**Platina One – Teljes Felhasználói és Fejlesztői Dokumentáció**

**Vígh János-Székely Olivér-Géró Nikolasz Dominik**

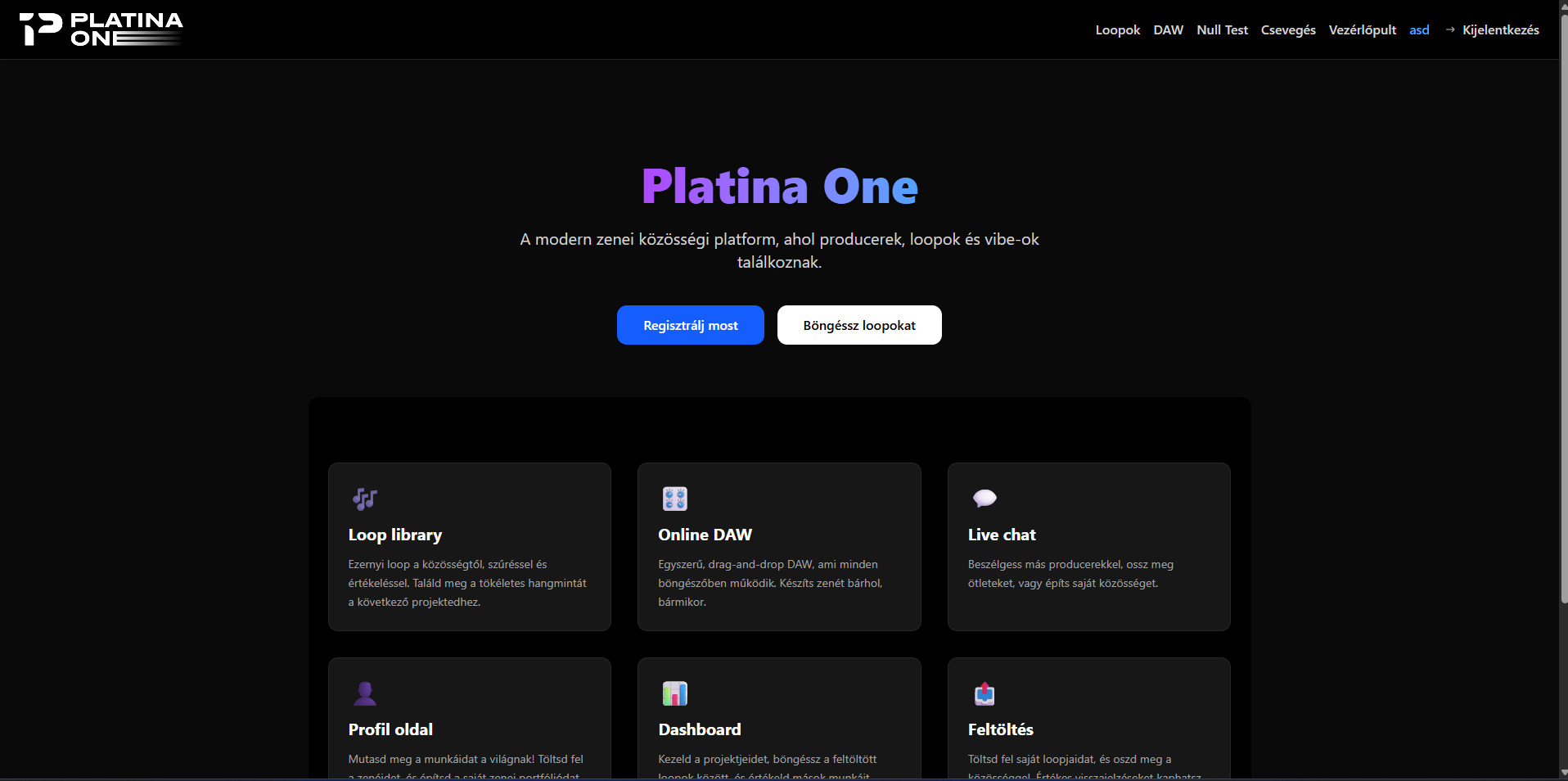
**I. Bevezetés**

**A Platina One egy modern, webalapú zenei platform, amelyet szenvedélyes producerek, hangmérnökök és zene iránt érdeklődő közösségek számára fejlesztettünk ki. A rendszer fő célja, hogy egy intuitív, hatékony és inspiráló környezetet biztosítson a zenei loopok megosztására, értékelésére, kommentelésére, valamint közösségi kommunikációra. A felhasználók nemcsak saját anyagaikat tölthetik fel, hanem más alkotásait is meghallgathatják, értékelhetik, és visszajelzést adhatnak. Mindez segít a kreativitás fejlődésében, a hálózatépítésben, valamint a technikai és művészi készségek csiszolásában is.**

**Ez a dokumentáció részletesen bemutatja a Platina One felhasználói és fejlesztői aspektusait, különös figyelemmel a technikai struktúrára, API-kra, adatbázis-kezelésre, valamint a felhasználói élményre és interakciókra. A cél az, hogy minden érintett – legyen szó fejlesztőről, tesztelőről, adminisztrátorról vagy végfelhasználóról – átfogó képet kapjon a rendszer működéséről, felépítéséről és lehetőségeiről.**

**II. Felhasználói Útmutató**

**A következő szakaszok célja, hogy részletes, lépésről lépésre történő segítséget nyújtsanak minden egyes funkció használatához, kiegészítve tippekkel, példákkal, gyakori hibák kezelésével és a háttérben működő folyamatok ismertetésével.**



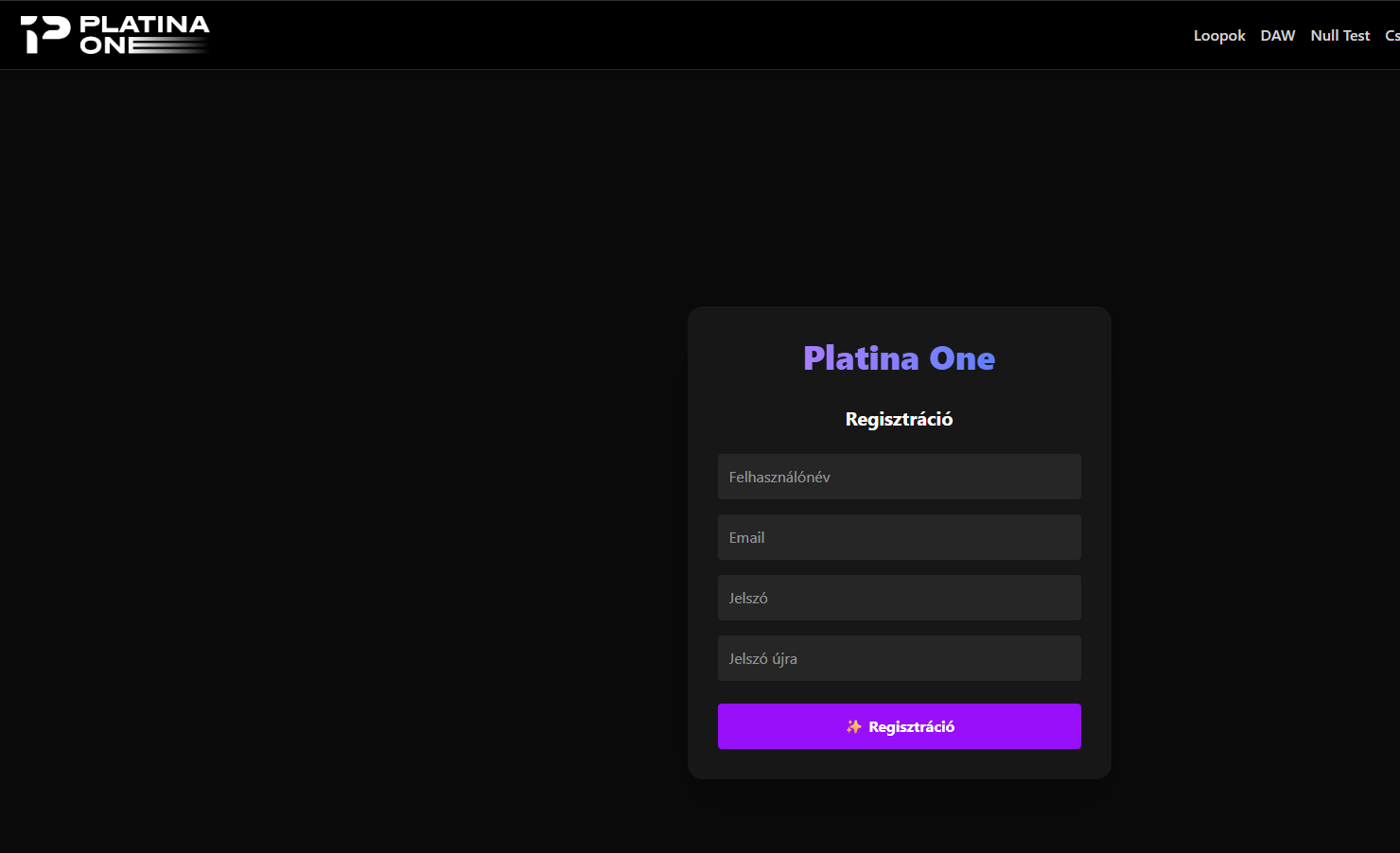
**1. Regisztráció**

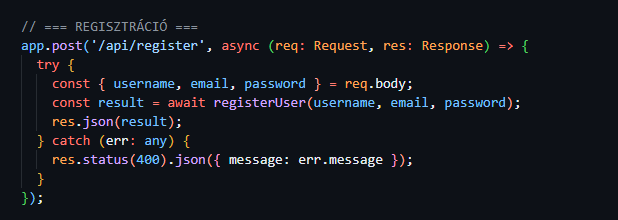
**Miért fontos? A regisztráció lehetővé teszi, hogy egyedi azonosítót kapj a platformon, amellyel minden tevékenységed (loop feltöltés, komment, értékelés, üzenetküldés) hozzád kötődik. Ez biztonságot nyújt, és a közösség bizalmát is erősíti.**

**Regisztrációs mezők:**

* **Felhasználónév: Legyen egyedi, emlékezetes.**
* **E-mail cím: Valós e-mail szükséges a jövőbeli visszaállításokhoz.**
* **Jelszó: Legalább 8 karakter, tartalmazzon betűt és számot.**

**A háttérben: A megadott adatokat a backend hitelesítő rendszer fogadja, ellenőrzi az e-mail cím egyediségét, majd a jelszót bcrypt algoritmussal titkosítja, mielőtt mentésre kerülne az adatbázisba.**





**Hibakezelés:**

* **Már létező e-mail → "Ez az e-mail már foglalt."**
* **Gyenge jelszó → "A jelszónak minimum 8 karakteresnek kell lennie."**

**2. Bejelentkezés**

**Célja: A biztonságos hozzáférés a fiókhoz és a jogosultsággal védett funkciókhoz.**

**Folyamat:**

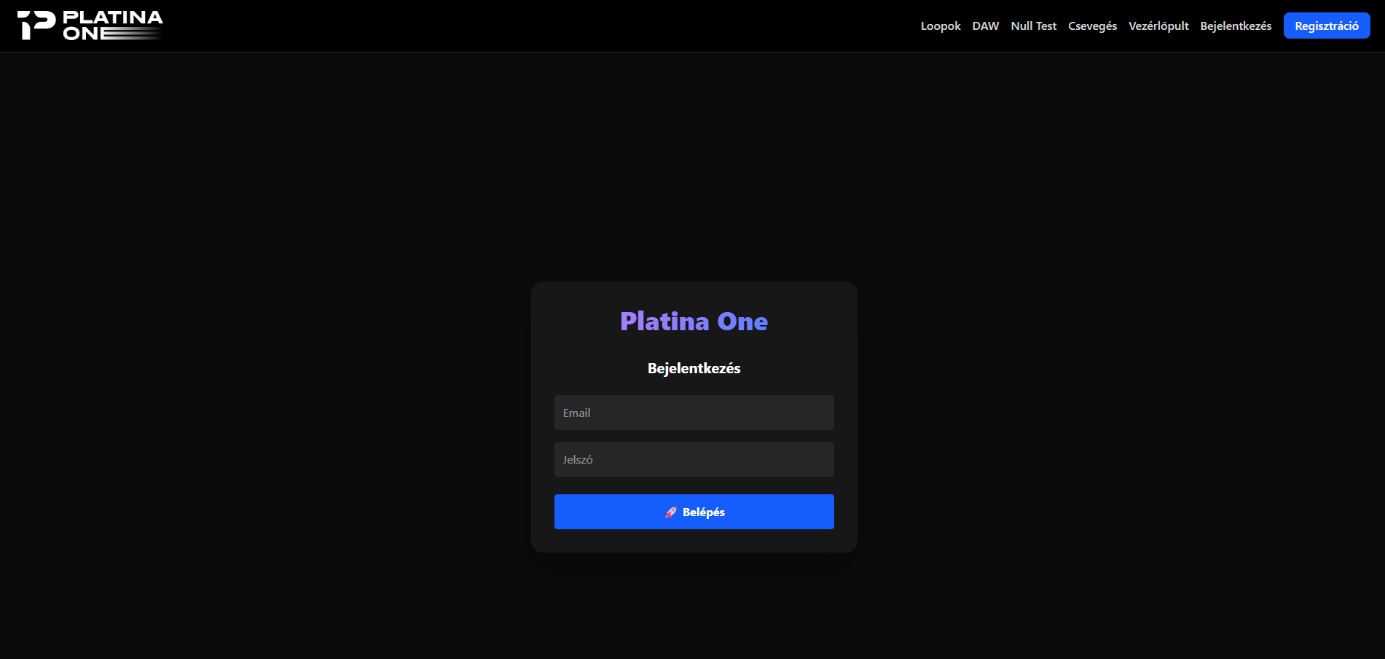
1. **E-mail + jelszó megadása**
2. **A backend összehasonlítja a beérkezett adatokat az adatbázisban tárolt, titkosított verzióval**
3. **Sikeres egyezés esetén JWT (JSON Web Token) kerül generálásra**
4. **A token a böngészőben kerül eltárolásra, és minden kéréshez csatolódik**

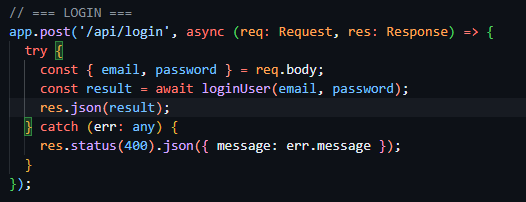
**Biztonság:**

* **Minden API endpoint csak érvényes token birtokában használható.**
* **A token lejárati idővel rendelkezik (pl. 24 óra).**

**Hibák:**

* **Rossz e-mail vagy jelszó: "Hibás belépési adatok."**
* **Token lejárata: automatikus kijelentkeztetés**





**# Dashboard Oldal Dokumentáció**

**## Célja**

**A `Dashboard` oldal a felhasználók központi felülete a bejelentkezés után. Gyors hozzáférést biztosít az alkalmazás kulcsfontosságú funkcióihoz, mint például a loopok böngészése, a DAW (Digital Audio Workstation) megnyitása, a felhasználói profil megtekintése, és a chat szoba elérése. Az oldal célja, hogy tiszta és intuitív felhasználói élményt nyújtson.**

**## Funkciók**

**### 1. \*\*Dinamikus Oldalcím\*\***

**- Az oldalcím dinamikusan `Platina One | Vezérlőpult`-ra van állítva a `document.title` segítségével.**

**- Ez biztosítja, hogy a böngésző fülén az aktuális oldal kontextusa jelenjen meg.**

**### 2. \*\*Felhasználói Hitelesítés\*\***

**- Az oldal ellenőrzi, hogy a felhasználó be van-e jelentkezve a `localStorage`-ból lekért `user` objektum alapján.**

**- Ha nincs bejelentkezett felhasználó, az oldal átirányít a bejelentkezési oldalra (`/login`).**

**### 3. \*\*Üdvözlő Üzenet\*\***

**- Személyre szabott üdvözlő üzenetet jelenít meg a felhasználó felhasználónevével.**

**- Példa: `Üdvözöllek, [felhasználónév]! 🎶`**

**### 4. \*\*Akciógombok\*\***

**Az oldal négy fő akciógombot tartalmaz, amelyek az alkalmazás különböző részeire vezetnek:**

**#### a. \*\*Loopok Böngészése\*\***

**- Gomb szövege: `🎧 Böngéssz Loopokat`**

**- A `/loops` oldalra navigál, ahol a felhasználók böngészhetik az elérhető loopokat.**

**#### b. \*\*DAW Megnyitása\*\***

**- Gomb szövege: `🎚️ Nyisd meg a DAW-ot`**

**- A `/daw` oldalra navigál, ahol a felhasználók zenei projekteket hozhatnak létre vagy szerkeszthetnek.**

**#### c. \*\*Profil Megtekintése\*\***

**- Gomb szövege: `👤 Saját profilod`**

**- A felhasználó profiloldalára navigál (`/profile/[felhasználónév]`), ahol személyes adataikat kezelhetik.**

**#### d. \*\*Chat Szoba\*\***

**- Gomb szövege: `💬 Chat szoba`**

**- A `/chat` oldalra navigál, ahol a felhasználók valós időben kommunikálhatnak egymással.**

**### 5. \*\*Reszponzív Dizájn\*\***

**- Az elrendezés reszponzív, alkalmazkodik a különböző képernyőméretekhez.**

**- Rácsrendszert használ, amely egy vagy két oszlopba rendezi a gombokat a képernyő szélességétől függően.**

**### 6. \*\*Stílus\*\***

**- Az oldal modern dizájnt alkalmaz:**

**- Színátmenetes szöveg az üdvözlő üzenethez.**

**- Lekerekített gombok hover-effektekkel és átmenetekkel.**

**- Sötét téma semleges és élénk színű kiemelésekkel.**

**## Kód Áttekintése**

**### Állapotkezelés**

**- A `useState` a `user` állapot kezelésére szolgál.**

**### Hatások**

**- Két `useEffect` van használatban:**

**1. Az oldalcím beállítására.**

**2. A bejelentkezett felhasználó ellenőrzésére és az átirányítás kezelésére.**

**### Feltételes Renderelés**

**- Ha nincs felhasználó, a komponens `null`-t ad vissza, hogy megakadályozza az oldal tartalmának renderelését.**

**### Navigáció**

**- A `react-router-dom`-ból származó `useNavigate` programozott navigációt biztosít a gombok kattintásakor.**

**## Összegzés**

**A `Dashboard` oldal jól strukturált és felhasználóbarát belépési pont az alkalmazás fő funkcióinak eléréséhez. Dinamikus és reszponzív kialakítása zökkenőmentes élményt biztosít a felhasználók számára.**

**3. Loop feltöltés**

**Célja: A kreatív munkáid megosztása más zenészekkel. A feltöltött loopokat a rendszer tárolja, indexeli és elérhetővé teszi minden felhasználó számára.**

**Támogatott formátumok: MP3, WAV, OGG Maximális fájlméret: 25MB**

**Lépések:**

1. **Jelentkezz be.**
2. **Navigálj a „Loopok” menüpontra.**
3. **Add meg a loop nevét (pl. „TrapDrums01”).**
4. **(Opcionálisan) Írd le, milyen hangulata van, vagy hogyan használd (pl. „dark trap, 140 BPM”).**
5. **Válaszd ki a fájlt a számítógépedről.**
6. **Kattints a „Feltöltés” gombra.**

**Mi történik a háttérben:**

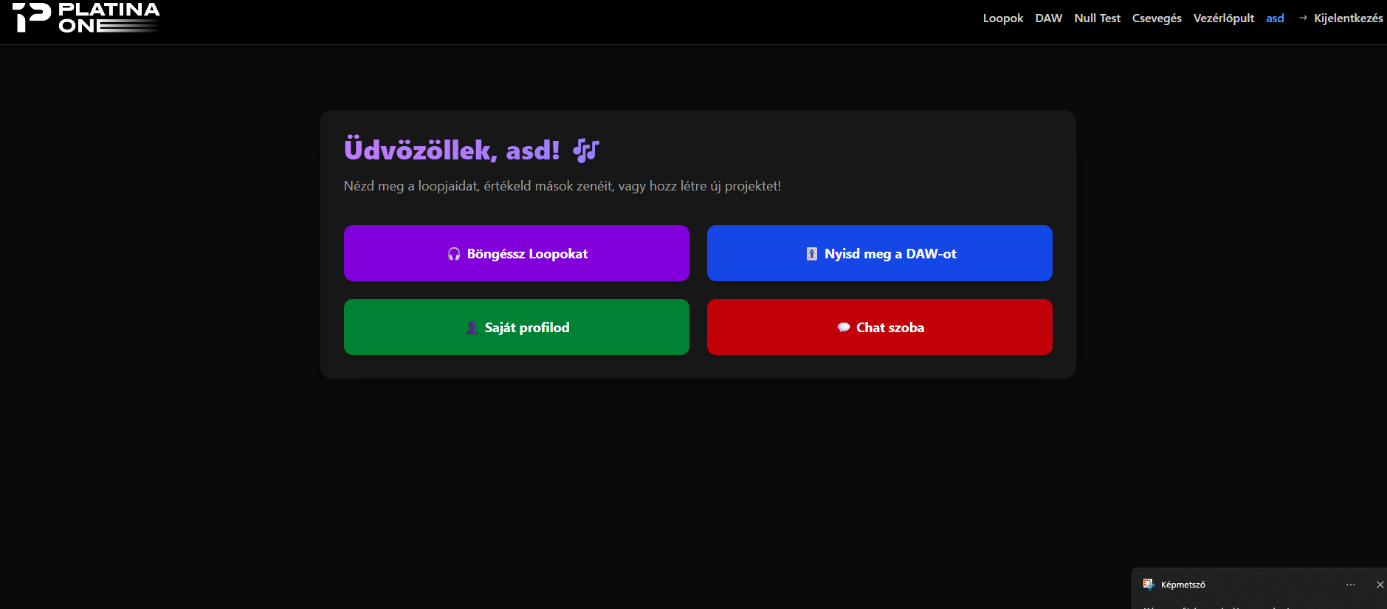
* **A frontend létrehoz egy FormData objektumot.**
* **A fájl és a metaadatok elküldésre kerülnek a /api/loops/upload endpointon keresztül.**
* **A szerver a multer csomaggal feldolgozza a fájlt, elmenti az uploads/ könyvtárba.**
* **Az adatbázisban egy új rekord keletkezik a loops táblában.**

**Tipp:**

* **Ne adj meg azonos című fájlokat egymás után, mivel azok fájlrendszeri ütközést okozhatnak.**

**Hibakezelés:**

* **Ha nincs fájl vagy cím megadva → alert üzenet: „Adj meg címet és válassz fájlt!”**
* **Ha a fájl mérete túl nagy → hibaüzenet: „A fájl túl nagy.”**



**4. Loopok böngészése és lejátszása**

**Célja:** A platform egyik legfőbb funkciója, hogy a felhasználók megismerjék és meghallgassák mások által feltöltött loopokat. Ez nemcsak inspirációs forrás, de lehetőséget nyújt zenei együttműködésre is.

**Funkciók:**

* Böngészés cím, leírás vagy feltöltő szerint
* Hangfájlok azonnali meghallgatása beépített lejátszóval
* Vizualizált hullámforma (waveform) megjelenítés
* Hangerőszabályzás, időzítés, és idővonal

**Lépések a böngészéshez:**

1. Nyisd meg a „Loopok” oldalt.
2. Használhatod a keresőt, vagy egyszerűen görgethetsz a listán.
3. Minden loop kártya tartalmazza:
   * A címét
   * A készítő nevét
   * A feltöltés idejét
   * Egy rövid leírást (ha van)
   * Lejátszás gombot

**Lejátszás működése:**

* Az alkalmazás a Tone.js vagy a <audio> HTML elem segítségével játssza le a fájlt.
* Ha WaveSurfer is telepítve van, akkor látványos hullámformát rajzol a lejátszó mellé.

**Tipp:**

* Egyidejűleg több loop is lejátszható – például összehasonlításhoz.
* A lejátszás automatikusan megállítható, ha másik loopot indítasz el (ez beállítás kérdése).

**Hibakezelés:**

* Ha a fájl nem érhető el: „A loop fájlja nem található vagy hibás URL.”
* Ha a lejátszás nem indul el: „Engedélyezned kell a hanglejátszást a böngésződben.”

**Fejlesztés alatt álló funkciók:**

* Loop előnézeti időintervallum beállítása (pl. csak az első 15 másodperc lejátszása)
* BPM-alapú szűrés
* Kulcs szerinti szűrés (pl. C-minor, D-major)

**5. Loopok értékelése és kommentelése**

**Célja:** A felhasználók közötti interakció egyik legfontosabb része az értékelés és kommentelés. Ezek az eszközök nemcsak a tartalmak minőségének visszajelzésére szolgálnak, hanem hozzájárulnak a közösségi szellem kialakulásához, az alkotók motiválásához és a kiemelkedő loopok felfedezéséhez.

**Típusok:**

* **Értékelés:** három ikonikus kategória (csillag = arany, gyémánt, platina)
* **Kommentelés:** szöveges vélemény vagy visszajelzés az adott loophoz kapcsolódóan

**Értékelési rendszer részletei:**

* ⭐ **Arany**: jó minőségű, használható alap
* 💎 **Gyémánt**: kiemelkedő, kreatív ötlet
* 💿 **Platina**: professzionális szintű, kiadásra kész loop



**Lépések az értékeléshez:**

1. Jelentkezz be.
2. Nyisd meg a kiválasztott loop adatlapját.
3. Válassz egy értékelési ikont.
4. Az értékelés azonnal mentésre kerül (vagy frissül, ha már értékelted).

**Lépések a kommenteléshez:**

1. Ugyanitt írhatsz szöveges megjegyzést is.
2. Kattints a „Komment küldése” gombra.
3. A hozzászólás megjelenik a loop alatt, a neveddel és időbélyeggel együtt.

**Hogyan működik a háttérben:**

* Az értékelések a ratings adatbázistáblába kerülnek mentésre.
* Ha ugyanaz a felhasználó új értékelést ad, akkor frissül a meglévő rekord.
* A kommentek szintén ebbe a táblába kerülnek, opcionális mezőként.

**Adatstruktúra példa:**

{

"user\_id": "u123",

"loop\_id": "l456",

"rating": 3,

"comment": "Ez egy brutál drill alap, grat!",

"created\_at": "2025-05-12T14:45:00Z"

}

**Megjelenítés:**

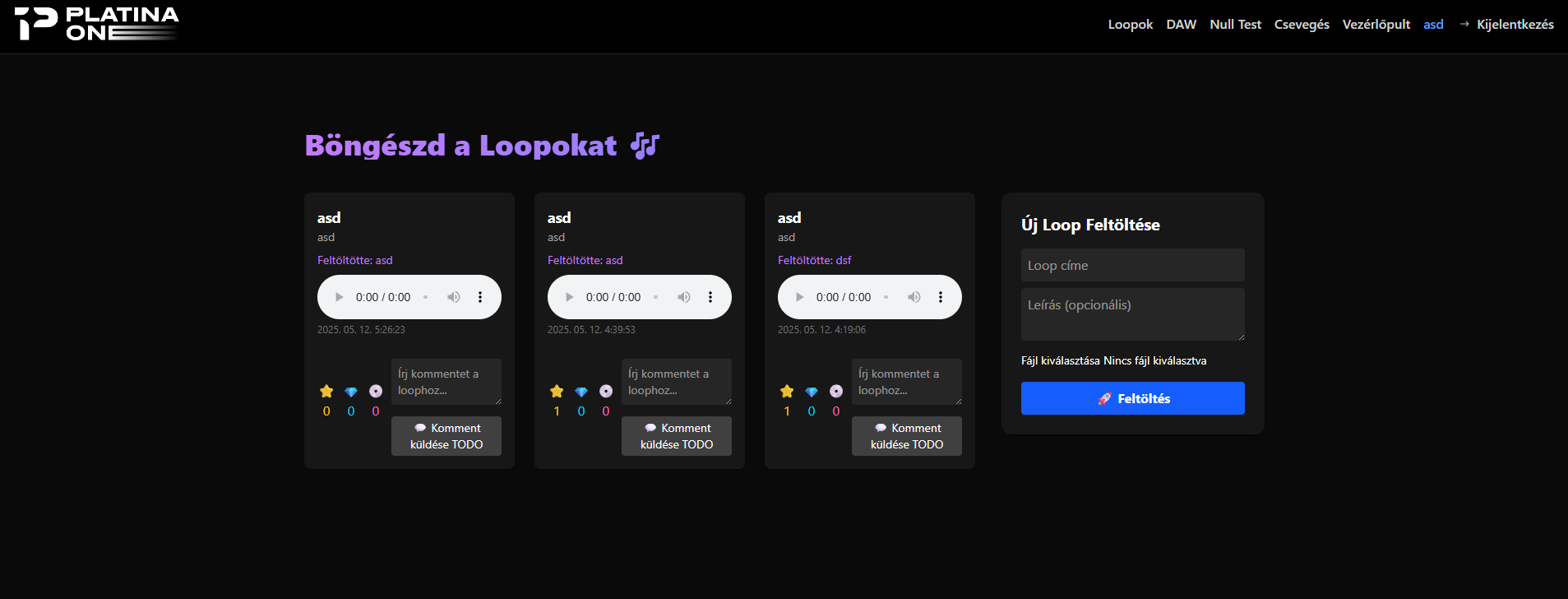
* A loop adatlapján megjelenik az összes hozzászólás időrendben
* A rendszer összesíti az értékelések számát típusonként (pl. 12 platina, 5 gyémánt, 8 arany)

**Hibakezelés:**

* Be nem jelentkezett felhasználó nem értékelhet → "Be kell jelentkezned."
* Üres komment → "Írj be egy szöveget a küldéshez."

**Jövőbeli funkciók:**

* Kommentek like/dislike rendszere
* Értékelések súlyozása (pl. megbízhatóbb felhasználók nagyobb súlyt kapnak)
* Értékelések alapján ranglista generálása (Top10 loop)



**6. Chat és közösségi funkciók**

**Célja:** A Platina One nem csupán egy loop megosztó felület, hanem egy közösségi platform is, amely lehetőséget biztosít az alkotók közötti közvetlen kommunikációra, együttműködésre és tapasztalatcserére. A chat funkció segíti a kapcsolatteremtést, közös projektek indítását és általános eszmecserét is.

**Fő funkciók:**

* Privát üzenetküldés (1:1 felhasználók között)
* Nyilvános közösségi chat (pl. „Lobby” vagy „Közös szoba”)
* Értesítések új üzenetről
* Időbélyeg és felhasználónév megjelenítés

**Privát üzenetküldés:**

1. Navigálj a „Chat” menüpontba.
2. Válaszd ki a kívánt felhasználót a listából (vagy keress rá).
3. Írd be az üzenetet.
4. Kattints a „Küldés” gombra.
5. Az üzenet azonnal megjelenik a beszélgetésben.

**Közösségi chat használata:**

1. A „Chat” oldalon válaszd ki a nyilvános szobát.
2. Itt mindenki láthatja mindenki üzenetét.
3. A válaszok időrendben, automatikus frissítéssel jelennek meg.

**Hogyan működik a háttérben:**

* A frontend websocket kapcsolatot létesít a szerverrel.
* A bejövő üzenetek a felhasználó ID-jával és időbélyeggel együtt kerülnek mentésre az adatbázisba.
* Privát és nyilvános üzenetek is külön táblába vagy megjelöléssel kerülnek mentésre.

**Adatstruktúra példa (privát üzenet):**

{

"sender\_id": "u123",

"receiver\_id": "u456",

"message": "Szia, csináljunk közösen egy alapot?",

"created\_at": "2025-05-12T18:30:00Z"

}

**Tipp:**

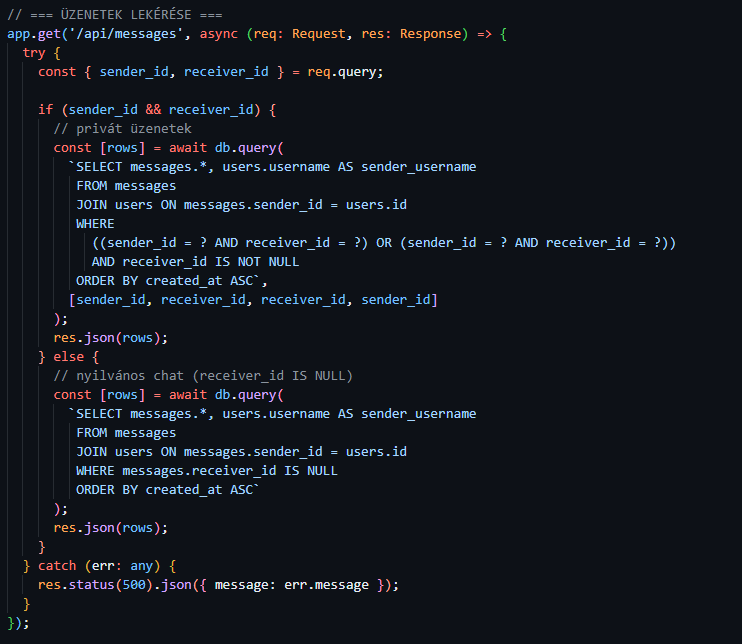
* A chat mobilon is reszponzívan működik.
* A privát beszélgetések titkosítottan tárolhatók a jövőben (pl. end-to-end encryption).

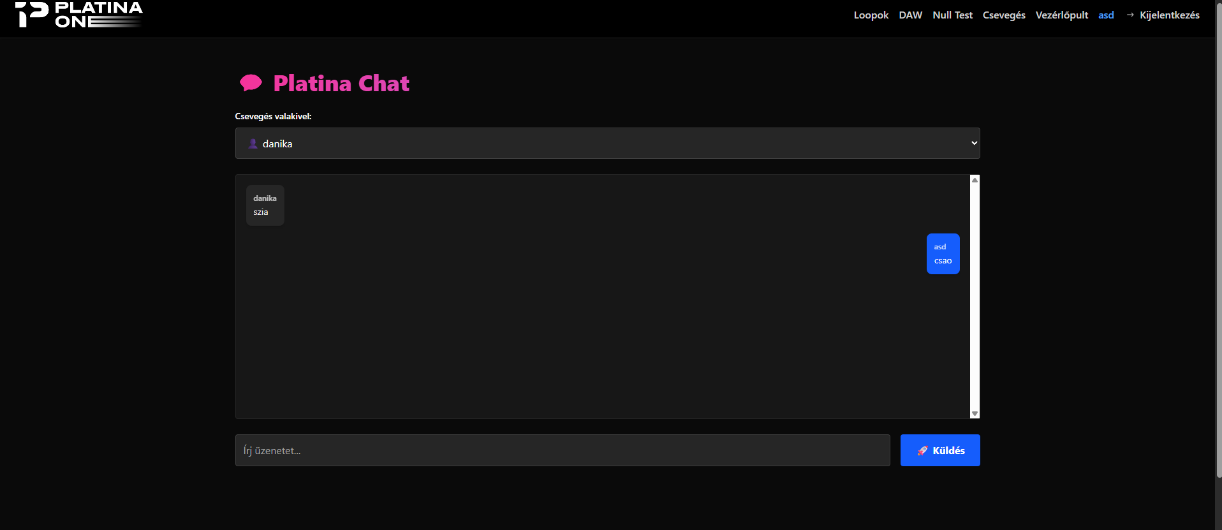
**Hibakezelés:**

* Ha a kapcsolat megszakad: „Nem sikerült kapcsolódni a szerverhez.”
* Ha a felhasználó nem választható: „Nem található ilyen felhasználó.”

**Jövőbeli bővítések:**

* Chat státusz (online/offline)
* Hang- és videóüzenetek
* Fájlmegosztás a chatben
* Moderációs funkciók (pl. jelentés, tiltás)







**7. Fejlesztői Dokumentáció – Backend API-k részletes bemutatása**

A backend architektúra kulcsszerepet játszik a Platina One rendszerben, hiszen ez felel a felhasználók hitelesítéséért, a fájlok tárolásáért, adatbázisműveletekért, valamint a RESTful interfészen keresztüli kommunikációért. A backend Node.js + Express keretrendszerre épül, és MySQL adatbázissal dolgozik. Az API endpointok jól strukturáltak és könnyen bővíthetők.

**Regisztráció – POST /api/register**

**Leírás:** Új felhasználó létrehozása. A regisztráció során az adatokat validáljuk, majd a jelszót titkosítjuk (bcrypt) és elmentjük az adatbázisba.

**Bemeneti JSON:**

{

"username": "beatmaker24",

"email": "beat@loop.com",

"password": "Titkos123"

}

**Válasz:**

{ "message": "User registered successfully" }

**Tipikus hibák:**

* Duplikált e-mail cím
* Hibás formátumú mezők
* Hiányzó adatok

**Bejelentkezés – POST /api/login**

**Leírás:** Felhasználó hitelesítése JWT token generálással. A sikeres belépés után a frontend ezt a tokent használja minden további védett kéréshez.

**Bemeneti JSON:**

{

"email": "beat@loop.com",

"password": "Titkos123"

}

**Válasz:**

{ "token": "eyJhbGciOiJI..." }

**Megjegyzés:** A token érvényessége típikusan 1-2 nap, és a felhasználó azonosítója is szerepel benne payloadként.

**Loop feltöltés – POST /api/loops/upload**

**Leírás:** Multipart/form-data formátumban történik. A fájlokat a multer middleware fogadja és a uploads/ mappába menti.

**FormData mezők:**

* loop: fájl (audio)
* title: szöveg
* description: szöveg (opcionális)
* user\_id: szöveg vagy szám

**Szerver válasz:**

{ "message": "Loop uploaded successfully" }

**Szerveroldali műveletek:**

1. Ellenőrzi a fájl típusát és méretét
2. Létrehozza a fájl nevét egyedi azonosítóval
3. Bejegyzést készít az adatbázisban

**Loopok listázása – GET /api/loops**

**Leírás:** Ez az endpoint lekéri az összes feltöltött loopot az adatbázisból. A válasz tartalmazza a loop metaadatait, a fájl URL-jét, a címet, a leírást, valamint a feltöltő felhasználónevet is. A lekérdezés INNER JOIN kapcsolaton keresztül vonja be a users táblát, hogy a felhasználói név is megjelenjen.

**Szerveroldali művelet:**

SELECT loops.\*, users.username FROM loops

JOIN users ON loops.user\_id = users.id

ORDER BY loops.created\_at DESC;

**Válasz példa:**

[

{

"id": 1,

"title": "Trap Melody 01",

"description": "Dark piano 140 BPM",

"file\_url": "http://localhost:3001/uploads/filename.mp3",

"username": "beatmaker24"

}

]

**Megjegyzés:**

* A fájl URL közvetlenül lejátszható <audio> vagy Tone.Player komponenssel a frontend oldalon.

**Fájlok kiszolgálása – GET /uploads/:filename**

**Leírás:** Ez az útvonal biztosítja a feltöltött fájlok elérhetőségét. A kérések során a szerver ellenőrzi, hogy a megadott fájl létezik-e, majd beállítja a megfelelő MIME típust és elküldi a tartalmat a kliensnek.

**Működés:**

* A szerver fs.existsSync() függvénnyel ellenőrzi a fájl létezését
* A mime csomag határozza meg a fájl típusát (pl. audio/mp3)
* A fájlt streameli vissza a kliens felé

**Tipikus válasz:**

* Siker: audio fájl tartalma
* Hiba: { "message": "File not found" }

**Biztonság:**

* A fájlnév szigorú validálása javasolt, hogy elkerüljük a path traversal típusú támadásokat

**Loop értékelések beküldése – POST /api/ratings**

**Leírás:** Lehetővé teszi, hogy egy felhasználó értékelje egy másik felhasználó által feltöltött loopot, és opcionálisan hozzászólást is írjon.

**Bemeneti JSON:**

{

"user\_id": "u123",

"loop\_id": "l456",

"rating": 3,

"comment": "Nagyon profi, szinte kész track!"

}

**Szerveroldal:**

* Ha létezik már értékelés ettől a felhasználótól az adott loopra, akkor UPDATE
* Ha nincs, akkor új INSERT

**Válasz:**

{ "message": "Rating/comment saved" }

**Hibák:**

* Hiányzó loop\_id vagy user\_id
* Érvénytelen értékelési érték (csak 1–3 lehet)

**Loop értékelések lekérdezése – GET /api/loops/:id/ratings**

**Leírás:** Összesíti, hogy az adott loopra hány arany (1), gyémánt (2) és platina (3) értékelés érkezett.

**Szerveroldali SQL:**

SELECT rating, COUNT(\*) as count FROM ratings

WHERE loop\_id = ?

GROUP BY rating;

**Példa válasz:**

{ "gold": 3, "diamond": 7, "platina": 10 }

**Loop kommentek lekérdezése – GET /api/loops/:id/comments**

**Leírás:** Lekéri az adott loophoz tartozó hozzászólásokat, beleértve a felhasználónevet és az időbélyeget.

**Példa válasz:**

[

{

"username": "loopfan99",

"comment": "Kész banger!",

"created\_at": "2025-05-12T14:45:00Z"

}

]

**8. Frontend architektúra és komponensek részletes ismertetése**

A frontend a Platina One projektben a modern fejlesztési eszközökre és könyvtárakra épül, különösen a következőkre:

* **React** (komponensalapú felépítés)
* **Vite** (gyors fejlesztői szerver és build tool)
* **TypeScript** (statikus típusosság)
* **Tailwind CSS** (használatra kész, utility-first stílusok)
* **Tone.js** (audio lejátszás és manipuláció)
* **WaveSurfer.js** (waveform megjelenítés)

**Alapvető mappa- és fájlstruktúra:**

src/

├── api/ # Backendhez kapcsolódó fetch függvények

├── components/ # Újrafelhasználható UI elemek

├── pages/ # Oldalszintű React komponensek (route-ok)

├── daw/ # DAW modul specifikus komponensek

├── utils/ # Segédfüggvények, konverziók

├── hooks/ # Egyedi React hookok

├── assets/ # Statikus képek, ikonok

└── types/ # Típusdefiníciók (pl. Loop, Rating, User)

**Példakomponensek és működésük:**

**LoopCard.tsx**

* Megjeleníti a loop nevét, leírását, feltöltőt
* Tartalmaz lejátszó gombot, értékelő ikonokat
* Inicializálja az AudioContext-et (pl. await Tone.start())
* Kezeli a lejátszási hibákat (onError handler)

**Loops.tsx**

* Betölti a loopokat a GET /api/loops végpontról
* Tartalmaz fájlfeltöltési formot
* Használja a FormData API-t új loop küldéséhez
* Újratölti a listát sikeres feltöltés után

**CommentSection.tsx**

* Megjeleníti a loophoz tartozó kommenteket
* Beküldő mező és gomb
* POST /api/ratings (kommenttel)

**Chat.tsx**

* Két fő nézet: privát üzenetek és nyilvános szoba
* Realtime frissítés websockettel
* Üzenetlista renderelése
* Üzenetküldő mező

**Állapotkezelés és adatfolyam**

* A legtöbb komponens useState és useEffect hookokat használ
* Adatok fetch-szel kerülnek lekérésre és frissítésre
* Token hitelesítés: localStorage vagy sessionStorage
* Adatok újratöltése: egyes komponensek setInterval-lal is frissítenek (pl. chat)

**Stílus és reszponzivitás**

* Minden komponens Tailwind CSS-sel formázott (pl. className="text-lg font-bold text-white")
* Mobilnézet támogatás (pl. Flex layoutok, breakpoints)
* Sötét mód később integrálható dark: prefixekkel

**9. DAW modul és haladó funkciók kifejtése**

A DAW (Digital Audio Workstation) modul a Platina One egyik leginnovatívabb és legösszetettebb része. Célja, hogy egy webes környezetben is professzionálisnak ható loopkezelési és hangmanipulációs élményt nyújtson. A DAW nem csupán egy egyszerű lejátszó, hanem egy vizuálisan és funkcionálisan is fejlett munkafelület, amely több sávos hangkezelést, idővonalas szerkesztést és interaktív funkciókat tesz lehetővé.

**Főbb jellemzők:**

* Többsávos hangkezelés (trackek)
* Waveform vizualizáció minden sávhoz
* Idővonal és playhead (lejátszási pozíció jelölése)
* Zoom funkció idővonalra
* Drag & drop hangfájl betöltés
* Saját fájlok behúzása (local vagy feltöltött)
* Lejátszás, stop, playhead követés

**Működés alapjai**

A DAW modul tipikusan több külön canvas vásznat használ:

* **Waveform vászon:** a WaveSurfer kirajzolja a hullámformát
* **Rácsvonal vászon:** a ritmikai beosztás (pl. 4/4 ütem) megjelenítése
* **Playhead vászon:** a mozgó lejátszási csík

Minden sáv külön Track komponens, amelyek dinamikusan generálódnak, ha a felhasználó loopot helyez el rajta.

**Kulcsfontosságú komponensek:**

**Daw.tsx**

* A fő munkafelület, amely koordinálja a play/stop állapotot, lejátszási pozíciót
* Tartalmaz minden sávot, canvasokat, timeline-t

**Track.tsx**

* Egyedi sáv, amelyhez egy vagy több loop csatolható
* Tartalmazza a hullámforma megjelenítést és a drag & drop logikát

**Timeline.tsx**

* Az idő skálázása pixelben
* Segít az igazításhoz, ütemhez kötéshez (snap-to-grid)

**Playhead.tsx**

* Egy animált vonal, amely mutatja a jelenlegi lejátszási pozíciót
* Frissül requestAnimationFrame segítségével

**Interaktív funkciók:**

* **Loop cut at playhead:** a kiválasztott loop kettévágása a lejátszó pozíciójánál
* **Duplikálás:** loop megkettőzése ugyanabban a sávban vagy másikban
* **Mozgatás:** egérrel húzva a loop elhelyezhető máshová
* **Kijelölés:** több loop kijelölése shift+kattintással

**Hangmotor**

A Tone.js Player objektuma kerül használatra a lejátszáshoz. Minden loophoz külön Tone.Player példány jön létre, amely szinkronban indul el egy globális Transport óra segítségével.

**Példa:**

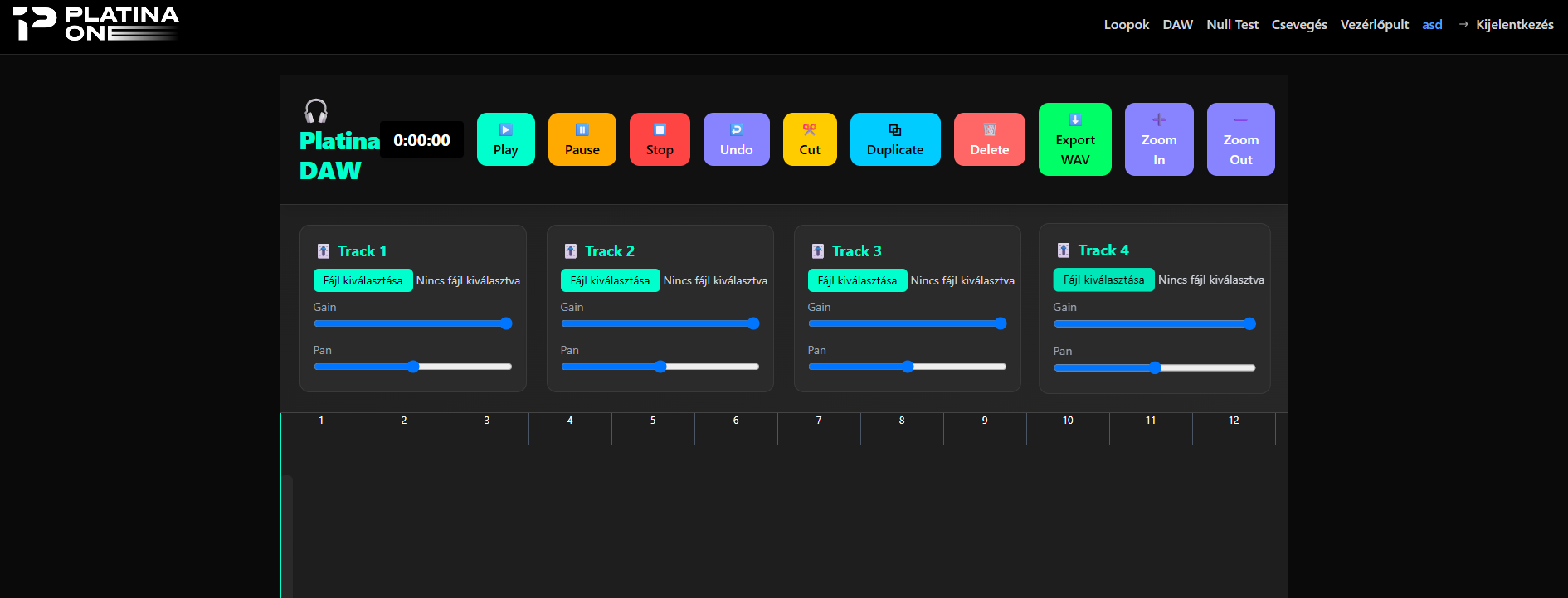
const player = new Tone.Player(url).toDestination();

player.sync().start(0);

Tone.Transport.start();

**Jövőbeli fejlesztési lehetőségek:**

* Loop vágás és crossfade
* Sávok némítása/solo mód
* Tempo és pitch manipuláció
* Saját presetek mentése (projekt fájl)
* Export WAV/MP3

chat

**10. Fejlesztési környezet beállítása, hibakezelés és technológiák bővebben**

A fejlesztési környezet megfelelő beállítása kulcsfontosságú a Platina One projekt gördülékeny és fenntartható fejlesztéséhez. A következő szakasz részletezi, hogyan lehet gyorsan elindítani a projektet, milyen eszközökre és beállításokra van szükség, valamint hogyan kezelhetők a leggyakoribb fejlesztési és futásidejű hibák.

**1. Alapvető környezeti feltételek**

**Minimális követelmények:**

* Node.js 18.x vagy újabb
* NPM vagy Yarn (javasolt: npm 9.x)
* MySQL 8.x adatbázisszerver
* Git (verziókezeléshez)
* Modern böngésző (pl. Chrome, Firefox) fejlesztői eszközökkel

**Ajánlott fejlesztői környezet:**

* Visual Studio Code
  + Bővítmények: ESLint, Prettier, Tailwind CSS IntelliSense, TypeScript Hero
* Docker (opcionálisan, adatbázis és backend izolálásához)

**2. Projekt telepítése és indítása**

# Repository klónozása

git clone https://github.com/felhasznalo/platina-one.git

cd platina-one

# Függőségek telepítése

npm install

# Szerver indítása

-npm run dev

-npx ts-node server.ts

**3. Adatbázis beállítása**

1. Hozz létre egy platina\_one nevű adatbázist:

CREATE DATABASE platina\_one;

1. Importáld a db.sql fájlt, amely tartalmazza a szükséges táblák létrehozását (users, loops, ratings, messages, stb.)
2. Konfiguráld a kapcsolódást a backendben (pl. .env fájl):

DB\_HOST=localhost

DB\_USER=root

DB\_PASS=yourpassword

DB\_NAME=platina\_one

**4. Hibakezelés fejlesztés közben**

**Gyakori problémák és megoldásaik:**

|  |  |
| --- | --- |
| **Probléma** | **Lehetséges ok / megoldás** |
| AudioContext was not allowed... | A felhasználónak előbb kattintania kell (Tone.start) |
| Failed to fetch /api/... | Backend nem fut vagy CORS probléma |
| Unknown column 'xyz' in field list | Elavult adatbázis – frissítsd a db.sql alapján |
| Unhandled Rejection (TypeError) | Nem nullcheckelt adat használata (pl. user?.id) |
| Fájl nem tölt be / loop URL hibás | Helytelen fájlnév, vagy nem fut a kiszolgálás /uploads |

**5. Használt technológiák részletesen**

**Frontend:**

* React (18.x)
* Vite (vite.config.ts)
* TypeScript (TSX komponensek)
* Tailwind CSS
* WaveSurfer.js (vizuális hullámforma)
* Tone.js (szinkronizált audio playback)

**Backend:**

* Node.js + Express
* MySQL + mysql2 csomag
* multer (fájlkezelés)
* jsonwebtoken (JWT tokenek)
* dotenv (környezeti változók)

**Egyéb:**

* Git + GitHub (verziókövetés)
* Postman (API teszteléshez)
* ESLint, Prettier (kódfelügyelet)

**11. Összegzés és jövőbeli fejlesztési irányok**

A Platina One projekt célja, hogy egy modern, felhasználóbarát és fejlesztőbarát környezetet biztosítson a zenei loopok kezeléséhez, megosztásához és kreatív együttműködéshez. Az eddig kialakított rendszer alapjai stabilak: a frontend és backend technológiai stack naprakész, jól dokumentált és bővíthető. Az alkalmazás funkciói – regisztráció, bejelentkezés, loop feltöltés, értékelés, kommentelés, chat, valamint a DAW modul – egy átfogó platform alapjait képezik.

**Jelenlegi állapot értékelése**

* ✅ Működő felhasználói hitelesítés és jogosultságkezelés
* ✅ Fájlfeltöltés és kiszolgálás
* ✅ Loopok listázása és értékelése
* ✅ Kommentelési lehetőség
* ✅ Közösségi chat funkció (privát és nyilvános)
* ✅ DAW modul alapverziója
* ✅ Teljes TypeScript és Tailwind CSS integráció

**Javasolt jövőbeli fejlesztési irányok**

**1. DAW modul bővítése**

* Vágás, fade-in/fade-out, loop stretching
* Tempo és hangnem szerinti igazítás
* Export lehetőség WAV vagy MP3 formátumban
* Preset/projekt mentés-funkció (pl. .platina fájl)

**2. Társasági funkciók fejlesztése**

* Profiloldal testreszabás (bio, képek, kedvenc loopok)
* Közösségi hírcsatorna
* Követés / értesítési rendszer

**3. Tartalom moderálás és biztonság**

* Jelentés funkció (pl. sértő komment, jogsértő tartalom)
* Admin panel
* IP-szűrés, korlátozás

**4. Teljesítmény optimalizálás**

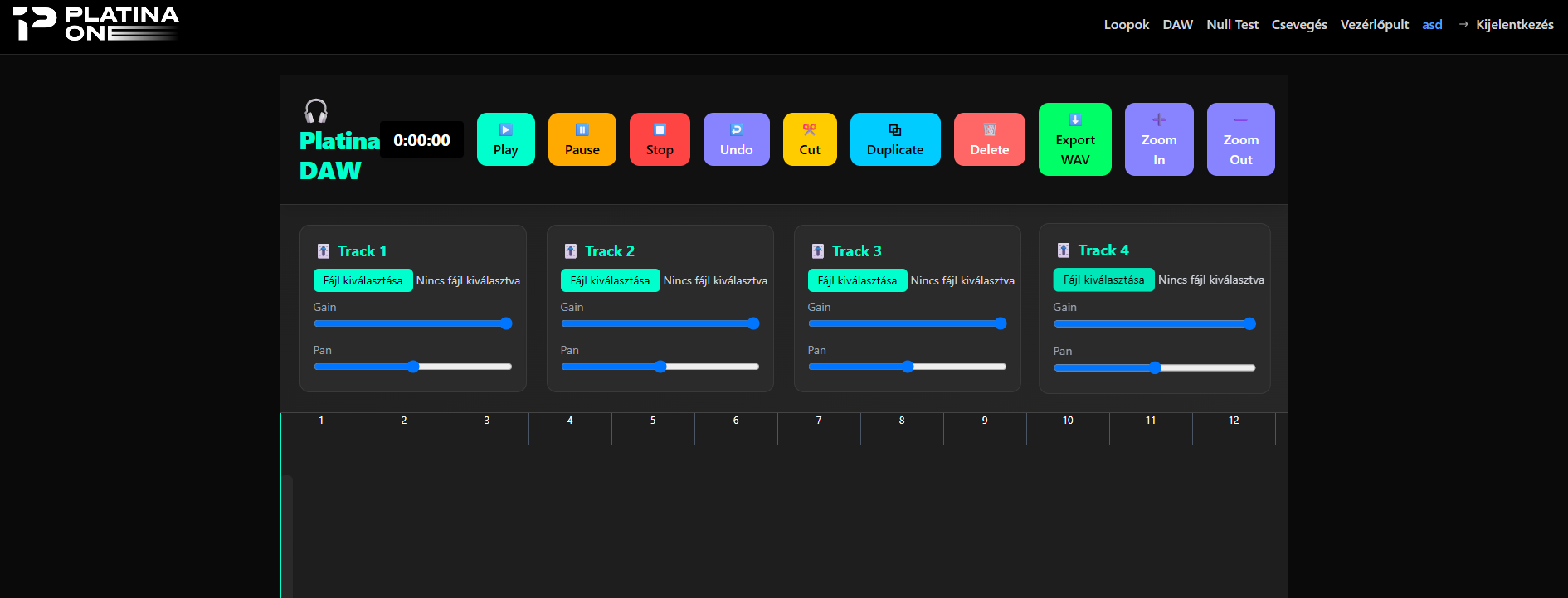
* Loop caching, lazy loading
* Mobil optimalizáció (pl. egyszerűsített DAW nézet)

**5. Nemzetközi elérhetőség**

* Többnyelvűség (i18n)
* Lokalizált dátum/időformátum

**6. Üzleti modellek és bővítmények**

* Freemium modell (alap + prémium DAW funkciók)
* Preset store (felhasználók értékesíthetik saját beállításaikat)
* Licenszelési rendszer a loopokhoz



**Záró gondolat**

A Platina One projekt nemcsak egy alkalmazás, hanem egy digitális közösségi tér, amely támogatja a zenei kreativitást, együttműködést és tanulást. A technikai háttér stabil alapot biztosít a folyamatos fejlesztéshez. A dokumentáció célja, hogy átláthatóvá és könnyen kezelhetővé tegye a rendszert a jövőbeli fejlesztők, üzemeltetők és felhasználók számára egyaránt.

A javasolt fejlesztési irányok követésével a Platina One hosszú távon is versenyképes és értékteremtő platformmá válhat a zeneipari ökoszisztémában.

**Vígh János-Székely Olivér-Géró Nikolasz Dominik**