.js)
│   ├── src/
│   │   ├── routes/    # Endpoints de la API
│   │   ├── models/    # Modelos de base de datos
│   │   └── ...
├── frontend/          # Código del frontend (React)
│   ├── public/
│   ├── src/
│   │   ├── components/ # Componentes React
│   │   └── ...
└── docker-compose.yml # Configuración de Docker
```

Inicializar Repositorio

bash Copy

```
bash                                                                Copy
└── docker-compose.yml # Configuración de Docker
```

Inicializar Repositorio

```
bash                                                                Copy

mkdir music-player && cd music-player
mkdir backend frontend
git init
```

3. Backend (Node.js + Express + Spotify API)

Paso 1: Inicializar el Backend

```
bash                                                                Copy

cd backend
npm init -y
npm install express dotenv cors axios mongoose jsonwebtoken passport-spotify
```

Paso 2: Configurar la API de Spotify

1. **Registrar una App en Spotify**:
   - Ve a Spotify Developer Dashboard.
   - Crea una nueva app y obtén el **Client ID** y **Client Secret**.
   - Agrega `http://localhost:3000/callback` como Redirect URI.
2. **Archivo** `.env` :

```
env                                                                 Copy

SPOTIFY_CLIENT_ID=tu_client_id
SPOTIFY_CLIENT_SECRET=tu_client_secret
SPOTIFY_CALLBACK_URL=http://localhost:3000/callback
JWT_SECRET=mi_secreto_jwt
DATABASE_URL=mongodb://localhost:27017/musicdb
```

env Copy

Paso 3: Crear Autenticación con Spotify OAuth

- **Código de Autenticación** (`backend/src/auth/spotify.js`):

javascript Copy

```
const passport = require('passport');
const SpotifyStrategy = require('passport-spotify').Strategy;

passport.use(
  new SpotifyStrategy(
    {
      clientID: process.env.SPOTIFY_CLIENT_ID,
      clientSecret: process.env.SPOTIFY_CLIENT_SECRET,
      callbackURL: process.env.SPOTIFY_CALLBACK_URL,
    },
    (accessToken, refreshToken, expires_in, profile, done) => {
      // Guardar usuario en la base de datos
      return done(null, { accessToken, profile });
    }
  )
);
```

Paso 4: Endpoints de la API

- **Login con Spotify** (`backend/src/routes/auth.js`):

javascript Copy

```
const express = require('express');
const router = express.Router();
const passport = require('passport');

router.get('/spotify', passport.authenticate('spotify', {
  scope: ['user-read-email', 'user-read-private', 'streaming'],
}));

router.get('/spotify/callback',
  passport.authenticate('spotify', { failureRedirect: '/login' }),
  (req, res) => {
```

Message DeepSeek

DeepThink (R1) Search

- **Login con Spotify** (`backend/src/routes/auth.js`):

javascript Copy

```
const express = require('express');
const router = express.Router();
const passport = require('passport');

router.get('/spotify', passport.authenticate('spotify', {
  scope: ['user-read-email', 'user-read-private', 'streaming'],
}));

router.get('/spotify/callback',
  passport.authenticate('spotify', { failureRedirect: '/login' }),
  (req, res) => {
    // Redirigir al frontend con el token
    res.redirect(`http://localhost:3000/?token=${req.user.accessToken}`);
  }
);
```

- **Buscar Canciones en Spotify** (`backend/src/routes/spotify.js`):

javascript Copy

```
router.get('/search', async (req, res) => {
  const { q } = req.query;
  const response = await axios.get('https://api.spotify.com/v1/search', {
    headers: { 'Authorization': `Bearer ${req.user.accessToken}` },
    params: { q, type: 'track', limit: 10 }
  });
  res.json(response.data.tracks.items);
});
```

4. Frontend (React.js + Reproductor)

Paso 1: Inicializar React

bash Copy

```
cd frontend
npx create-react-app .
npm install react-router-dom axios react-icons react-player @reduxjs/toolkit
```

4. Frontend (React.js + Reproductor)

Paso 1: Inicializar React

bash Copy

```bash
cd frontend
npx create-react-app .
npm install react-router-dom axios react-icons react-player @reduxjs/toolkit
```

Paso 2: Configurar Redux para Estado Global

- **Store de Redux** (`frontend/src/store/store.js`):

javascript Copy

```javascript
import { configureStore } from '@reduxjs/toolkit';
import playerReducer from './playerSlice';

export default configureStore({
  reducer: {
    player: playerReducer,
  },
});
```

Paso 3: Componente Reproductor de Música

- **Player.jsx**:

jsx Copy

```jsx
import ReactPlayer from 'react-player';

export default function Player({ url }) {
  return (
    <div className="fixed-bottom bg-dark text-white p-3">
      <ReactPlayer
        url={url}
        controls
        playing
        width="100%"
```

javascript Copy

Paso 3: Componente Reproductor de Música

- **Player.jsx**:

jsx Copy

```jsx
import ReactPlayer from 'react-player';

export default function Player({ url }) {
  return (
    <div className="fixed-bottom bg-dark text-white p-3">
      <ReactPlayer
        url={url}
        controls
        playing
        width="100%"
        height="50px"
        config={{
          file: {
            attributes: { controlsList: 'nodownload' },
          },
        }}
      />
    </div>
  );
}
```

Paso 4: Integración con la API de Spotify

- **Búsqueda de Canciones** (`frontend/src/components/Search.jsx`):

jsx Copy

```jsx
import axios from 'axios';

function Search() {
  const [query, setQuery] = useState('');
  const [tracks, setTracks] = useState([]);

  const searchTracks = async () => {
    const response = await axios.get('http://localhost:5000/api/spotify/se
```

Paso 4: Integración con la API de Spotify

- **Búsqueda de Canciones** (`frontend/src/components/Search.jsx`):

jsx Copy

```jsx
import axios from 'axios';

function Search() {
  const [query, setQuery] = useState('');
  const [tracks, setTracks] = useState([]);

  const searchTracks = async () => {
    const response = await axios.get('http://localhost:5000/api/spotify/search', {
      params: { q: query },
      headers: { Authorization: `Bearer ${localStorage.getItem('token')}` },
    });
    setTracks(response.data);
  };

  return (
    <div>
      <input type="text" onChange={(e) => setQuery(e.target.value)} />
      <button onClick={searchTracks}>Buscar</button>
      {tracks.map((track) => (
        <div key={track.id}>{track.name}</div>
      ))}
    </div>
  );
}
```

5. Dockerización

Paso 1: Dockerfile del Backend

Dockerfile Copy

```
FROM node:18
```

Message DeepSeek

DeepThink (R1) Search

5. Dockerización

Paso 1: Dockerfile del Backend

Dockerfile Copy

```
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5000
CMD ["node", "src/index.js"]
```

Paso 2: Dockerfile del Frontend

Dockerfile Copy

```
FROM node:18 as build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
EXPOSE 80
```

Paso 3: docker-compose.yml

yaml Copy

```yaml
version: '3.8'
services:
  backend:
    build: ./backend
    ports:
      - "5000:5000"
    environment:
      - DATABASE_URL=mongodb://mongo:27017/musicdb
```

6. Despliegue en AWS ECS

Paso 1: Subir Imágenes a Amazon ECR

1. **Autenticar Docker en ECR**:

bash Copy

```
aws ecr get-login-password --region tu-region | docker login --username AW
S --password-stdin tu-id-cuenta.dkr.ecr.tu-region.amazonaws.com
```

2. **Crear Repositorios**:

bash Copy

```
aws ecr create-repository --repository-name music-player-backend
aws ecr create-repository --repository-name music-player-frontend
```

3. **Subir Imágenes**:

bash Copy

```
docker tag backend:latest tu-ecr-url/music-player-backend:latest
docker push tu-ecr-url/music-player-backend:latest

docker tag frontend:latest tu-ecr-url/music-player-frontend:latest
docker push tu-ecr-url/music-player-frontend:latest
```

Paso 2: Configurar Amazon ECS

1. **Crear un Clúster ECS** (tipo Fargate).
2. **Definir Task Definitions** para frontend y backend.
3. **Crear Servicios**:
   - Asignar 1 GB de RAM y 0.5 vCPU.
   - Configurar balanceador de carga (ALB) si es necesario.

7. Pruebas y Documentación

Pruebas Locales

7. Pruebas y Documentación

Pruebas Locales

1. **Ejecutar con Docker Compose**:

```bash
docker-compose up --build
```

2. **Verificar**:
   - Frontend: `http://localhost:3000`.
   - Backend: `http://localhost:5000`.

Documentación de la API

Usa **Swagger** o **Postman** para documentar los endpoints. Ejemplo:

```yaml
openapi: 3.0.0
info:
  title: Music Player API
  version: 1.0.0
paths:
  /api/spotify/search:
    get:
      summary: Buscar canciones en Spotify
      parameters:
        - name: q
          in: query
          required: true
          schema:
            type: string
```

8. Posibles Mejoras

1. **Sistema de Playlists**.