Outil de suivi gamifié de réponse aux marchés publics

V1.2 - 14 avril 2025

Cahier des Charges Technique et Fonctionnel

1. Présentation du projet

1.1 Contexte

Ce projet consiste en un outil web autonome permettant de suivre l'avancement d'une réponse à appel d'offres (AO) de manière ludique et engageante. L'application s'inspire des codes du jeu vidéo rétro de type Game Boy (8-bit) et repose sur une interface interactive et personnalisable. Elle est utilisée dans le cadre des réponses de Tonton Studio aux marchés publics.

1.2 Objectifs

- Offrir un outil de **gestion de tâches et livrables** pour le suivi d'un AO
- Favoriser une approche **gamifiée, motivante et visuellement attractive**
- Permettre une **utilisation autonome, hors ligne**, sans dépendance serveur
- Créer une expérience utilisateur satisfaisante et engageante

1.3 Public cible

- Équipe de Tonton Studio travaillant sur les réponses aux appels d'offres
- Personnes ayant une sensibilité "jeu vidéo rétro"

2. Structure de l'application

2.1 Architecture technique

L'application est construite en HTML, CSS et JavaScript vanilla, sans frameworks externes ni dépendances serveur. Elle fonctionne entièrement côté client avec sauvegarde des données en localStorage.

Structure des fichiers

```
/__CHECKLIST__/
 ├── index.html        # Structure HTML principale
 ├── style.css         # Feuille de style Game Boy
 ├── script.js         # Logique interactive
 ├── tasks.js          # Liste des livrables et sous-tâches
 ├── img/
|   ├── TTS_Logo.png         # Logo standard
|   └── TTS_Logo_Checked.png  # Logo activé (quand tout est complété)
└── audio/
    ├── success.mp3          # Son de complétion d'un livrable
    ├── check.mp3            # Son de cochage d'une tâche
    ├── uncheck.mp3          # Son de décochage d'une tâche
    └── finish.mp3           # Son de finalisation complète
```

2.2 Schéma de données

L'application utilise une structure de données simple :

Format des tâches (tasks.js)

```
const tasks = [
 {
 label: 'Nom du livrable',
 filename: 'Nom_de_fichier_global.pdf', // Fichier global optionnel
 subtasks: [
  { label: 'Sous-tâche 1', filename: 'Fichier_Soustache1.pdf' }, // Sous-tâche avec fichier
  { label: 'Sous-tâche 2', filename: 'Fichier_Soustache2.pdf' }, // Sous-tâche avec fichier
  { label: 'Sous-tâche 3' }, // Sous-tâche sans fichier
  ...
 ]
 },
 ...
];
```

Format de stockage (localStorage)

```
// Clé : tontonAoProgress
// Valeur (exemple) :
{
  "0": [0, 1, 3],  // Indices des sous-tâches complétées pour le livrable 0
  "1": [0, 1, 2],  // Indices des sous-tâches complétées pour le livrable 1
  ...
}
```

3. Fonctionnalités implémentées

3.1 Gestion des livrables

- Affichage de chaque **livrable** sous forme de **bloc (accordion)**
- Chaque livrable contient une **liste de sous-tâches** à réaliser
- Chaque sous-tâche peut être **marquée comme faite/non faite** via case à cocher
- Les sous-tâches peuvent être cochées/décochées en cliquant n'importe où sur la ligne (pas uniquement sur la checkbox)
- L'état des tâches est **sauvegardé automatiquement** en localStorage
- Possibilité de **cocher/décocher toutes les sous-tâches d'un livrable en un seul clic** via une checkbox principale
- Visualisation intuitive avec **symboles de coche (✓)** pour les tâches complétées
- **Mise en évidence visuelle** des tâches cochées avec une couleur de fond différente
- Certaines sous-tâches peuvent être associées à un fichier livrable spécifique
- Le nom du fichier associé à une sous-tâche s'affiche sous le libellé de la sous-tâche
- Possibilité de copier le nom du fichier directement depuis une sous-tâche via un bouton dédié

3.2 Comportement dynamique des blocs

- Les accordéons des livrables **non complétés sont ouverts par défaut**

- Lorsqu'**une tâche est cochée**, l'état est mis à jour et un son de "check" est joué
- Lorsqu'**une tâche est décochée**, l'état est mis à jour et un son de "uncheck" est joué
- Lorsqu'**un livrable est entièrement complété** :
    - Le bloc se replie automatiquement
    - Une **animation de particules / effet visuel** se joue
    - Un **flash blanc** apparaît brièvement sur tout l'écran
    - Une **bande-son** de type "victoire" est jouée
    - Le **nom du fichier attendu s'affiche** avec un bouton **"Copier"**
    - Si le livrable contient des sous-tâches avec des fichiers individuels, un **message informatif** "Les fichiers individuels sont accessibles depuis chaque sous-tâche" s'affiche
    - Si le livrable ne contient pas de sous-tâches avec des fichiers, le nom du fichier global s'affiche avec un bouton "**Copier**"
    - Un badge "DONE!" apparait sur le côté gauche du bloc

3.3 Gestion du nom de fichier

- Chaque livrable possède un **nom de fichier préconisé** (format AO)
- Deux types de noms de fichiers sont gérés :
    - Noms de fichiers associés aux **sous-tâches** (fichiers individuels)
    - Noms de fichiers associés aux **livrables complets** (fichiers globaux)
- Les noms de fichiers des sous-tâches sont affichés directement sous le libellé de la sous-tâche
- Des **boutons "Copier" permettent de copier chaque nom de fichier** dans le presse-papier
- Animation visuelle et feedback "Copié !" confirmant la copie
- Les boutons de copie s'adaptent automatiquement aux différentes tailles d'écran

3.4 Barre de progression

- Une barre de progression globale affiche le pourcentage total de complétion
- La progression est mise à jour en temps réel

3.5 Mode "Victoire" (mode doré)

Lorsque toutes les tâches sont complétées (100%) :

- L'écran **défile automatiquement vers le haut avant** l'activation du mode doré
- Le logo change pour sa version "Checked"
- Toute l'interface passe en thème doré
- Un flash lumineux apparaît
- Un son de victoire finale est joué
- Des feux d'artifice continus apparaissent à l'écran
- Le message "T'as plus qu'à envoyer ton dossier ! Bonne chance !!!" s'affiche
- Le compteur "100%" pulse et grossit
- Si une tâche est décochée, tout revient à l'état normal

3.6 Réinitialisation

- Bouton "Reset Progress" permettant de remettre à zéro toute la progression
- Confirmation demandée avant réinitialisation

4. Direction artistique

4.1 Style visuel

L'application utilise un style visuel inspiré des jeux Game Boy classiques :

Palette de couleurs

- **Mode standard (Game Boy)**
  - Vert foncé (#0f380f) - Bordures, texte foncé
  - Vert moyen (#306230) - En-têtes, pieds de blocs
  - Vert clair (#8bac0f) - Éléments actifs, highlights
  - Vert très clair (#9bbc0f) - Fond des blocs
  - Beige/vert pâle (#c4cfa1) - Fond général
- **Mode doré** (activé à 100%)
  - Or clair (#ffd700) - Accents, highlights
  - Or foncé (#b8860b) - Bordures, ombres
  - Or moyen (#daa520) - En-têtes des blocs
  - Beige doré (#f5e7a3) - Fond général
  - Beige plus clair (#f0d875) - Fond des blocs
  - Marron (#8b4513) - Texte

Police de caractères

- Police pixel "Press Start 2P" de Google Fonts
- Tailles variées selon l'importance des éléments

Éléments visuels

- Bordures pixelisées
- Boutons avec effet d'enfoncement
- Cases à cocher personnalisées style rétro
- Badge "DONE!" animé pour les tâches complétées
- Séparateurs en pointillés entre les sous-tâches

4.2 Animations & effets

Animations d'interface

- Transitions de dépliage/repliage des accordéons
- Effet de survol sur les lignes de sous-tâches
- Animation des boutons (survol, clic)

- Pulse du badge "DONE!"
- Pulse du compteur 100% en mode doré

Effets spéciaux

- **Particules** :
    - Apparaissent lors de la complétion d'un livrable
    - Variées en couleurs et formes (carrés, étoiles)
    - Trajectoires avec légère gravité
    - Différentes tailles et vitesses
- **Feux d'artifice** :
    - Apparaissent en continu lorsque tout est complété
    - Explosions avec éclats dans toutes les directions
    - Couleurs variées (or, orange, rouge)
    - Traînées lumineuses
- **Flash écran** :
    - Flash blanc bref sur tout l'écran
    - Apparaît lors de la complétion d'un livrable
    - Intensité modérée pour ne pas éblouir

Effets sonores

- Son de check : joué en cochant une tâche
- Son de uncheck : joué en décochant une tâche
- Son de succès : joué à la complétion d'un livrable
- Son de victoire finale : joué lorsque tout est complété

4.3 Responsive

- Optimisé pour desktop en mode paysage
- Support amélioré pour tablettes
- Adaptation complète pour mobile avec :
    - Positionnement optimisé des badges "DONE!"
    - Taille de texte adaptative
    - Dimensions cohérentes des checkboxes
    - Boutons "Copier" à taille fixe pour éviter les sauts de mise en page

5. Guide technique détaillé

5.1 Structure HTML

Le fichier index.html contient la structure de base de l'application avec :

- L'en-tête avec logo et titre
- Un message de complétion (masqué par défaut)
- La barre de progression globale
- Le conteneur principal pour les tâches (rempli dynamiquement via JavaScript)

- Le pied de page avec bouton de réinitialisation
- Les éléments pour les effets (conteneurs de particules, flash overlay)
- Les balises audio pour les effets sonores

5.2 CSS détaillé

Organisation du CSS

Le fichier style.css est organisé en sections :

1. Variables CSS (palette de couleurs, mode standard et doré)
2. Styles du conteneur principal et en-tête
3. Styles des accordéons et sous-tâches
4. Styles de la barre de progression
5. Styles du footer et boutons
6. Animations (particules, flash, transitions)
7. Media queries pour responsive
8. **Optimisation mobile** avec media queries spécifiques
9. **Flexbox** pour une mise en page plus flexible sur différentes tailles d'écran
10. **Adaptation des contrastes** pour une meilleure lisibilité sur tous les appareils

Points clés

- **Utilisation de variables CSS** pour faciliter les changements de thème
- **Animations fluides** avec transitions et keyframes
- **Design d'interface cohérent** avec la thématique Game Boy
- **Mode doré** avec surcharge complète du thème via classes CSS
- **Styles adaptés** pour l'affichage des noms de fichiers des sous-tâches
- **Gestion des boutons de copie** pour les fichiers individuels et les fichiers globaux
- **Animation de feedback** lors de la copie d'un nom de fichier

5.3 JavaScript détaillé

Organisation du code

Le fichier script.js contient plusieurs fonctions clés :

1. **Initialisation**
    - initTasks() : Génère dynamiquement les accordéons à partir de tasks.js
    - Chargement de la progression sauvegardée
2. **Gestion des événements**
    - Gestion des clics sur les en-têtes d'accordéon
    - Gestion des changements d'état des cases à cocher
    - Gestion des clics sur les lignes de sous-tâches
    - Gestion des clics sur les boutons de copie
    - Gestion du bouton de réinitialisation

    - Gestion des clics sur la checkbox principale pour cocher/décocher toutes les sous-tâches
3. **Gestion de la progression**
    - updateTaskProgress() : Met à jour la progression d'un livrable
    - updateOverallProgress() : Met à jour la progression globale
4. **Effets visuels et sonores**
    - createParticleEffect() : Génère l'effet de particules
    - createFirework() : Génère les feux d'artifice
    - createFlash() : Crée le flash lumineux
    - playSuccessSound(), playCheckSound(), etc. : Gèrent les sons
5. **Gestion des modes**
    - activateGoldenMode() : Active le mode doré
    - deactivateGoldenMode() : Désactive le mode doré
    - startContinuousFireworks() : Lance les feux d'artifice continus
    - stopContinuousFireworks() : Arrête les feux d'artifice

Points clés

- **Modularité** des fonctions pour faciliter la maintenance
- **Sauvegarde automatique** après chaque interaction
- **Gestion efficace des événements** avec délégation
- **Animations optimisées** avec requestAnimationFrame
- **Gestion d'état** claire et cohérente

5.4 Performances et optimisation

- Utilisation de requestAnimationFrame pour les animations fluides
- Limitation du nombre de particules pour éviter les surcharges
- Nettoyage des éléments d'animation après utilisation
- Préchargement des sons pour éviter les délais

6. Guide de maintenance et d'extension

6.1 Modification des tâches

Pour modifier la liste des livrables et des sous-tâches, il suffit d'éditer le fichier tasks.js.

Exemple d'ajout d'un livrable :

```
// Dans tasks.js
const tasks = [
 // Livrables existants...

 // Nouveau livrable à ajouter :
 {
 label: 'Nouveau livrable',
 filename: 'Nouveau_Fichier.pdf',
 subtasks: [
  { label: 'Première étape', filename: 'Etape1_Fichier.pdf' },
```

```
  { label: 'Deuxième étape' }, // Sans fichier associé
  { label: 'Troisième étape', filename: 'Etape3_Fichier.pdf' }
 ]
 }
];
```

6.2 Personnalisation visuelle

Pour modifier l'apparence de l'application :

1. **Changer la palette de couleurs** : Modifier les variables CSS dans :root {} au début du fichier style.css
2. **Remplacer le logo** : Mettre à jour les fichiers dans /img/
3. **Modifier les sons** : Remplacer les fichiers dans /audio/

6.3 Extensions possibles

L'application pourrait être étendue avec les fonctionnalités suivantes :

1. **Ajout de dates limites** pour les livrables
2. **Mode multi-utilisateur** avec synchronisation cloud
3. **Export PDF** de l'état d'avancement
4. **Ajout de notes** ou commentaires sur les tâches
5. **Filtres** pour voir uniquement les tâches à faire
6. **Personnalisation des thèmes** par l'utilisateur
7. **Notifications** pour rappeler les tâches en retard

7. Tests et debugging

7.1 Compatibilité navigateur

L'application est compatible avec les navigateurs modernes :

- Chrome (recommandé)
- Firefox
- Safari
- Edge

Spécificités sur mobile :

- Optimisation spécifique pour les navigateurs mobiles (Chrome Android, Safari iOS)
- Interface adaptative avec mise en page et tailles de texte optimisées pour les petits écrans

7.2 Problèmes connus et solutions

1. **Son ne joue pas** : Certains navigateurs bloquent la lecture automatique. Solution : ajouter une interaction utilisateur avant de jouer le son.

2. **LocalStorage non disponible** : En mode navigation privée, localStorage peut être désactivé. Solution : vérifier sa disponibilité et proposer une alternative.

7.3 Debugging

Pour faciliter le debugging :

- Consulter la console du navigateur (F12)
- Vérifier le localStorage via l'onglet "Application" des outils de développement
- Tester les fonctionnalités une par une en cas d'erreur

8. Conclusion

L'outil de suivi gamifié pour les appels d'offres est une application web autonome et ludique qui facilite le suivi des tâches à accomplir pour répondre à un AO. Avec son esthétique inspirée de la Game Boy et ses nombreux effets visuels et sonores, elle transforme une activité potentiellement fastidieuse en expérience engageante.

Sa structure modulaire et son absence de dépendances externes en font un outil facile à maintenir et à faire évoluer. L'utilisation du localStorage garantit la persistance des données sans nécessiter de connexion à un serveur, permettant une utilisation en toute autonomie.

L'application représente parfaitement la philosophie de Tonton Studio, alliant efficacité professionnelle et approche créative inspirée de la culture du jeu vidéo.