

Voici une **liste structurée des améliorations et modifications** à apporter au projet **21 BYTS**, avec une **priorisation par ordre logique et efficacité** :



Ordre des modifications pour une efficacité maximale

Étape 1 – Renforcer la fiabilité de l'application (Backend / Core)

1. Ajouter une couverture de tests unitaires et d'intégration

-  Créer des tests pour :
 - `download-manager.js`
 - `auth-manager.js`
 - `metadata-manager.js`
 - `format-converter.js`
-  Utiliser les mocks depuis `/tests/mocks/` pour simuler les API tierces.
-  **But** : Garantir la stabilité avant d'apporter des changements fonctionnels.

2. Améliorer la gestion des erreurs côté système

-  Enrichir `error-handler.js` pour :
 - Journaliser chaque erreur avec `event-logger.js`
 - Réémettre une erreur user-friendly vers `error-dialog.js`
 -  Émettre un événement `ERROR_OCCURRED` standardisé.
-

Étape 2 – Optimiser la communication et architecture modulaire

3. Factoriser les flux d'authentification OAuth

- 📁 Créer un module partagé `oauth-handler.js` dans `/modules/auth/`
- 🎯 Réutilisé par `spotify-adapter`, `tidal-adapter`, etc.
- ✅ **But** : Éviter duplication de logique + simplifier maintenance.

4. Isoler les appels aux outils système

- 📁 Créer `/modules/system/` :
 - `ffmpeg-wrapper.js`
 - `yt-dlp-wrapper.js`
- 📡 Chaque wrapper doit écouter un événement (`CONVERT_REQUEST`, `DOWNLOAD_REQUEST`) et émettre (`CONVERT_SUCCESS`, `DOWNLOAD_FAILED...`).

✅ Étape 3 – Améliorations UX/UI et design responsive

5. Améliorer le feedback visuel dans l'UI

- 🎨 Dans `download-item.js` :
 - Ajouter des états visuels : ⌚ "En cours", ✅ "Terminé", ❌ "Échec"
 - Ajouter une barre dynamique de progression (via événements `DOWNLOAD_PROGRESS`)

6. Ajouter des actions utilisateur

- ⚙️ Boutons :
 - "Pause/Resume"
 - "Retry"
 - "Supprimer de la liste"
- ⚙️ Utiliser le `event-bus.js` pour envoyer des commandes ciblées.

7. Rendre l'interface responsive

- 💡 Modifier les styles dans `main.css` et `themes.css` :
 - Flexbox ou CSS Grid pour adapter les colonnes de téléchargement.
 - Adapter les marges et tailles de police pour mobile/tablette.
 - ✅ Bonus : Ajouter un mode sombre clair/foncé si souhaité.
-

✅ Étape 4 – Améliorations qualité de code et documentation

8. Documenter tous les événements dans `EVENTS.md`

- 📁 Pour chaque module :
 - Ajouter les événements écoutés + émis
 - Inclure les payloads types

Exemple :

```
js
CopierModifier
Event: DOWNLOAD_REQUEST
Payload: { url: string, format: string, provider:
'youtube'|'spotify' }

◦
```

9. Documenter les modules dans `MODULES.md`








- 📖 Inclure :
 - Rôle
 - Points d'entrée (événements)
 - Comportement en cas d'erreur

10. Automatiser le build multiplateforme

- 🔧 Vérifier que les scripts `mac.js`, `windows.js`, `linux.js` sont à jour.

- **+** Intégrer un `build-all.js` global pour simplifier le packaging.

Récapitulatif par priorité

Priorité	Action	Impact
 1	Ajouter tests unitaires	Fiabilité
 2	Améliorer gestion d'erreurs	Robustesse
 3	Factoriser OAuth / wrappers systèmes	Maintenabilité
 4	Ajouter états + feedback visuel UI	UX
 5	Rendre l'UI responsive	Accessibilité
 6	Documenter les événements / modules	Clarté dev
 7	Automatiser le build	Efficacité
