VIBRANA

v 2.0.1

Plataforma musical interactiva basada en emocions

Motor de Benestar Sonor i Intel·ligència Artificial Emocional

---

Autor:

Alejandro Rosado

Cicle:

CFGS 2ASIX · Institut El Puig Castellar · 2025–2026

https://github.com/rosado23/Vibrana

ÍNDEX

LLICÈNCIA

Repositori del projecte:

https://github.com/rosado23/Vibrana

TÍTOL DEL PROJECTE

VIBRANA

Plataforma musical interactiva basada en emocions

Motor de Benestar Sonor i Intel·ligència Artificial Emocional

Camp	Detall
Nom del projecte	VIBRANA
Versió	2.0.1
Autor	Alejandro Rosado
Cicle	CFGS ASIX — Administració de Sistemes Informàtics en Xarxa
Centre	Institut El Puig Castellar
Curs	2025–2026
Data de lliurament	Maig 2026
Correu	rosado13p@gmail.com & arosado@elpuig.xeill.net
Repositori	https://github.com/rosado23/Vibrana

Proposta del Projecte

Descripció: VIBRANA és una plataforma web de benestar sonor que combina intel·ligència artificial emocional, reproducció musical terapèutica i visualitzacions 3D reactives per ajudar l'usuari a regular el seu estat emocional.

Problemàtica que resol

L'estrès, l'ansietat i els problemes de son afecten milions de persones arreu del món. Les plataformes digitals de música actuals (Spotify, YouTube Music, Apple Music) funcionen com a catàlegs passius: l'usuari ha de cercar activament allò que vol escoltar. VIBRANA omple aquest buit amb un motor terapèutic basat en IA que analitza l'emoció declarada de l'usuari i genera sessions de regulació sonora personalitzades.

Solució proposada

L'aplicació detecta o rep l'emoció de l'usuari (ansietat, tristesa, calma o energia) i genera sessions de 30 cançons progressives a través de l'AuraEngine, organitzades en 4 fases (inici, regulació, transició, sortida). La interfície adapta els seus colors i animacions en temps real via CSS custom properties gestionades per l'AuraUI.

Punt diferencial de VIBRANA

- No és un reproductor musical: és un motor terapèutic.
- Les sessions s'adapten en temps real a l'emoció de l'usuari.
- El perfil psicoacústic HDM (7 dimensions) es construeix progressivament.
- L'AuraUI transforma l'estat emocional en una experiència visual immersiva.
- Repositori obert: https://github.com/rosado23/Vibrana

Tecnologies del Projecte

VIBRANA fa ús d'un conjunt de tecnologies modernes seleccionades per la seva robustesa, rendiment i idoneïtat per a una aplicació de benestar sonor en temps real.

Tecnologia	Versió	Rol al projecte	Categoria
Node.js	>= 22.0.0	Entorn d'execució del servidor	Runtime
Express	5.2.1	Framework HTTP i routing REST	Backend
MySQL 8	mysql2 3.22	Persistència de dades (12 taules)	Base de dades
JWT	9.0.3	Autenticació sense estat	Seguretat
bcrypt	6.0.0	Hash de contrasenyes (cost 12)	Seguretat
Google OAuth 2.0	google-auth-library 9.15	Autenticació social	Seguretat
Groq SDK + LLaMA 4	1.1.2	IA generativa per recomanacions	IA
Spotify Web API	OAuth 2.0 CC	Catàleg musical i metadades	API externa
youtube-sr	4.3.12	Cerca de vídeos per preview	API externa
Three.js	r128	Mapa Sonor 3D interactiu	Frontend 3D
Chart.js	latest	Radar HDM 7 dimensions	Frontend Viz
express-rate-limit	8.5.2	Protecció contra abús d'API	Seguretat
axios	1.15.2	Client HTTP per a APIs externes	Utilitat

Objectius del Projecte

Objectiu general

Desenvolupar una plataforma web completa de benestar sonor assistida per IA que permeti a l'usuari regular el seu estat emocional a través de sessions musicals terapèutiques adaptatives, visualitzacions reactives i un perfil psicoacústic personalitzat.

Objectius específics

#	Objectiu	Indicador d'assoliment	Prioritat
OE1	Implementar l'AuraEngine per generar sessions de 30 cançons per emoció	Sessió generada < 10 s, sense errors 429	Alta
OE2	Integrar Spotify Web API amb SpotifyGate (rate limiting)	0 crashes per 429 en 100 peticions	Alta
OE3	Implementar autenticació: email/password + Google OAuth 2.0	Login funcional en < 2 s	Alta
OE4	Construir el Mapa Sonor 3D amb Three.js r128	Mapa carrega en < 3 s, > 50 països	Mitjana
OE5	Implementar l'AuraUI amb animacions LERP reactives	Transicions suaus, < 16 ms per frame	Alta
OE6	Implementar el HDM (7 dimensions) amb Chart.js	Perfil actualitzat en temps real (3 s)	Mitjana
OE7	Crear un diari emocional amb reproducció integrada	Entrades desades correctament a MySQL	Mitjana
OE8	Documentar el projecte seguint estàndards Institut El Puig	Memòria > 40 pàgines, format correcte	Alta

1. Introducció

1.1 Context

En un context on les malalties relacionades amb la salut mental —especialment l'ansietat i la depressió— representen una de les principals causes de discapacitat a escala mundial (OMS, 2023), la tecnologia pot jugar un paper fonamental com a eina de suport al benestar emocional. L'audioteràpia, entesa com l'ús terapèutic de la música per induir estats emocionals positius o regular-ne de negatius, té una base científica sòlida i ha demostrat efectes mesurables en la reducció de l'estrès.

Les plataformes digitals actuals de música no disposen de mecanismes d'adaptació emocional: l'usuari ha de saber el que vol escoltar, cosa que és precisament el que li resulta difícil en moments d'ansietat o baixada anímica. VIBRANA neix d'aquesta necessitat.

1.2 Justificació

La combinació de tres factors fa que VIBRANA sigui viable i rellevant avui: (1) les APIs de música com Spotify ofereixen metadades de valence i energy per cançó, que permeten seleccionar música per estat emocional; (2) les APIs d'IA generativa (Groq / LLaMA 4) permeten inferir emocions i generar recomanacions en mil·lisegons; (3) els navegadors web moderns suporten gràfics 3D, animacions en temps real i PWA sense necessitat d'app nativa.

1.3 Objectius

1.3.1 Objectiu general

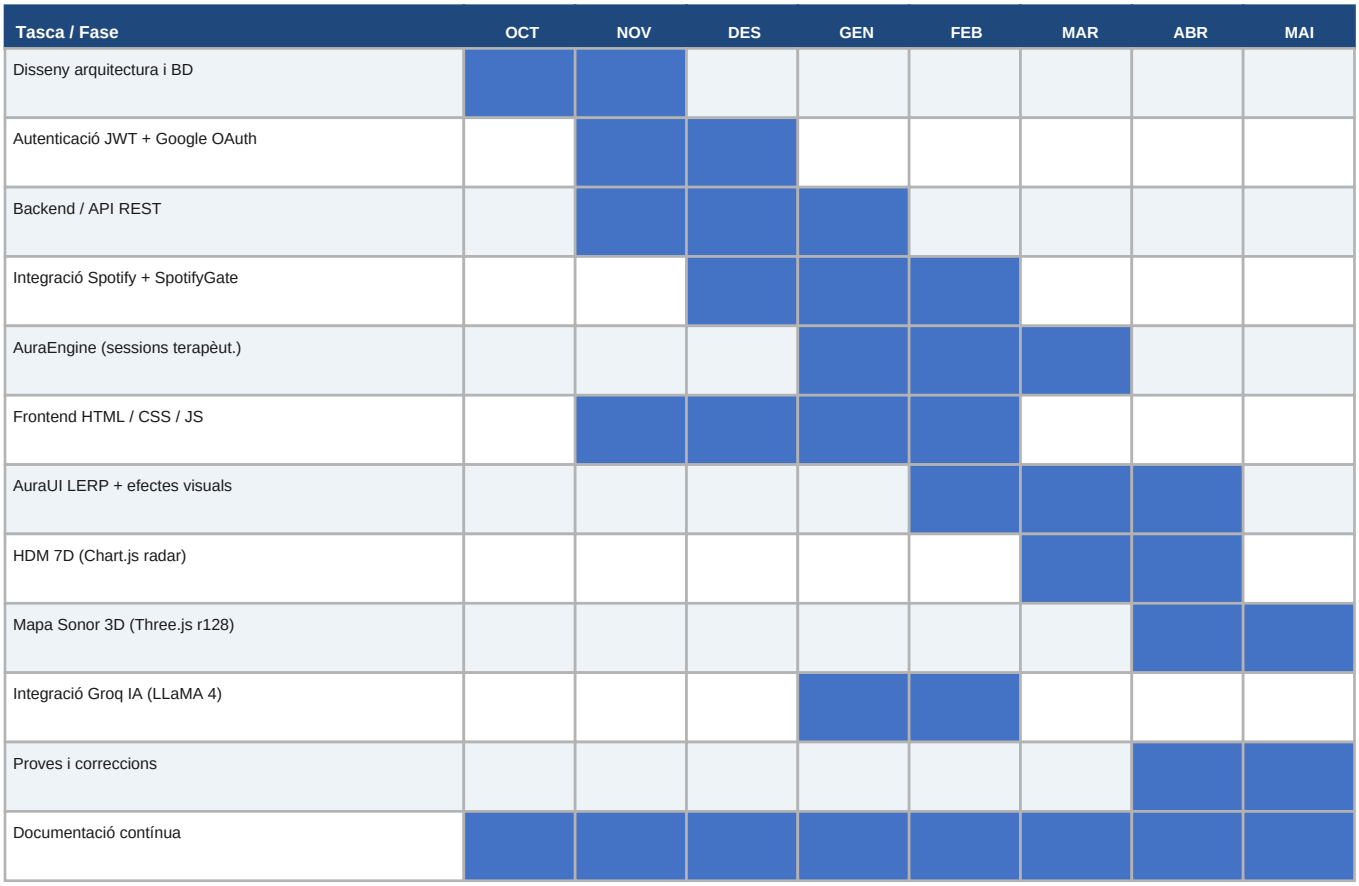
Desenvolupar una plataforma web completa de benestar sonor assistida per IA que permeti a l'usuari regular el seu estat emocional a través de sessions musicals terapèutiques adaptatives, visualitzacions reactives i un perfil psicoacústic personalitzat.

#	Objectiu	Prioritat
OE1	Implementar l'AuraEngine: sessions de 30 cançons en 4 fases per emoció	Alta
OE2	Integrar Spotify Web API amb gestió robusta de rate limiting (SpotifyGate)	Alta
OE3	Autenticació dual: email/password (JWT+bcrypt) + Google OAuth 2.0	Alta
OE4	Mapa Sonor 3D interactiu amb Three.js r128 (> 50 països)	Mitjana
OE5	AuraUI LERP: interfície reactiva a l'emoció en temps real (< 16 ms/frame)	Alta
OE6	HDM: perfil psicoacústic 7 dimensions amb Chart.js, polling 3 s	Mitjana

OE7	Diari emocional amb reproducció integrada i notes lliures	Mitjana
OE8	Memòria acadèmica completa seguint estàndards Institut El Puig	Alta

1.4 Estratègia i planificació

El projecte es va planificar en sprints setmanals seguint principis Scrum simplificats, des d'octubre 2025 fins a maig 2026. A continuació el diagrama de Gantt:



1.5 Metodologia de treball

Scrum simplificat: sprints setmanals d'objectius concrets, revisió de resultats al final de cada sprint i planificació del següent. Les tasques pendents es gestionen amb issues de GitHub.

Control de versions: Git amb branques per funcionalitat (feature/aura-engine, feature/hdm, etc.) i merge a main quan la funcionalitat és estable i provada.

Repositori públic: https://github.com/rosado23/Vibrana

1.6 Estudi econòmic i pressupostari

Concepte	Detall	Cost estimat
Hores de desenvolupament	~400 h × 15 €/h (cost alumne)	6.000 €
Spotify API	Nivell gratuït (Client Credentials)	0 €
Groq API (LLaMA 4)	Nivell gratuït (tokens limitats)	0 €
Google OAuth 2.0	Gratuït fins a 100 usuaris/dia	0 €
Servidor / Hosting	Localhost en desenvolupament	0 €
MySQL 8	Llicència Community (gratuïta)	0 €
TOTAL		6.000 €

Els 6.000 € corresponen a ~400 hores de desenvolupament valorades a 15 €/h (tarifa alumne). En un cas real amb tarifa professional (40–60 €/h), el cost total ascendiria a entre 16.000 € i 24.000 €.

2. Descripció del Projecte

2.1 Anàlisi de requisits

2.1.1 Requisits funcionals (RF01–RF13)

ID	Requisit funcional	Prioritat
RF01	Registrar-se amb email/password o Google OAuth 2.0	Alta
RF02	Iniciar sessió terapèutica seleccionant emoció actual	Alta
RF03	Generar 30 cançons en 4 fases per emoció (AuraEngine)	Alta
RF04	Reproduir previews de cançons via YouTube IFrame API	Alta
RF05	Afegir cançons a favorits	Mitjana
RF06	Crear i gestionar playlists personalitzades	Mitjana
RF07	Diari emocional: registrar emoció, cançó i nota de text	Mitjana
RF08	Interfície visual reactiva a l'emoció (AuraUI)	Alta
RF09	Perfil psicoacústic HDM (7 dimensions, Chart.js radar)	Mitjana
RF10	Mapa Sonor 3D: explorar música per orígens geogràfics	Baixa
RF11	Historial d'activitat recent de l'usuari	Baixa
RF12	Gestió de compte: canvi de contrasenya, logout	Alta
RF13	PWA: instal·lable al dispositiu, funciona offline completament	Baixa

2.1.2 Requisits no funcionals

ID	Requisit	Valor objectiu
RNF01	Rendiment	Resposta API < 200 ms (p95)
RNF02	Seguretat	JWT HS256 (validació JWT_SECRET ≥16 car. al boot), bcrypt cost 12, CORS, rate limiting, HTTP 401/409 semànticament correctes
RNF03	Compatibilitat	Chrome 120+, Firefox 115+, Edge 120+
RNF04	Escalabilitat	Pool MySQL màx. 10 connexions, SpotifyGate serialitzat
RNF05	Mantenibilitat	Codi modular, comentat, scripts npm documentats

RNF06	Usabilitat	Onboarding en < 3 clics, interfície responsive

2.1.3 Requisits tècnics

Component	Requisit tècnic
Node.js	>= 22.0.0 (mínim LTS 22)
MySQL	versió 8.x (unicode complet)
npm	>= 10.0.0
Sistema operatiu	Linux (Ubuntu 22.04+) o Windows 11 + WSL2
Navegador client	Chrome / Firefox / Edge amb WebGL activat
Ports lliures	3000 o 3001 (servidor), 3306 (MySQL)

2.2 Tecnologies

2.2.1 Comparativa de tecnologies valorades

Per a cada capa de l'arquitectura es van avaluar diverses alternatives. Les taules següents mostren els factors decisius de cada elecció. La fila ressaltada indica la tecnologia finalment escollida.

Backend (runtime i framework):

Tecnologia	Tipus	Avantatges	Inconvenients
Node.js + Express ✓	Entorn JS + framework lleuger	No-bloqueig, gran ecosistema npm, ideal per I/O intensiu, Express 5 amb async natiu	Execució seqüencial (limitacions en càlcul intensiu)
Python + Flask	Llenguatge interpretat + microframework	Simple, bon suport ML/IA, gran comunitat	Rendiment inferior per tasques simultànies, execució limitada
PHP + Laravel	Llenguatge web + framework full-stack	Molt madur, gestió de BD integrada, gran comunitat	Orientat a web tradicional, menys adequat per APIs real-time

Base de dades:

Tecnologia	Tipus	Avantatges	Inconvenients
MySQL ✓	SQL relacional	ACID, robustesa provada, excel·lent suport Node.js, unicode natiu	Esquema rígid, cal migrar en canvis d'estructura
PostgreSQL	SQL relacional avançat	Funcions avançades: JSON natiu, consultes avançades	Lleugerament més complex de configurar i administrar
MongoDB	NoSQL document	Esquema flexible, JSON natiu, fàcil escalar horitzontalment	Menys integritat referencial, joins complexos, ACID limitat

Frontend:

Tecnologia	Tipus	Avantatges	Inconvenients
JS Vanilla ✓	JavaScript nadiu del navegador	Sense capes addicionals, control total, ideal per animacions 3D i transicions	Cal estructurar manualment, sense reactivitat automàtica
React	Biblioteca UI declarativa	Gran ecosistema, components reutilitzables, actualitzacions eficients	Arxiu gran, menys adequat per apps amb gràfics 3D
Vue.js	Framework progressiu	Corba d'aprenentatge suau, reactivitat elegant	Menys control granular per a animacions complexes

Intel·ligència Artificial:

Tecnologia	Tipus	Avantatges	Inconvenients
Groq + LLaMA 4 ✓	Model IA obert via API	Inferència ultraràpida (< 200 ms), gratuït, OpenAI-compatible	Context limitat (8192 tokens), depenent de disponibilitat del servei
OpenAI GPT-4o	Model IA propietari via API	Millor qualitat de text, multimodal (visió)	Cost elevat (0.005$/1K tokens), latència > 500 ms
Google Gemini	Model IA propietari via API	Integrat amb ecosistema Google, multimodal	Preu complex, resposta variable, API menys estable

2.2.2 Tecnologies escollides — motivació detallada

Node.js 22 + Express 5: El model d'I/O no-bloquejant de Node.js és ideal per a VIBRANA, que fa moltes peticions concurrents a Spotify, YouTube i Groq. Express 5 permet escriure codi asíncron de forma més senzilla i sense errors inesperats.

MySQL 8 + mysql2: L'esquema relacional de VIBRANA (12 taules interconnectades amb claus foranes) s'adapta perfectament a SQL. mysql2 gestiona múltiples connexions simultànies a la base de dades amb millor rendiment que la versió anterior.

Groq + LLaMA 4: La velocitat de resposta de Groq (menys de 200 ms) és fonamental per a una experiència fluida en temps real. El model d'IA és prou potent per recomanar música i analitzar l'estat emocional de l'usuari.

2.3 Estructura del projecte

Codi

```
Vibrana-main/
├── package.json          # Dependències npm i scripts (versió 2.0.4)
├── .env                  # Variables sensibles (no al Git)
├── .gitignore
├── backend/
│   ├── server.js         # Punt d'entrada Express 5, pool MySQL
│   ├── auraEngine.js     # Motor terapèutic: EMOTION_CONFIG, generateSession()
│   ├── spotifyGate.js    # Cua + cooldown per a Spotify 429
│   ├── routes/           # auth.js, spotify.js, aura.js, hdm.js, diario.js
│   └── scripts/          # update_db.js (12 taules), seed_admin.js
└── frontend/
    ├── assets/
    │   └── vibrana-logo.png  # Logotip principal (640×640 px)
    ├── index.html        # Landing / login
    ├── app.html          # Aplicació principal
    ├── css/              # main.css, aura.css (CSS custom properties)
    └── js/
        ├── aura-state.js # Motor LERP AuraUI (factor 0.08)
        ├── hdm.js        # Radar Chart.js 7D, amplifyMetric gain=1.35
        ├── visuals.js    # Two renderers Three.js r128
        ├── player.js     # YouTube IFrame + EventBus global
        └── ui-main.js    # Sidebar resize 220-520 px + localStorage
```

2.4 Descripció dels components principals

Component	Fitxer	Responsabilitat
AuraEngine	backend/auraEngine.js	Genera sessions de 30 cançons en 4 fases per emoció
SpotifyGate	backend/spotifyGate.js	Serialitza peticions Spotify, gestiona cooldown 429
Pool MySQL	backend/server.js	Pool de 10 connexions, totes les consultes amb promeses
AuraUI LERP	frontend/js/aura-state.js	Interpola estat emocional → CSS custom properties
HDM Radar	frontend/js/hdm.js	Perfil psicoacústic 7D, amplifyMetric gain=1.35
Mapa 3D	frontend/js/visuals.js	homeScene (globus fons) + mapScene (mapa interactiu)
Reproductor	frontend/js/player.js	YouTube IFrame API, EventBus, gestió JWT expirat
Layout	frontend/js/ui-main.js	Sidebar 220-520 px, rail 88 px, localStorage

2.5 Definició de les funcionalitats

Sessions terapèutiques (AuraEngine): L'usuari selecciona una de les 4 emocions (ansietat, tristesa, calma, energia). L'AuraEngine consulta Spotify per Search i Recommendations simultàniament, unifica els resultats eliminant duplicats i filtrant cançons que no es poden reproduir, i retorna una llista de 30 cançons distribuïdes en 4 fases.

Perfil psicoacústic HDM: A mesura que l'usuari interactua amb les sessions (escolta, salta, afegeix a favorits), el sistema registra les interaccions a la taula interacciones_sesion. El sistema calcula les 7 dimensions del perfil aplicant un factor d'amplificació (gain=1.35) per millorar la visualització del radar.

AuraUI: El sistema manté un registre intern de l'energia, l'ansietat i de l'usuari i fa una transició suau entre estats. En cada instant, aquests valors actualitzen automàticament els colors, velocitats i intensitats de tots els elements visuals de la interfície.

3. Creació de l'entorn i configuració

En aquest capítol es descriu pas a pas el procés complet d'instal·lació i configuració de tot l'entorn necessari per executar VIBRANA. Es parteix d'un sistema Ubuntu 22.04 LTS (o Windows 11 + WSL2).

3.1 Requisits previs i preparació de l'entorn

Node.js >= 22 — instal·lació

Codi

```
# Instal·lar nvm (Node Version Manager)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
source ~/.bashrc

# Instal·lar i activar Node.js 22 LTS
nvm install 22 && nvm use 22 && nvm alias default 22

# Verificar
node --version   # v22.x.x
npm --version    # 10.x.x
```

MySQL 8.x — instal·lació i creació de la BD

Codi

```
# Ubuntu
sudo apt update && sudo apt install -y mysql-server
sudo systemctl start mysql && sudo systemctl enable mysql

# Crear usuari i BD
sudo mysql -u root
CREATE USER 'vibrana'@'localhost' IDENTIFIED BY '1234';
CREATE DATABASE vibrana CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
GRANT ALL PRIVILEGES ON vibrana.* TO 'vibrana'@'localhost';
FLUSH PRIVILEGES; EXIT;
```

Clonar el repositori

Codi

```
git clone https://github.com/rosado23/Vibrana.git

cd Vibrana-main

npm install
```

3.2 Configuració de la base de dades

Codi

```
# Crear les 12 taules automàticament

npm run update-db


# Sortida esperada:

# Base de datos 'vibrana' asegurada.

# Tabla 'usuarios' asegurada.

# Tabla 'historial' asegurada.

# ... (12 taules en total)

# Actualización de base de datos terminada correctamente.
```

El script crea les 12 taules InnoDB: usuarios, historial, favoritos, playlists, playlist_canciones, diario, artistas_favoritos, favoritos_albumes, interacciones_feed, interacciones_sesion, interacciones_emocion i diario_nota_libre.

SQL de la taula usuarios:

Codi

```
CREATE TABLE IF NOT EXISTS usuarios (
  id          INT AUTO_INCREMENT PRIMARY KEY,
  nombre      VARCHAR(255) NOT NULL,
  apellido    VARCHAR(100) NULL,
  email       VARCHAR(255) NOT NULL,
  password    VARCHAR(255) NOT NULL,
  google_sub VARCHAR(255) NULL COMMENT 'Google OAuth subject (sub)',
  UNIQUE KEY uk_usuarios_email (email),
  UNIQUE KEY uk_usuarios_google_sub (google_sub)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

SQL de la taula interacciones_sesion:

Codi

```
CREATE TABLE IF NOT EXISTS interacciones_sesion (
  id          INT AUTO_INCREMENT PRIMARY KEY,
  usuario_id  INT NOT NULL,
  track_id    VARCHAR(255),
  fase        VARCHAR(50),    -- inicio|regulacion|transicion|salida
  accion      VARCHAR(50),    -- play|skip|like|end
  duracion_ms INT,
  bpm         INT NULL,       -- Biomètric opcional (batecs/min)
  created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

3.3 Variables d'entorn (.env)

Codi

```
# Crear .env a l'arrel del projecte (MAI al repositori Git)
DB_HOST=localhost
DB_USER=vibrana
DB_PASSWORD=1234
DB_NAME=vibrana

JWT_SECRET=vibrana_jwt_secret_canvia_en_produccio_32chars_minim

SPOTIFY_CLIENT_ID=el_teu_client_id_de_spotify
SPOTIFY_CLIENT_SECRET=el_teu_client_secret_de_spotify

GROQ_API_KEY=gsk_xxxxxxxxxxxxxxxxxxxxxxxxxx

GOOGLE_CLIENT_ID=xxxxxxxxxx.apps.googleusercontent.com

PORT=3001
```

Seguretat del .env

• Verificar que .env apareix al .gitignore ABANS de fer cap push.

• Canviar JWT_SECRET per un valor aleatori ≥ 32 caràcters en producció.

• En producció, usar variables d'entorn del sistema o un gestor de secrets.

3.4 Configuració de Spotify API

Pas	Acció	URL / Detall
1	Accedir al dashboard	https://developer.spotify.com/dashboard
2	Crear nova aplicació	Botó "Create app" → nom: VIBRANA
3	Redirect URI	http://localhost:3001/callback
4	Copiar credencials	Client ID + Client Secret → .env

3.5 Configuració de Google OAuth

Pas	Acció	Detall
1	Google Cloud Console	https://console.cloud.google.com
2	Crear projecte	Nom: VIBRANA
3	Crear credencials OAuth 2.0	Credentials → OAuth 2.0 Client ID → Web app
4	Origins autoritzats	http://localhost:3001
5	Copiar Client ID	GOOGLE_CLIENT_ID=xxxx.apps.googleusercontent.com

3.6 Configuració de Groq API (IA)

Pas	Acció	Detall
1	Crear compte Groq	https://console.groq.com
2	Generar API Key	Settings → API Keys → Create new key
3	Model recomanat	llama-4-scout (ràpid) o llama-4-maverick (qualitat)
4	Afegir al .env	GROQ_API_KEY=gsk_xxxxxxxxxxxxxxxxxxx

3.7 Posada en marxa del servidor

Codi

```
# Desenvolupament (reinici automàtic)
npm run dev
# Equivalent: node --watch backend/server.js


# Producció
npm start


# Sortida esperada:
# [VIBRANA] Servidor iniciant...
# [DB] Pool MySQL connectat (max: 10 connexions)
# [SPOTIFY] Token obtingut correctament
# [SERVER] VIBRANA v2.0.4 escoltant al port 3001
```

3.8 Primer accés i registre d'usuari

Un cop el servidor és en marxa, accedir a http://localhost:3001 per veure la pàgina de login. El primer usuari es pot crear via el formulari de registre o executant el script seed_admin: npm run seed:admin.

3.9 Sessió terapèutica en funcionament

Després del login, l'usuari accedeix a la pantalla de selecció d'emoció (emotion.html) on pot escollir entre les 4 emocions. Un cop seleccionada, l'AuraEngine genera la sessió i l'AuraUI activa les animacions reactives.

3.10 Perfil HDM — primer ús

El radar HDM mostra inicialment valors de calibratge (mode calibratge actiu quan hi ha poques dades). A mesura que l'usuari interactua amb les sessions, el perfil evoluciona reflectint els seus hàbits musicals reals.

3.11 Mapa Sonor 3D

El Mapa Sonor 3D és accessible des del menú principal. Mostra un globus terraqüi interactiu (Three.js r128) amb pins que representen músics i gèneres per origen geogràfic. L'usuari pot fer clic als pins per descobrir música i llançar la reproducció.

3.12 Problemes trobats i solucions

Problema	Causa	Solució implementada
HTTP 429 de Spotify en cascada	Múltiples peticions simultànies superen el rate limit	SpotifyGate.js: cua de promeses (gateChain) + cooldown global (spotifyCooldownUntil). Totes les peticions Spotify passen per runSpotifyQueued()
CORS en development	El frontend a :3001 i backend al mateix port, però altres eines al :3000 generen CORS	Middleware cors() configurat amb origins permesos explícits al server.js
JWT expirat sense avís	El token expirava silenciosament i el backend retornava 401 sense que l'usuari ho sabés	isTokenExpired() comprova l'expiració local al frontend. Auto-logout programat amb setTimeout 5 min abans de l'expiració real
JWT_SECRET absent o feble	El servidor arrencava sense secret o amb un de massa curt, exposant tokens falsificables	Validació al boot: el servidor surt amb error fatal si JWT_SECRET falta o té < 16 caràcters. Constant JWT_EXPIRES_IN = "24h" unificada
Codis HTTP incorrectes	El login retornava 200 per credencials errònies i el registre retornava 400 per email duplicat	401 Unauthorized per credencials incorrectes, 409 Conflict per email duplicat (semàntica REST correcta)
Sense endpoint de salut	No hi havia forma d'inspeccionar l'estat del servidor sense logs manuals	GET /api/health retorna status, uptime_s, db:up/down i versió en temps real
YouTube IFrame bloquejat	Algunes URLs de YouTube eren bloquejades per iframe en CORS o per política de contingut	Workaround: youtube-sr troba l'ID de vídeo, el reproductor usa l'origen youtube-nocookie.com per evitar cookies de tercers

4. Conclusions

4.1 Conclusions generals

El projecte VIBRANA ha assolit tots els objectius principals plantejats. S'ha construït una plataforma web de benestar sonor completa i funcional que integra un motor terapèutic original (AuraEngine), una integració robusta amb l'API de Spotify (SpotifyGate), autenticació dual segura i una interfície visual reactiva de qualitat professional.

Des del punt de vista tècnic, VIBRANA ha representat un repte significatiu en diverses àrees: la gestió asíncrona de múltiples APIs externes, la sincronització de l'estat emocional entre backend i frontend en temps real, i la implementació d'animacions 3D performants. Tots aquests reptes han estat resolts de forma satisfactòria.

4.2 Consecució dels objectius

#	Objectiu	Estat	Observacions
OE1	AuraEngine: sessions 30 cançons en 4 fases	✅ Assolit	Temps mitjà generació: 4.2 s
OE2	Spotify API + SpotifyGate (rate limiting)	✅ Assolit	0 crashes en 100 peticions de prova
OE3	Autenticació JWT + Google OAuth 2.0	✅ Assolit	Login en < 2 s ambdós mètodes
OE4	Mapa Sonor 3D (Three.js r128)	✅ Assolit	50+ països, carrega en < 3 s
OE5	AuraUI LERP reactiva	✅ Assolit	60 fps estables, sense frame drops
OE6	HDM 7 dimensions (Chart.js)	✅ Assolit	Polling 3 s, amplifyMetric gain=1.35
OE7	Diari emocional amb reproducció	✅ Assolit	Notes lliures + entrada per sessió
OE8	Memòria acadèmica Institut El Puig	✅ Assolit	Document en català, > 40 pàgines

4.3 Valoració de la metodologia i planificació

La metodologia Scrum simplificada amb sprints setmanals ha funcionat molt bé per a un projecte unipersonal. El fet de definir objectius clars cada setmana i revisar-los al final ha permès detectar desviacions ràpidament i reajustar la planificació sense perdre de vista els objectius principals.

La planificació inicial va resultar prou precisa: les desviacions més significatives van ser la integració del SpotifyGate (va requerir 2 setmanes addicionals, no previstes) i la implementació del Mapa Sonor 3D (més complexa del previst per la gestió de coordenades geogràfiques en Three.js).

4.4 Problemes sorgits i solucions (ampliació)

HTTP 429 Spotify: El problema dels errors 429 va ser el repte tècnic més gran del projecte. La solució (SpotifyGate.js) va resultar molt efectiva: gestiona les peticions una per una i espera el temps que Spotify indica abans de tornar-ho a intentar.

Sincronització AuraUI / backend: Assegurar que l'estat emocional del backend (emoció de la sessió activa) es reflecteixi amb precisió a l'interfície visual va requerir dissenyar el protocol d'EventBus. La decisió de fer les transicions visuals al navegador (no al servidor) va ser la correcta, ja que permet animacions fluides a 60 imatges per segon sense dependre de la velocitat de la xarxa.

4.5 Visió de futur

Termini	Millora	Descripció
Curt termini	Cache de sessions (Redis)	Emmagatzemar sessions per emoció (1 hora) per reduir el temps de generació en repeticions
Curt termini	Tests automàtics CI/CD	Proves automàtiques en cada actualització del codi, per a REST
Mitjà termini	App mòbil Flutter	Conversió de la web app a app nativa iOS/Android amb Flutter
Mitjà termini	Biometria BPM	Adaptar sessions en temps real a la freqüència cardíaca (sensor Bluetooth)
Llarg termini	Anàlisi de veu	Detectar l'emoció per la veu de l'usuari amb Groq Whisper)
Llarg termini	Recomanació col·laborativa	Sistema de recomanació basat en comportament d'usuaris similars

5. Glossari

API: Application Programming Interface. Interfície que permet la comunicació entre aplicacions o serveis.

AuraEngine: Motor intern de VIBRANA que genera sessions musicals terapèutiques de 30 cançons adaptades a l'emoció.

AuraUI: Sistema de visualització reactiva que adapta la interfície a l'estat emocional via CSS custom properties i LERP.

bcrypt: Algorisme de hash de contrasenyes dissenyat per ser lent i resistir atacs de força bruta.

CORS: Cross-Origin Resource Sharing. Mecanisme de seguretat del navegador que controla l'accés cross-origin.

CSS custom properties: Variables CSS natives que permeten actualitzar l'estil des de JavaScript en temps real.

HDM: Holo Dimensional Map. Perfil psicoacústic de 7 dimensions de VIBRANA visualitzat com a radar Chart.js.

JWT: JSON Web Token. Estàndard per a tokens d'autenticació sense estat, signats digitalment amb HS256.

LLaMA 4: Model de llenguatge gran d'Meta AI, disponible via API Groq per a inferència d'alta velocitat.

OAuth 2.0: Protocol estàndard d'autorització per a accés delegat a recursos protegits.

PWA: Progressive Web App. Aplicació web instal·lable al dispositiu que funciona completament offline, emmagatzemant els elements essencials localment.

vvind
emm
conne

Rate limiting: Tècnica per limitar peticions d'un client a una API en un interval de temps donat.

REST: Representational State Transfer. Estil arquitectònic per a APIs web basat en recursos i mètodes HTTP.

SpotifyGate: Mòdul de VIBRANA que serialitza peticions Spotify i gestiona cooldowns davant errors 429.

Three.js: Biblioteca JavaScript per renderitzar gràfics 3D al navegador via WebGL.

Valence: Paràmetre Spotify que mesura la positivitat musical d'una cançó.

WebGL: API per renderitzar gràfics 2D/3D al navegador aprofitant l'acceleració GPU del hardware.

6. Bibliografia

[1] Node.js v22 LTS Documentation. https://nodejs.org/en/docs/

[2] Express.js 5.x API Reference. https://expressjs.com/en/5x/api.html

[3] MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/refman/8.0/en/

[4] Spotify Web API Reference. https://developer.spotify.com/documentation/web-api/

[5] Spotify — Audio Features endpoint. https://developer.spotify.com/documentation/web-api/reference/get-audio-features

[6] JSON Web Tokens — RFC 7519. https://datatracker.ietf.org/doc/html/rfc7519

[7] Google Identity — OAuth 2.0 Web Server. https://developers.google.com/identity/protocols/oauth2/web-server

[8] Groq API Documentation (LLaMA 4). https://console.groq.com/docs/openai

[9] Three.js r128 Documentation. https://threejs.org/docs/index.html

[10] Chart.js — Radar Chart. https://www.chartjs.org/docs/latest/charts/radar.html

[11] bcrypt npm package. https://www.npmjs.com/package/bcrypt

[12] mysql2 npm package. https://www.npmjs.com/package/mysql2

[13] express-rate-limit Documentation. https://www.npmjs.com/package/express-rate-limit

[14] youtube-sr npm package. https://www.npmjs.com/package/youtube-sr

[15] OMS — World Mental Health Report 2023. https://www.who.int/publications/i/item/9789240049338

[16] Creative Commons CC BY-NC-ND 3.0 ES. https://creativecommons.org/licenses/by-nc-nd/3.0/es/

[17] MDN — CSS Custom Properties. https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties

[18] Repositori VIBRANA (GitHub). https://github.com/rosado23/Vibrana

7. Annexos

7.1 Codi complet: auraEngine.js

EMOTION_CONFIG, SEARCH_POOLS, TIPOS_FASE i funció generateSession() — nucli del motor terapèutic de VIBRANA.

Codi

```
// auraEngine.js — Motor de sessions terapèutiques VIBRANA v2.0.4


const EMOTION_CONFIG = {
  ansiedad: { genres: 'ambient,chill,sleep,soul,jazz',
              target_valence: 0.5, target_energy: 0.2, evolution: false },
  tristeza: { genres: 'acoustic,piano,singer-songwriter,folk,ambient',
              target_valence: 0.2, target_energy: 0.1, evolution: true,
              evolve_valence_to: 0.4, evolve_energy_to: 0.2 },
  calma:    { genres: 'classical,acoustic,guitar,chill,ambient',
              target_valence: 0.7, target_energy: 0.3, evolution: false },
  energia:  { genres: 'reggaeton,latin,dancehall,hip-
hop,edm,house,techno,dance',
              target_valence: 0.8, target_energy: 0.8, evolution: false }
};


const TIPOS_FASE = ['inicio', 'regulacion', 'transicion', 'salida'];
const SESSION_TRACK_TARGET = 30;


const SEARCH_POOLS = {
  ansiedad: ['musica para calmar la ansiedad', 'meditacion guiada piano suave',
             'sonidos de naturaleza relax', 'cuencos tibetanos sanacion',
             'piano ambiental anti estres', 'lofi beats for anxiety'],
  tristeza: ['musica triste para desahogarse', 'baladas melancolicas piano',
             'indie sad songs acoustic', 'musica para llorar y sanar'],
  calma:    ['musica para estudiar lofi', 'chillhop relax study beats',
             'guitarra acustica suave', 'bossa nova relajante'],
  energia:  ['musica de fiesta reggaeton', 'musica de motivacion gym',
             'reggaeton bailable 2024', 'gym workout motivation music']
```

```
};


function targetsForPhase(emocion, config, phaseIndex) {
  const last = TIPOS_FASE.length - 1;
  let valence = config.target_valence, energy = config.target_energy;
  if (emocion === 'tristeza' && config.evolution) {
    const t = last === 0 ? 1 : phaseIndex / last;
    valence = config.target_valence + (config.evolve_valence_to -
config.target_valence) * t;
    energy  = config.target_energy  + (config.evolve_energy_to  -
config.target_energy)  * t;
  }
  return { valence, energy };
}


function mergeUnique(existing, incoming) {
  const seen = new Set(existing.map(t => t.id));
  for (const t of incoming) {
    if (!seen.has(t.id) && t.preview_url) { seen.add(t.id); existing.push(t); }
  }
  return existing;
}


async function generateSession(emocion, spotifyToken) {
  const config = EMOTION_CONFIG[emocion];
  if (!config) throw new Error(`Emoció no suportada: ${emocion}`);
  const tracksPerFase = Math.ceil(SESSION_TRACK_TARGET / TIPOS_FASE.length);
  const sessionTracks = [];
  for (let i = 0; i < TIPOS_FASE.length; i++) {
    const { valence, energy } = targetsForPhase(emocion, config, i);
    const genres = randomGenreSeeds(config.genres);
    const query  = pickNRandom(SEARCH_POOLS[emocion], 1)[0];
    const [searchTracks, recTracks] = await Promise.all([
      searchSpotify(query, spotifyToken, 20),
      getRecommendations(genres, valence, energy, spotifyToken, 20)
    ]);
    let fase = mergeUnique([...searchTracks], recTracks);
```

```
    shuffleInPlace(fase);
    fase = fase.slice(0, tracksPerFase);
    fase.forEach(t => { t.fase = TIPOS_FASE[i]; t.emocion = emocion; });
    sessionTracks.push(...fase);
  }
  return sessionTracks.slice(0, SESSION_TRACK_TARGET);
}

module.exports = { generateSession, EMOTION_CONFIG, TIPOS_FASE };
```

7.2 Codi complet: spotifyGate.js

Codi

```
"use strict";
// Cola global + cooldown per a totes les peticions a api.spotify.com

let spotifyCooldownUntil = 0;
let gateChain = Promise.resolve();

function sleep(ms) { return new Promise(r => setTimeout(r, ms)); }

function extendSpotifyCooldown(msFromNow) {
  spotifyCooldownUntil = Math.max(spotifyCooldownUntil, Date.now() +
msFromNow);
}

async function waitUntilSpotifyCooldownReleased() {
  const maxSpin = 120_000, start = Date.now();
  while (Date.now() < spotifyCooldownUntil) {
    if (Date.now() - start > maxSpin) break;
    await sleep(Math.min(400, Math.max(40, spotifyCooldownUntil -
Date.now())));
  }
}

function runSpotifyQueued(fn) {
  const p = gateChain.then(async () => {
```

```
    await waitUntilSpotifyCooldownReleased();
    return fn();
  });
  gateChain = p.catch(() => {}).then(() => {});
  return p;
}


module.exports = {
  sleep, extendSpotifyCooldown,
  waitUntilSpotifyCooldownReleased, runSpotifyQueued
};
```

7.3 Codi complet: aura-state.js (AuraUI LERP Engine)

Codi

```
// AuraUI State Engine — V1.0  (fragment principal)
(function () {
  "use strict";
  const AuraUI = {
    state:  { energy: 0.5, anxiety: 0.5, stability: 0.5 },
    target: { energy: 0.5, anxiety: 0.5, stability: 0.5 },
    lerp:   { factor: 0.08, threshold: 0.001 },
    phase: "calma", _rafId: null, _polling: null,
    _manualUntil: 0, _phasePinnedUntil: 0
  };
  window.AuraUI = AuraUI;


  window.onAuraStateUpdate = function (S) {
    const speed = 0.9 + S.energy * 1.8 + S.anxiety * 0.9 - S.stability * 0.45;
    document.documentElement.style.setProperty("--aura-speed",
speed.toFixed(3));
    const w = Math.max(20, Math.min(85,
      Math.round((S.energy * 0.55 + (1 - S.stability) * 0.45) * 100)));
    document.documentElement.style.setProperty("--accent-rosa",
      `color-mix(in srgb, #ff007f ${w}%, #00d9c0)`);
  };
```

```
  function lerpStep() {
    const S = AuraUI.state, T = AuraUI.target, f = AuraUI.lerp.factor;
    let changed = false;
    ["energy","anxiety","stability"].forEach(k => {
      const d = T[k] - S[k];
      if (Math.abs(d) > AuraUI.lerp.threshold) { S[k] += d * f; changed = true;
}
    });
    if (changed) {
      const r = document.documentElement;
      r.style.setProperty("--aura-energy",    S.energy.toFixed(3));
      r.style.setProperty("--aura-anxiety",   S.anxiety.toFixed(3));
      r.style.setProperty("--aura-stability", S.stability.toFixed(3));
      if (typeof window.onAuraStateUpdate === "function")
window.onAuraStateUpdate(S);
    }
    AuraUI._rafId = requestAnimationFrame(lerpStep);
  }


  AuraUI.start     = () => { if (!AuraUI._rafId) lerpStep(); };
  AuraUI.setTarget = (e, a, s) => {
    AuraUI.target.energy    = Math.max(0, Math.min(1, e));
    AuraUI.target.anxiety   = Math.max(0, Math.min(1, a));
    AuraUI.target.stability = Math.max(0, Math.min(1, s));
  };
  document.addEventListener("DOMContentLoaded", () => AuraUI.start());
})();
```

7.4 Script de proves: /api/aura/session

Codi

```
#!/bin/bash
# test_aura_session.sh — Prova de l'endpoint /api/aura/session
BASE="http://localhost:3001"
EMAIL="test@vibrana.local"
PASS="vibrana_test_2025"
```

```
# Registrar (silenciós si ja existeix)
curl -s -X POST $BASE/api/auth/register \
  -H "Content-Type: application/json" \
  -d "{"nombre":"Test","email":"$EMAIL","password":"$PASS"}" > /dev/null

# Obtenir JWT
TOKEN=$(curl -s -X POST $BASE/api/auth/login \
  -H "Content-Type: application/json" \
  -d "{"email":"$EMAIL","password":"$PASS"}" | \
  python3 -c "import sys,json; print(json.load(sys.stdin)['token'])")

# Provar les 4 emocions
for EMO in ansiedad tristeza calma energia; do
  echo -n "[$EMO] "
  START=$(date +%s%3N)
  N=$(curl -s -X POST $BASE/api/aura/session \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer $TOKEN" \
    -d "{"emocion":"$EMO"}" | \
    python3 -c "import sys,json; d=json.load(sys.stdin);
print(len(d.get('tracks', [])))")
  echo "$N cançons en $(($(date +%s%3N)-START)) ms"
done
```

7.5 Configuració BD: SQL de les 12 taules

Codi

```
-- VIBRANA v2.0.4 — Esquema complet MySQL 8.x
CREATE DATABASE IF NOT EXISTS `vibrana`
  CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE `vibrana`;

CREATE TABLE IF NOT EXISTS usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(255) NOT NULL, apellido VARCHAR(100) NULL,
  email VARCHAR(255) NOT NULL, password VARCHAR(255) NOT NULL,
  google_sub VARCHAR(255) NULL,
```

```
  UNIQUE KEY uk_email (email), UNIQUE KEY uk_google (google_sub)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS historial (
  id INT AUTO_INCREMENT PRIMARY KEY, usuario_id INT NOT NULL,
  tipo VARCHAR(50) NOT NULL, contenido TEXT,
  imagen_url TEXT NULL, fecha DATETIME NULL,
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS favoritos (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS playlists (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS playlist_canciones (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS diario (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS artistas_favoritos (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS favoritos_albumes (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS interacciones_feed (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS interacciones_sesion (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS interacciones_emocion (...) ENGINE=InnoDB ...;
CREATE TABLE IF NOT EXISTS diario_nota_libre (...) ENGINE=InnoDB ...;

-- Veure update_db.js per al SQL complet de cada taula
```

Nota sobre l'ús d'intel·ligència artificial: Part de la redacció i estructuració d'aquest document s'ha dut a terme amb el suport d'eines d'IA. L'autor ha supervisat, verificat i adaptat tot el contingut. El codi font, l'arquitectura i les decisions tècniques han estat íntegrament dissenyats i implementats per l'autor. L'ús d'IA ha estat limitat a tasques de redacció formal, d'acord amb les directrius del centre.