

# RÉCAPITULATIF PAGE FAST PASS - AMBIANCE+

## OBJECTIF DE LA PAGE

Permettre aux clients de payer pour faire passer leurs morceaux en priorité, avec un système d'enchères qui maximise les revenus tout en garantissant une expérience DJ contrôlée et un délai client respecté.

---

## INTERFACE & DESIGN

### Design Visuel

- **Couleurs** : Vert néon ( #00FF41) sur fond noir ( #000000)
- **Police** : Monospace (Monaco, SF Mono, Courier New)
- **Style** : Terminal/Matrix minimaliste
- **Effets visuels** : Sur-illumination hover (glow effect intense)
- **Icônes** : Éclair  uniquement (cohérence avec titre)
- **Animation** : Points clignotants pour enchères live

### Structure Interface

← RETOUR FAST PASS

← Header avec navigation

TABLE 12

← Info client (ou ZONE BAR #347)

⚡ PASSER UN MORCEAU

← Titre section

Coller URL YouTube:

← Label input

[ ] | ← Input URL (focus principal)

Ex: youtube.com/watch?v=... | ← Exemple format



PAYER AU PRIX ACTUEL | | ← Option 1 : Standard

8€

🚀 PAYER PLUS POUR

← Option 2 : Surenchère

PASSER DEVANT LES AUTRES | |

[ ] € [+][-]

← Sélecteur montant

[PAYER \_\_€]

← Bouton dynamique

Restant: 2/3 Fast Pass | ← Limite soirée

⌚ Temps estimé: 30min | ← Garantie temporelle

## 🎵 LOGIQUE FAST PASS

🔗 Sélection Morceau : URL YouTube UniqueMent

## Validation URL

javascript

```
function validateYouTubeURL(url) {  
    const patterns = [  
        /youtube\.com\/watch\?v=([a-zA-Z0-9_-]+)/,  
        /youtu\.be\/([a-zA-Z0-9_-]+)/,  
        /m\.youtube\.com\/watch\?v=([a-zA-Z0-9_-]+)/  
    ];  
  
    return patterns.some(pattern => pattern.test(url));  
}
```

## Extraction Métadonnées

javascript

```
async function extractYouTubeInfo(url) {  
    const videoid = extractVideoid(url);  
    const response = await fetch(`/api/youtube/info/${videoid}`);  
    return {  
        title: response.title,  
        artist: response.artist || "Artiste inconnu",  
        duration: response.duration,  
        thumbnail: response.thumbnail,  
        isAvailable: response.available  
    };  
}
```



## Tarification Simplifiée

### Prix de Base Uniforme

javascript

```
function getPrixBase() {  
    return 12; // Prix fixe toute la soirée, toutes plages horaires  
}
```

## Système Enchères Sans Limite

javascript

```
function getCurrentMinimumBid() {  
    const currentHighestBid = getHighestActiveBid();  
    return currentHighestBid ? currentHighestBid + 1 : 13; // Base + 1€ minimum  
}  
  
// Pas de limite maximale - libre arbitre client  
function validateBidAmount(amount) {  
    const minimum = getCurrentMinimumBid();  
    return amount >= minimum; // Aucune limite haute  
}
```

## ⌚ Animation Enchères Temps Réel

### Animation Points Clignotants

javascript

```
function animateEnchere() {  
    const enchereElement = document.querySelector('.enchere-loading');  
    const baseText = 'Enchère en cours';  
    let dotCount = 0;  
  
    setInterval(() => {  
        dotCount = (dotCount + 1) % 4; // 0, 1, 2, 3, puis retour à 0  
        const dots = '.'.repeat(dotCount);  
        enchereElement.textContent = baseText + dots;  
    }, 500); // Changement toutes les 500ms  
}  
  
// Résultat visuel:  
// "Enchère en cours"  
// "Enchère en cours."  
// "Enchère en cours.."  
// "Enchère en cours..."  
// (puis recommence)
```

## Mise à Jour Enchères Live

javascript

```

// WebSocket pour updates temps réel
socket.on('fast_pass_bid_update', (data) => {
  if (data.highest_bid > 12) {
    showEnchereStatus(data.highest_bid, data.bidder_info);
    updateMinimumBid(data.highest_bid + 1);
  } else {
    hideEnchereStatus();
    updateMinimumBid(13);
  }
});

function showEnchereStatus(amount, bidder) {
  document.querySelector('.enchere-status').style.display = 'block';
  document.querySelector('.enchere-details').textContent = `${amount}€ - ${bidder}`;
  animateEnchere(); // Redémarre l'animation
}

```

## ⌚ SYSTÈME TEMPOREL & GARANTIES

### Timeline Fast Pass

Client paie → DJ reçoit (instantané) → DJ valide → Timer 30min démarre →  
Morceau joué OU Expiration + remboursement

### ⌚ Gestion Timer 30 Minutes

#### Démarrage Timer

javascript

```

function startDJTimer(requestId) {
  const request = {
    id: requestId,
    validated_at: Date.now(),
    deadline: Date.now() + (30 * 60 * 1000), // +30min
    status: 'validated_pending'
  };
  // Timer côté serveur
  setTimeout(() => {
    handleExpiredRequest(requestId);
  }, 30 * 60 * 1000);

  return request;
}

```

## Countdown Temps Réel

javascript

```

function updateDJCountdown(request) {
  const remaining = request.deadline - Date.now();

  if (remaining <= 0) {
    return "EXPIRÉ";
  }

  const minutes = Math.floor(remaining / (60 * 1000));
  const seconds = Math.floor((remaining % (60 * 1000)) / 1000);

  return `${minutes}min ${seconds}sec`;
}

```

## ⚠️ Gestion Expiration Automatique

javascript

```
async function handleExpiredRequest(requestId) {
    const request = await db.fastpass.findById(requestId);

    // 1. Remboursement automatique Stripe
    await stripe.refunds.create({
        payment_intent: request.payment_intent_id,
        reason: 'requested_by_customer'
    });

    // 2. Update statut
    await db.fastpass.update(requestId, {
        status: 'expired_refunded',
        refunded_at: Date.now()
    });

    // 3. Notifications
    await sendClientNotification(request.client_id, {
        type: 'fast_pass_expired',
        title: '⌚ Fast Pass expiré',
        message: 'Remboursement automatique effectué',
        amount: request.montant
    });

    await sendDJAlert({
        type: 'fast_pass_expired',
        message: `Fast Pass "${request.title}" expiré - remboursement auto`
    });
}
```

## WORKFLOW PAIEMENT DOUBLE

### Option 1 : Prix Standard

javascript

```
async function payStandardPrice() {
    const montant = getPrixActuel();

    try {
        // 1. Création PaymentIntent
        const paymentIntent = await stripe.paymentIntents.create({
            amount: montant * 100,
            currency: 'eur',
            capture_method: 'manual', // Capture après validation DJ
            metadata: {
                type: 'fast_pass_standard',
                club_id: clubId,
                client_id: clientId
            }
        });

        // 2. Apple Pay / Google Pay
        const result = await stripe.confirmPayment({
            elements,
            confirmParams: { return_url: window.location.href }
        });

        if (result.paymentIntent.status === 'requires_capture') {
            await submitFastPassRequest(paymentIntent.id, montant);
        }
    } catch (error) {
        showPaymentError(error);
    }
}
```



## Option 2 : Surenchère

javascript

```
async function paySurencherePrice(montantCustom) {  
    const minAmount = getPrixActuel();  
  
    if (montantCustom < minAmount) {  
        throw new Error(`Montant minimum: ${minAmount}€`);  
    }  
  
    // Même workflow que standard mais avec montant personnalisé  
    return await processPayment(montantCustom, 'fast_pass_surenchere');  
}
```

## ■ Interface Paiement Optimisée Mobile

### Saisie Montant Surenchère

javascript

```
// Input avec clavier numérique automatique  
<input  
    type="number"  
    inputmode="numeric"  
    min="48"  
    class="amount-input"  
/>  
  
// CSS suppression flèches natives  
.amount-input::-webkit-outer-spin-button,  
.amount-input::-webkit-inner-spin-button {  
    -webkit-appearance: none;  
    margin: 0;  
}
```

### Contrôles Tactiles

javascript

```
// Boutons + et - pour ajustements fins
function adjustAmount(delta) {
    const current = parseInt(amountInput.value);
    const newAmount = Math.max(current + delta, getMinimumBid());

    amountInput.value = newAmount;
    updatePayButton(newAmount);
}

// Saisie libre pour montants importants
amountInput.addEventListener('input', () => {
    const value = parseInt(amountInput.value);
    if (value >= getMinimumBid()) {
        updatePayButton(value);
    }
});
```

## Effets Visuels Mobiles

css

```
.payment-option:hover {
    background: #00FF4125;
    box-shadow: 0 0 15px #00FF41; /* Sur-illumination intense */
    border-color: #00FF41;
    transform: scale(1.02); /* Léger zoom pour feedback tactile */
}
```



## INTERFACE DJ



## Dashboard DJ Fast Pass

## FAST PASS MANAGEMENT



## 🎵 Fonctionnalités DJ

### Preview Audio

javascript

```
async function previewTrack(youtubeUrl) {  
    const audioPlayer = document.getElementById('dj-preview');  
  
    // Extraction audio via YouTube API  
    const audioUrl = await extractAudioUrl(youtubeUrl);  
  
    audioPlayer.src = audioUrl;  
    audioPlayer.currentTime = 30; // Start à 30sec  
    audioPlayer.play();  
  
    // Auto-stop après 30sec preview  
    setTimeout(() => {  
        audioPlayer.pause();  
    }, 30000);  
}
```

## Validation/Refus

javascript

```
async function validateFastPass(requestId) {
  try {
    // 1. Capture paiement Stripe
    const request = await db.fastpass.findById(requestId);
    await stripe.paymentIntents.capture(request.payment_intent_id);

    // 2. Start timer 30min
    const validated = await startDJTimer(requestId);

    // 3. Notification client
    await sendClientNotification(request.client_id, {
      type: 'fast_pass_accepted',
      title: '🎉 FAST PASS ACCEPTÉ !',
      message: 'Votre morceau sera joué dans 30min max',
      track: request.title
    });

    return validated;
  } catch (error) {
    await rejectFastPass(requestId, error.message);
  }
}

async function rejectFastPass(requestId, reason = 'Morceau inapproprié') {
  const request = await db.fastpass.findById(requestId);

  // 1. Annulation paiement
  await stripe.paymentIntents.cancel(request.payment_intent_id);

  // 2. Update statut
  await db.fastpass.update(requestId, {
    status: 'rejected',
    rejection_reason: reason
  });
}
```

```
// 3. Notification client
await sendClientNotification(request.client_id, {
    type: 'fast_pass_rejected',
    title: '✖ FAST PASS REFUSÉ',
    message: 'Remboursement automatique effectué',
    reason: reason
});
}
```

## NOTIFICATIONS TEMPS RÉEL

### Notifications Client

#### Après Paiement

javascript

```
const notification = {
    type: 'fast_pass_sent',
    title: '✓ DEMANDE ENVOYÉE',
    message: 'Votre Fast Pass a été envoyé au DJ.',
    subtitle: 'Patientez pour la décision du DJ...',
    data: {
        montant: '15€',
        track: 'Blinding Lights - The Weeknd',
        timestamp: Date.now()
    }
};
```

#### Acceptation DJ

javascript

```
const acceptedNotification = {
  type: 'fast_pass_accepted',
  title: '⚠ FAST PASS ACCEPTÉ !',
  message: 'Votre morceau sera joué dans 30min max',
  data: {
    track: 'Blinding Lights - The Weeknd',
    estimated_play_time: '23:45',
    amount_charged: '15€'
  }
};
```

## Morceau Joué

javascript

```
const playingNotification = {
  type: 'fast_pass_playing',
  title: '🎵 VOTRE MORCEAU JOUE !',
  message: 'Merci pour votre Fast Pass',
  data: {
    track: 'Blinding Lights - The Weeknd',
    play_time: Date.now()
  }
};
```

## ⚠ Alertes Système

### DJ - Timer Critique

javascript

```
function checkCriticalTimers() {
  const criticalRequests = validatedRequests.filter(r => {
    const remaining = r.deadline - Date.now();
    return remaining <= 5 * 60 * 1000; // 5min restantes
  });

  if (criticalRequests.length > 0) {
    showDJAlert({
      type: 'critical_timer',
      title: '⚠️ TIMERS CRITIQUES',
      message: `${criticalRequests.length} Fast Pass expirent bientôt !`,
      urgent: true
    });
  }
}
```

## 🛠️ IMPLÉMENTATION TECHNIQUE

### Base de Données

sql

-- Table Fast Pass requests

```
CREATE TABLE fast_pass_requests (
    id SERIAL PRIMARY KEY,
    client_id INTEGER REFERENCES clients(id),
    youtube_url TEXT NOT NULL,
    track_title VARCHAR(200),
    track_artist VARCHAR(100),
    montant DECIMAL(10,2) NOT NULL,
    prix_base DECIMAL(10,2) NOT NULL,
    is_surechere BOOLEAN DEFAULT FALSE,
    payment_intent_id VARCHAR(100),
    status VARCHAR(20) DEFAULT 'pending', -- pending, validated, played, rejected, expired
    created_at TIMESTAMP DEFAULT NOW(),
    validated_at TIMESTAMP,
    played_at TIMESTAMP,
    deadline TIMESTAMP, -- Date limite pour jouer (validated_at + 30min)
    rejection_reason TEXT
);
```

-- Index pour performance

```
CREATE INDEX idx_fast_pass_status ON fast_pass_requests(status);
CREATE INDEX idx_fast_pass_deadline ON fast_pass_requests(deadline);
CREATE INDEX idx_fast_pass_client ON fast_pass_requests(client_id);
```

## ⚡ Cache & Performance

javascript

```
// Cache Redis pour tarification temps réel
const cacheKey = `fast_pass_pricing:${clubId}:${hour}`;
await redis.setex(cacheKey, 300, JSON.stringify({
    prix_base: prixBase,
    nb_demandes: nbDemandes,
    prix_final: prixFinal
}));
```

  

```
// Cache YouTube metadata (24h)
const ytCacheKey = `youtube:${videoid}`;
await redis.setex(ytCacheKey, 86400, JSON.stringify(metadata));
```

## Sécurité & Limitations

### Rate Limiting

javascript

```
const rateLimits = {
    fast_pass_requests: {
        max: 3, // 3 Fast Pass max par soirée
        window: 24 * 60 * 60 * 1000, // 24h
        reset_at: '06:00' // Reset à 6h du matin
    },
    url_submissions: {
        max: 10, // 10 URLs max par heure (anti-spam)
        window: 60 * 60 * 1000
    }
};
```

### Validation Contenu

javascript

```

async function validateContent(youtubeUrl) {
    const metadata = await getYouTubeMetadata(youtubeUrl);

    // Blacklist mots-clés inappropriés
    const blacklist = ['explicit', 'nsfw', 'nazi', 'terrorist'];
    const title = metadata.title.toLowerCase();

    if (blacklist.some(word => title.includes(word))) {
        throw new Error('Contenu inapproprié détecté');
    }

    // Durée maximum (10 minutes)
    if (metadata.duration > 600) {
        throw new Error('Morceau trop long (max 10min)');
    }

    return true;
}

```

## MÉTRIQUES & ANALYTICS

### KPIs Business Réajustés

- **Revenus Fast Pass** : Total € générés par période
- **Prix moyen** : Évolution tarifs payés vs prix base (12€)
- **Taux surenchère** : % clients qui paient plus que 12€
- **Montant surenchère moyen** : Prix moyen des enchères (attendu: 25-40€)
- **Taux acceptation DJ** : % Fast Pass validés vs refusés
- **Respect délais** : % morceaux joués dans les 30min

### Projections Revenus

- **Prix de base** : 12€ fixe
  - **Enchères moyennes** : 25-50€ selon compétition
  - **Enchères maximales** : 100-500€ (sans limite)
  - **Volume estimé** : 20-30 Fast Pass/soirée
  - **CA attendu** : 600-1200€/soirée selon dynamique enchères
  - **Commission Ambiance+** : 60-120€/soirée par club
- 

## OBJECTIFS BUSINESS

### Revenus Attendus Réalistes

- **Prix moyen Fast Pass** : 35€ (12€ base + surenchères fréquentes)
- **Volume** : 25 Fast Pass par soirée (club 1000 personnes)
- **CA Fast Pass** :  $875\text{€}/\text{soirée} \times 4 \text{ soirées} = 3500\text{€}/\text{semaine}$
- **Commission Ambiance+** :  $10\% = 350\text{€}/\text{semaine}$  par club
- **Revenus mensuels** :  $\sim 1400\text{€}/\text{mois}$  par club Fast Pass uniquement

## Expérience Musicale

- **Engagement DJ** : Contrôle qualité + respect timeline
- **Satisfaction client** : Garantie 30min + remboursement auto
- **Dynamique soirée** : Morceaux populaires payés = ambiance++

## Effet Viral

- **Reconnaissance** : "Ce morceau choisi par TABLE 12"
  - **Compétition** : Tables qui se battent pour leurs morceaux
  - **Social media** : Stories avec "J'ai choisi cette musique"
-

*Ce récapitulatif couvre l'intégralité de la page Fast Pass : interface, logique métier, système temporel, paiements, interface DJ et implémentation technique complète.*