ShazaPiano — Cursor Master Spec (ONE FILE)

Tout-en-un : UI sombre façon Shazam • Backend FastAPI • 4 niveaux + previews 16s • 1$ Unlock • Practice Mode • Firebase • IAP • CI/CD

Date: 31 Octobre 2025

0) But de ce document (pour Cursor)

Ce PDF fournit à Cursor une feuille de route **exécutable** : structure des dossiers, commandes, endpoints, tâches numérotées. Chaque bloc **TASK** est conçu pour être collé tel quel dans Cursor pour générer code et fichiers.

1) Résumé exécutif

- App: appui → capture 8s → backend génère 4 vidéos (Lvl1..4).
- Gratuit: previews 16s (4 tuiles). Payant: 1 $ débloque les 4 complètes.
- Practice Mode (MVP monophonique): détecte fausses notes via micro avec tolérance ±50 cents.
- Stack: Flutter + FastAPI + FFmpeg + BasicPitch + MoviePy (+ Fluidsynth optionnel) + Firebase (Auth/Firestore) + IAP Android.

2) Monorepo & Arborescence (à créer)

```
/shazapiano/ app/ # Flutter backend/ # FastAPI + ML + rendu infra/ # Docker, Nginx, CI/CD docs/ # Spécifications (ce PDF)
```

TASK 1 — Initialiser le mono-repo & dépendances

```
# Créer dossiers mkdir -p shazapiano/app shazapiano/backend shazapiano/infra shazapiano/docs cd shazapiano && git init # Flutter app cd app && flutter create --org com.ludo --project-name shazapiano . flutter pub add dio permission_handler record video_player go_router flutter_riverpod shared_preferences firebase_core firebase_auth cloud_firestore in_app_purchase # Backend cd ../backend && python -m venv .venv && source .venv/bin/activate pip install fastapi uvicorn[standard] numpy scipy soundfile librosa pretty_midi moviepy python-multipart torch basic-pitch pyfluidsynth
```

3) Flutter — Architecture & Pages (squelette)

```
app/lib/ core/ (config.dart, theme.dart, errors.dart) data/ datasources/ (piano_api.dart, firebase_ds.dart, prefs_ds.dart) models/ (dto) repositories/ domain/ entities/ (piano_video.dart, entitlements.dart, job.dart) repositories/ usecases/ (process_audio.dart, purchase_all_levels.dart, restore_purchases.dart) presentation/ state/ (providers.dart) pages/ (home.dart, previews.dart, paywall.dart, player.dart, practice.dart) widgets/ (big_record_button.dart, mode_chip.dart, video_tile.dart, paywall_modal.dart)
```

TASK 2 — Thème sombre & écran Home (Shazam-like)

```
// lib/core/theme.dart import 'package:flutter/material.dart'; final darkTheme = ThemeData( brightness: Brightness.dark, colorScheme: ColorScheme.fromSeed(seedColor: Color(0xFF2AE6BE), brightness: Brightness.dark), scaffoldBackgroundColor: Color(0xFF0B0F10), useMaterial3: true, );
```

```
// lib/presentation/pages/home.dart (simplifié) import 'package:flutter/material.dart'; class HomePage extends StatelessWidget { const HomePage({super.key}); @override Widget build(BuildContext context) { final color = Theme.of(context).colorScheme.primary; return Scaffold( body: Center( child: GestureDetector( onTap: () {/* start record → upload → navigate to previews */}, child: Container( width: 220, height: 220, decoration: BoxDecoration( shape: BoxShape.circle, gradient: LinearGradient(colors: [color, color.withOpacity(0.6)]), boxShadow: [BoxShadow(color: color.withOpacity(0.45), blurRadius: 30, spreadRadius: 4)], ), child: const Icon(Icons.mic_rounded, size: 88), ), ), ), ); } }
```

TASK 3 — Grille Previews 2×2 (16s)

```
// lib/presentation/pages/previews.dart (concept) class PreviewsPage extends StatelessWidget { final List items; // 4 niveaux const PreviewsPage({super.key, required this.items}); @override Widget build(BuildContext context) { return Scaffold( appBar: AppBar(title: const Text('Aperçus (16s)')), body: GridView.count( crossAxisCount: 2, childAspectRatio: 16/9, padding: const EdgeInsets.all(12), crossAxisSpacing: 12, mainAxisSpacing: 12, children: items.map((v) => VideoTile(video: v, preview: true)).toList(), ), bottomNavigationBar: Padding( padding: const EdgeInsets.all(12), child: ElevatedButton(onPressed: () {/* show paywall */}, child: const Text('Débloquer les 4 pour 1 $')), ), ); } }
```

4) Backend — Endpoints & Fichiers

```
backend/ app.py # routes, /process, /health, static mounts inference.py # ffmpeg_to_wav(), basicpitch_predict() arranger.py # arrange(level) -> PrettyMIDI render.py # keyboard_video_from_midi() (full + preview 16s, audio optionnel) config.py # LEVELS + chemins + limites media/in|out/ # I/O assets/piano.sf2 (option)
```

TASK 4 — Spécification API unique

```
POST /process (multipart: file=audio, query: with_audio=false) → Génère 4 niveaux en parallèle et renvoie: [L1..L4] Réponse: [ {"level":1,"preview_url":"/media/out/uid_L1_preview.mp4","video_url":"/media/out/uid_L1_full.mp4","midi_url":"/media/out/uid_L1.mid","key_guess":"G","tempo_guess":96,"duration_sec":27.2}, {"level":2,...}, {"level":3,...}, {"level":4,...} ]
```

TASK 5 — Presets des 4 niveaux (config.py)

```
LEVELS = { 1: dict(transpose_to_C=True, quant=0.25, mg='none', md='none', arp=False, tempo_mul=0.8, poly=False, range=(60,79)), 2: dict(transpose_to_C=True, quant=0.125, mg='root', md='none', arp=False, tempo_mul=0.9, poly=False, range=(48,84)), 3: dict(transpose_to_C=False, quant=0.125, mg='root5', md='triad', arp=False, tempo_mul=1.0, poly=True, range=(36,84)), 4: dict(transpose_to_C=False, quant=0.0625,mg='arp158', md='triad_brk', arp=True, tempo_mul=1.0, poly=True, range=(36,84)), }
```

TASK 6 — Practice Mode (MVP)

```
Micro PCM → Pitch (YIN/MPM) → MIDI pitch → matching timeline (MIDI attendu). Tolérances: ±50 cents; onset ±120 ms; min dur 80 ms. Feedback: CORRECT (vert), CLOSE (jaune), WRONG (rouge); score cumulatif. Packages: flutter_fft (ou mic_stream + algo custom). Latence < 120 ms.
```

5) Firebase — Setup & Règles

```
- Créer projet; activer Auth (Anonymous); Firestore (prod). - Ajouter app Android (com.ludo.shazapiano); placer google-services.json dans app/android/app/. - Data: users/{uid} entitlements: { allLevels: true|false, purchasedAt } history/{videoId}: { level, durationSec, keyGuess, createdAt, videoUrl } - Règles: rules_version = '2'; service cloud.firestore { match /databases/{database}/documents { function isSignedIn() { return request.auth != null; } match /users/{uid} { allow read, write: if isSignedIn() && uid == request.auth.uid; match /entitlements/{doc} { allow read, write: if isSignedIn() && uid == request.auth.uid; } match /history/{doc} { allow read, write: if isSignedIn() && uid == request.auth.uid; } } match /{document=**} { allow read, write: if false; } } }
```

6) IAP — 1 $ Unlock (Android)

```
- Play Console: créer produit non■consommable unique: piano_all_levels_1usd. - Flutter (in_app_purchase): 1) queryProductDetails(['piano_all_levels_1usd']) 2) buyNonConsumable(productDetails) 3) on purchaseStatus == purchased →
```

```
acknowledge + set entitlements (Prefs + Firestore) 4) restore purchases au
démarrage - UI: PaywallModal, bouton “Restaurer l’achat”.
```

7) Sécurité & Performance backend

```
Nginx: rate-limit 20 req/min/IP; client_max_body_size 10M. Timeouts:
FFmpeg 15s; BasicPitch 10s; Render 20s. Purge: /media/in > 1j, /media/out >
7j (cron). Parallelisme: limiter jobs (semaphore). /health pour readiness;
logs JSON latences.
```

8) CI/CD & Déploiement

```
GitHub Actions: - Flutter: analyze, test, build APK/AAB. - Backend: pytest,
build Docker, deploy Fly.io/Railway/VPS. Dockerfile backend:
python:3.11-slim + apt-get ffmpeg fluidsynth; pip install -r
requirements.txt; uvicorn.
```

9) Critères d’acceptation (QA)

- Tap unique → 4 previews 16 s visibles en < 60 s.
- Achat 1 $ → déblocage instantané des 4 versions complètes + persistance (Prefs + Firestore).
- Practice: feedback correct/close/wrong, latence < 120 ms, score affiché.
- Backend: /process renvoie 4 objets (preview/full/midi/tempo/key), erreurs gérées proprement.

10) Ordre d’exécution des tâches (Cursor)

1 TASK 1 — Init monorepo & deps.

2 TASK 4 — Backend /process + dossiers media + presets LEVELS + previews 16s.

3 TASK 2 — Thème sombre + Home (bouton).

4 TASK 3 — Previews 2×2 (consomme preview_url du backend).

5 IAP — Paywall (1$) + entitlements.

6 Practice Mode — Pitch détect + HUD (MVP).

7 CI/CD — Workflows + Docker backend.

Ce PDF est la source unique de vérité pour Cursor. Colle les blocs TASK successivement et implémente.