Outil de suivi gamifié de réponse aux marchés publics

v2.3 - 23 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

• Toute personne 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.

L'architecture a été refactorisée en version 2.0 pour adopter une approche modulaire, avec une séparation claire des responsabilités entre les différents composants :

• Fichiers CSS modulaires pour chaque composant de l'interface
• Modules JavaScript spécialisés pour chaque fonctionnalité
• Système audio centralisé pour une gestion robuste des effets sonores
• Gestionnaire d'événements optimisé pour les interactions utilisateur

2.2. Structure des fichiers

L'application suit une architecture modulaire avec séparation claire des responsabilités :

```
/__CHECKLIST__/
 ├── index.html           # Page principale
 ├── css/                 # Styles modulaires
|   ├── base.css         # Styles de base
|   ├── tasks.css        # Styles des tâches
|   ├── buttons.css      # Styles des boutons
|   ├── progress.css     # Styles de la barre de progression
|   ├── donate-block.css  # Styles du bloc de donation
```

```
|   ├── effects.css        # Styles des effets visuels
|   ├── modes.css          # Styles des modes (doré, warning)
|   ├── tasks-fix.css      # Correctifs pour les titres longs
|   ├── warning-override.css # Surcharge pour le mode warning
|   ├── responsive.css     # Styles adaptatifs
|   ├── welcome.css        # Styles de la page d'accueil
|   └── countdown.css      # Styles du compte à rebours
├── js/
|   ├── app.js            # Initialisation de l'application
|   ├── config.js         # Configuration globale
|   ├── utils.js          # Fonctions utilitaires
|   ├── sound-control.js   # Gestion du son
|   ├── countdown.js       # Gestion du compte à rebours
|   ├── effects.js         # Effets visuels et sonores
|   ├── tasks-manager.js   # Gestion des tâches
|   ├── fixed-progress-bar.js # Gestion de la barre de progression fixe
|   ├── sound-manager.js   # Gestion avancée du son
|   └── audio-engine.js    # Moteur audio
├── audio/
|   ├── check.mp3          # Son de cochage
|   ├── uncheck.mp3        # Son de décochage
|   ├── success.mp3        # Son de complétion
|   ├── finish.mp3         # Son de finalisation
|   └── warning.mp3        # Son d'avertissement
└── img/
    ├── TTS_Logo.png       # Logo standard
    ├── TTS_Logo_Checked.png # Logo activé (quand tout est complété)
    ├── tonton.png         # Avatar Tonton Studio
    └── mail.png           # Icône de mail en pixel art
```

2.3. Schéma de données

L'application utilise une structure de données simple :

2.3.1. Format des tâches (tasks.js)

```
// Configuration globale de l'appel d'offres
const aoConfig = {
  title: "Titre de l'appel d'offres",
  deadline: "YYYY-MM-DDThh:mm:ss", // Format ISO pour la date limite
  reference: "REF-YYYY-X" // Référence de l'AO
};

// Structure des tâches
const tasks = [
  {
    label: 'Nom du livrable',
    filename: 'Nom_de_fichier_global.pdf', // Fichier global optionnel
    isMultiFile: false, // Indique si le livrable est un fichier unique ou multiple
    subtasks: [
      { label: "Sous-tâche 1", filename: "Fichier_Soustache1.pdf" }, // Avec fichier
      { label: "Sous-tâche 2" } // Sans fichier
    ]
  },
  {
```

```
   label: 'Livrable multi-fichiers',
   isMultiFile: true, // Pour les livrables comportant plusieurs fichiers distincts
   subtasks: [
    { label: "Document 1", filename: "Document1.pdf" },
    { label: "Document 2", filename: "Document2.pdf" }
   ]
  }
];
```

2.3.2. 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
- L'état des tâches est sauvegardé automatiquement en localStorage
- Possibilité de cocher/décocher toutes les sous-tâches d'un livrable en un seul clic
- 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
- Gestion optimisée des titres longs avec troncature élégante et points de suspension

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

3.2.1. 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 s'affiche
• Si le livrable ne contient pas de sous-tâches avec des fichiers, le nom du fichier global s'affiche
• Un badge "DONE!" animé apparait sur le côté gauche du bloc, visible même lors du défilement

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
• La barre de progression reste visible en haut de l'écran lors du défilement
• Implémentation robuste utilisant requestAnimationFrame pour de meilleures performances
• Adaptation automatique de la largeur lors du redimensionnement de la fenêtre
• En mode doré, la barre de progression fixe adopte automatiquement les couleurs correspondantes
• En mode warning, la barre de progression adopte la palette rouge caractéristique
• Les badges "DONE!" restent visibles même lorsque la barre de progression est fixée

3.5. Compte à rebours

• Affichage du temps restant jusqu'à la date limite de l'AO
• Format jours/heures/minutes/secondes avec mise à jour en temps réel
• Activation automatique du mode warning lorsqu'il reste moins de 48h
• Messages contextuels selon le temps restant
• Adaptation visuelle en fonction du mode (normal, warning, doré)

3.6. Mode warning

• Activation automatique lorsqu'il reste moins de 48h avant la date limite
• Refonte complète avec palette de couleurs optimisée (rouge, blanc, noir)
• Fond noir en mode warning pour un meilleur contraste visuel
• Alerte sonore avec possibilité de couper le son via un bouton dédié
• Fade-out automatique du son d'alerte lorsque toutes les tâches sont complétées
• Élimination totale des éléments verts résiduels pour une cohérence visuelle parfaite
• Surcharge CSS spécifique pour garantir l'application complète du mode

3.7. 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

3.8. Interface d'accueil

- Page d'accueil redessinée avec style épuré et plus lisible
- Chargement automatique des fichiers tasks.js dès leur sélection
- Mise en avant du GPT spécialisé pour la génération des fichiers tasks.js
- Bouton direct vers le GPT avec icône distinctive
- Bloc de donation redesigné avec médaillon et meilleure mise en valeur
- Logo cliquable donnant accès direct au site Tonton Studio

3.9. Réinitialisation et navigation

- Bouton "Reset Progress" permettant de remettre à zéro toute la progression
- Confirmation demandée avant réinitialisation
- Bouton "Retour Accueil" clairement différencié du bouton de réinitialisation
- Arrêt automatique du son d'alerte lors du retour à l'accueil

3.10. Intégration du GPT spécialisé

- Intégration d'un GPT dédié pour la génération automatique des fichiers tasks.js
- Lien direct vers le GPT depuis l'interface d'accueil
- Le GPT analyse les documents d'appel d'offres (RC, CCTP, etc.)
- Génération automatique d'un fichier tasks.js parfaitement formaté
- Identification intelligente des livrables et sous-tâches
- Distinction automatique entre documents à fichier unique et multi-fichiers
- Accessible à l'adresse : https://chatgpt.com/g/g-680541e3745c8191b7bca4aa6861ad09

4. Direction artistique

4.1. Style visuel

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

4.2. Modes d'interface

• Mode normal : L'interface de base, inspirée de la Game Boy avec palette verte
• Mode warning : Activé automatiquement lorsqu'il reste moins de 48h, utilise une palette rouge
• Mode doré : Activé lorsque toutes les tâches sont complétées, avec effets spéciaux

4.3. 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 warning (palette rouge)
• Rouge foncé (#380f0f) - Bordures, texte foncé
• Rouge moyen (#622020) - En-têtes, pieds de blocs
• Rouge vif (#ac0f0f) - Éléments actifs, highlights
• Rouge clair (#bc0f0f) - Fond des blocs
• Beige/rouge pâle (#cfa1a1) - Fond général
• Noir (#000000) - Fond hors cadre pour meilleur contraste

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

4.4. Police de caractères

• Police pixel "Press Start 2P" de Google Fonts
• Tailles variées selon l'importance des éléments
• Gestion optimisée des titres longs avec troncature et ellipsis

4.5. 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

• Bloc de donation redesigné avec médaillon
• Bouton de contrôle du son avec icône explicite

4.6. Animations & effets

4.6.1. 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
• Animation de feedback "Copié !" pour les boutons de copie
• Transition fluide lors de la fixation de la barre de progression

4.6.2. 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
• Optimisées avec requestAnimationFrame pour de meilleures performances

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

4.6.3. 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
• Son d'alerte warning : joué en boucle lorsqu'il reste moins de 48h
• Fade-out automatique du son d'alerte lorsque toutes les tâches sont complétées
• Contrôle du son via un bouton dédié (activation/désactivation)

4.7. 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
  - Meilleure lisibilité des titres longs
  - Optimisation de l'espace pour les petits écrans

5. Guide technique détaillé

5.1. Structure HTML

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

• Deux écrans principaux :
  - Écran d'accueil avec sélection de fichier et informations
  - Écran de checklist avec la liste des tâches

• Éléments de l'écran d'accueil :
  - En-tête avec logo et titre
  - Conteneur de téléchargement de fichier
  - Bouton vers le GPT spécialisé
  - Section d'information et bloc de donation

• Éléments de l'écran de checklist :
  - En-tête avec logo et titre
  - Titre du marché et référence
  - Compte à rebours
  - Message de complétion (masqué par défaut)
  - Barre de progression globale
  - Conteneur principal pour les tâches (rempli dynamiquement via JavaScript)
  - Pied de page avec boutons de réinitialisation et retour
  - Bloc de donation

• Éléments pour les effets :
  - Conteneurs de particules et feux d'artifice
  - Flash overlay
  - Balises audio pour les effets sonores
  - Bouton de contrôle du son

5.2. CSS détaillé

5.2.1. Organisation du CSS

L'architecture CSS a été entièrement refactorisée en version 2.0 pour adopter une approche modulaire :

- base.css : Styles fondamentaux, variables CSS et reset
- tasks.css : Styles des accordéons, sous-tâches et checkboxes
- buttons.css : Styles de tous les boutons de l'interface
- progress.css : Styles de la barre de progression et son comportement fixe
- donate-block.css : Styles du bloc de donation redesigné
- effects.css : Styles des animations, particules et feux d'artifice
- modes.css : Styles des modes doré et warning
- tasks-fix.css : Correctifs pour les titres longs avec troncature
- warning-override.css : Surcharge spécifique pour le mode warning
- responsive.css : Styles adaptatifs pour différentes tailles d'écran
- welcome.css : Styles de la page d'accueil
- countdown.css : Styles du compte à rebours

5.2.2. 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
- Mode warning avec palette rouge et surcharges spécifiques
- 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
- Gestion d'une barre de progression fixe lors du défilement avec adaptation visuelle
- Transitions fluides pour les changements d'état des en-têtes de tâches au survol
- Z-index optimisés pour une hiérarchie claire des éléments
- Gestion des débordements pour les titres longs
- Optimisation mobile avec media queries spécifiques

5.3. JavaScript détaillé

5.3.1. Organisation du code

L'architecture JavaScript a été refactorisée en version 2.0 pour adopter une approche modulaire :

- app.js : Point d'entrée principal et initialisation de l'application
- config.js : Configuration globale et paramètres
- utils.js : Fonctions utilitaires réutilisables

• sound-control.js : Gestion du contrôle du son (activation/désactivation)
• countdown.js : Gestion du compte à rebours et détection du mode warning
• effects.js : Gestion des effets visuels (particules, feux d'artifice, flash)
• tasks-manager.js : Gestion des tâches, cochage/décochage, progression
• fixed-progress-bar.js : Gestion de la barre de progression fixe
• sound-manager.js : Gestion avancée du son avec fade-out
• audio-engine.js : Moteur audio centralisé

5.3.2. Fonctionnalités clés

Initialisation
• Chargement automatique des fichiers tasks.js dès leur sélection
• Génération dynamique des accordéons à partir de tasks.js
• Chargement de la progression sauvegardée
• Initialisation du compte à rebours

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 et retour accueil
• Gestion des clics sur la checkbox principale
• Gestion du défilement pour fixer la barre de progression
• Gestion du bouton de contrôle du son

Gestion de la progression
• Mise à jour de la progression d'un livrable
• Mise à jour de la progression globale
• Sauvegarde automatique dans localStorage
• Activation/désactivation du mode doré

Effets visuels et sonores
• Génération des effets de particules
• Génération des feux d'artifice
• Création du flash lumineux
• Gestion des sons avec contrôle et fade-out

Gestion des modes
• Activation du mode doré à 100%
• Activation du mode warning à moins de 48h
• Gestion des feux d'artifice continus
• Adaptation de la barre de progression fixe selon le mode

5.3.3. 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
• Détection intelligente du mode warning
• Chargement automatique des fichiers tasks.js
• Gestion robuste des erreurs de lecture audio
• Fade-out du son en mode warning lorsque tout est complété
• Arrêt du son lors du retour à l'accueil

5.4. Performances et optimisation

• Utilisation de requestAnimationFrame pour les animations fluides
• Limitation du nombre de particules pour éviter les surcharges
• Système de pool d'objets pour éviter la création/destruction fréquente d'éléments DOM
• Préchargement des sons pour éviter les délais
• Optimisation de la barre de progression fixe avec throttling amélioré
• Réduction des reflows et repaints via transform et opacity
• Gestion efficace des événements avec délégation
• Debouncing des opérations coûteuses comme les sauvegardes
• Mise en cache des éléments DOM fréquemment utilisés
• Optimisation des transitions CSS avec will-change
• Chargement différé des ressources non-critiques
• Préchargement des ressources critiques
• Réduction des logs en production
• Gestion des événements visibilité pour pause automatique des animations
• Optimisation des performances sur mobile

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 ou d'utiliser le GPT spécialisé pour générer un nouveau fichier.

6.1.1. Exemple d'ajout d'un livrable :

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

  // Nouveau livrable à ajouter :
  {
    label: 'Nouveau livrable',
```

```
  filename: 'Nouveau_Fichier.pdf',
  isMultiFile: false,
  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 :

- Changer la palette de couleurs : Modifier les variables CSS dans les fichiers CSS modulaires
- Remplacer le logo : Mettre à jour les fichiers dans /img/
- Modifier les sons : Remplacer les fichiers dans /audio/
- Adapter les styles : Modifier les fichiers CSS correspondants aux composants souhaités

6.3. Génération automatique avec GPT spécialisé

Pour créer rapidement un fichier tasks.js parfaitement formaté :

1. Accéder au GPT spécialisé via le lien dans l'application ou directement à l'adresse : https://chatgpt.com/g/g-680541e3745c8191b7bca4aa6861ad09

2. Télécharger les documents d'appel d'offres (RC, CCTP, etc.)

3. Le GPT analysera automatiquement les documents et générera un fichier tasks.js prêt à l'emploi

4. Télécharger le fichier généré et l'utiliser avec l'application

6.4. Extensions possibles

L'application pourrait être étendue avec les fonctionnalités suivantes :

- Ajout de dates limites pour les livrables individuels
- Mode multi-utilisateur avec synchronisation cloud
- Export PDF de l'état d'avancement
- Ajout de notes ou commentaires sur les tâches
- Filtres pour voir uniquement les tâches à faire
- Personnalisation des thèmes par l'utilisateur
- Notifications pour rappeler les tâches en retard
- Intégration avec des outils de gestion de projet
- Version mobile native (PWA)

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
- Gestion tactile améliorée pour les interactions

7.2. Problèmes connus et solutions

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

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

Problèmes de performance sur mobile : Sur certains appareils anciens, les animations peuvent être saccadées. Solution : réduire le nombre de particules et d'effets visuels.

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
- Utiliser le mode debug configurable dans config.js
- Vérifier les erreurs de chargement des ressources (sons, images)
- Tester sur différents navigateurs et tailles d'écran

8. Déploiement

L'application peut être déployée sur n'importe quel hébergement statique ou serveur web traditionnel :

- GitHub Pages
- Netlify

• Vercel
• Serveur web traditionnel (Apache, Nginx)
• Hébergement mutualisé

Étapes de déploiement :
1. Cloner le dépôt ou télécharger les fichiers
2. Personnaliser le fichier tasks.js selon les besoins
3. Téléverser les fichiers sur l'hébergement choisi
4. Aucune configuration serveur spécifique n'est nécessaire

L'application est une application front-end pure sans dépendances côté serveur, ce qui la rend facile à déployer et à maintenir.

9. 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.

La version 2.3 apporte de nombreuses améliorations significatives :
• Architecture modulaire pour une meilleure maintenabilité
• Interface responsive optimisée pour tous les appareils
• Barre de progression fixe pour un meilleur suivi
• Mode warning amélioré avec palette de couleurs optimisée
• Intégration d'un GPT spécialisé pour la génération des fichiers tasks.js
• Nombreuses optimisations de performance et d'interface

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.