Cahier des charges :
BlindTest

Groupe : 3 personnes
Début : 6 octobre 2025
Livraison finale : 5 janvier 2026 (présentation finale 05/01/2026)
Techniques utilisés : Java, JavaFX, JUnit (tests), JSON, Git (dépôt public), Apple iTunes Search API
Contrainte fondamentale du cours : le cahier des charges rendu le 6 octobre 2025 doit être complet et fidèle à l'application finale

Présentation du projet

Contexte / Client : Projet universitaire (Conception de logiciels). Réaliser un jeu de Blindtest musical multi-modes (solo / duel) avec interface graphique complète, sauvegarde des playlists, scores, options audio et indice.

But : Concevoir et implémenter une application Java/JavaFX qui joue des extraits musicaux récupérés via l'API iTunes, collecte des réponses (titre/artiste), gère le scoring, l'interface et la persistance. Projet réalisé en méthode Agile, livrables intermédiaires aux dates du cours.

Objectifs fonctionnels

Écran d'accueil

Dès le lancement, le joueur arrive sur un écran d'accueil intuitif. Le menu principal propose plusieurs options :

- Nouveau jeu pour commencer une partie
- Duel pour jouer à deux
- Classement pour consulter les meilleurs scores
- Paramètres pour modifier l'expérience de jeu
- Charger une playlist
- Quitter le jeu

Gestion des joueurs

Chaque joueur peut saisir son pseudo avant de jouer. Le jeu affiche ensuite les scores ainsi que l'historique des meilleures performances, sauvegardé d'une session à l'autre afin de suivre la progression dans le temps.

Modes de jeu

Deux modes sont disponibles :

Solo : un joueur affronte le jeu sur un nombre configurable de manches.

Duel : deux joueurs s'affrontent en alternance, un extrait par joueur, avec des scores séparés.

Sélection du genre musical

Avant de démarrer une partie, le joueur peut choisir un genre musical (rock, pop, rap, jazz, etc.). Le jeu utilisera cette sélection pour filtrer les extraits récupérés via l'API iTunes, afin de proposer un blindtest ciblé selon les goûts du joueur.

Lecture des extraits

Les extraits musicaux, d'une durée de 10 à 30 secondes, sont récupérés grâce à l'API iTunes Search. Cette API fournit à la fois les métadonnées (titre, artiste, album) et l'URL permettant la lecture de l'extrait.
Le son est diffusé à l'aide du composant JavaFX Media/MediaPlayer.
Le joueur peut également utiliser une playlist locale (fichier JSON) ou effectuer une recherche dynamique par genre, album, artiste ou mot-clé.

Saisie des réponses

Pendant le jeu, le joueur entre ses propositions dans plusieurs champs texte (titre et artiste).

Système de points

Le score se calcule ainsi :

- +10 points si le titre est correct,
- +5 points si l'artiste est correct,
- Un total de 15 points disponible sans compter les bonus

- Un bonus de rapidité peut être activé dans les paramètres : +2 points si la réponse est donnée avant 10 secondes.

Classement

Les résultats sont enregistrés localement dans un tableau des scores, avec sauvegarde automatique au format JSON.
Les meilleurs scores sont triés et affichés dans un classement clair et lisible.

Paramètres

Le menu des paramètres permet d'ajuster différents éléments du jeu :

- nombre de manches
- durée des extraits
- activation des indices
- activation du bonus de vitesse
- volume sonore par défaut
- un genre musical par défaut

Indices

Pour aider le joueur, il est possible d'activer des indices : par exemple, afficher la première lettre du titre ou de l'artiste.

Options audio

L'utilisateur peut régler le volume via un curseur, activer/désactiver le son (mute/unmute) et mettre la lecture en pause ou en lecture librement pendant un extrait.

Sauvegarde & chargement

Les playlists peuvent être importées au format JSON.
Le jeu assure aussi la sauvegarde automatique des meilleurs scores et des paramètres de configuration.

Tests & Documentation

Des tests unitaires (JUnit) garantissent la fiabilité du code pour les modules principaux : calcul du score, comparaison des réponses, gestion des playlists et client API Apple mocké.
La Javadoc décrit toutes les classes publiques, et un README accompagné d'un manuel d'utilisation complète la documentation.

Livrables

Le projet est structuré ainsi :

- Le code source dans le dossier `src/`
- Le dossier `doc/` contenant le CDC, les diagrammes UML, le Gantt, la Javadoc et le manuel utilisateur
- Un dépôt Git public, dont le lien est indiqué dans le README.

Contraintes techniques

Langage et interface utilisateur

Le projet est entièrement développé en Java, avec une interface construite à l'aide de JavaFX. Cette combinaison permet d'offrir une expérience fluide, portable et agréable, tout en assurant une bonne intégration entre la logique du jeu et l'affichage graphique.

API musicale

Le jeu exploite l'API iTunes Search proposée par Apple, en utilisant exclusivement les endpoints gratuits et en respectant les conditions d'utilisation d'Apple. Cette API permet de récupérer les métadonnées des morceaux (titre, artiste, album) ainsi que les extraits audio utilisés pendant les parties.

Gestion de l'audio

La lecture des extraits se fait grâce au composant JavaFX MediaPlayer, compatible avec les URLs audio renvoyées par l'API iTunes. Cela permet une lecture fluide et immédiate des extraits musicaux, sans nécessiter de stockage local.

Stockage des données

Les scores des joueurs sont sauvegardés au format JSON, dans un dossier local data/ situé à la racine du projet. Ce format léger et lisible facilite à la fois la persistance des données et leur réutilisation lors des sessions suivantes.

Tests et validation

L'application est testée à l'aide de JUnit 5, afin de garantir la fiabilité des principales fonctionnalités (calcul du score, gestion des playlists, communication avec l'API, etc.). Ces tests assurent la stabilité du projet et facilitent les futures évolutions du code.

Gestion de version

Le développement est géré sur GitHub, avec un dépôt public. Chaque membre de l'équipe s'engage à réaliser au moins un commit par semaine, garantissant ainsi un suivi régulier et une collaboration efficace.

Portabilité

Le programme est compatible avec Windows, Linux et macOS, à condition de disposer du JRE et de JavaFX. Une documentation précise explique la procédure d'installation et d'exécution du projet sur chaque système

.

Sécurité et confidentialité

Le jeu ne stocke aucune donnée personnelle sensible. Toutes les interactions respectent les conditions d'utilisation de l'API iTunes, notamment en matière de confidentialité et de droit d'auteur.

Performance

L'application vise une lecture d'extrait en moins de 3 secondes après la sélection, selon la latence du réseau. Un système de cache léger est prévu pour les métadonnées afin d'optimiser la rapidité et la fluidité de l'expérience utilisateur.

Charge de travail

Le rythme de travail est réparti de manière équilibrée, avec un investissement modéré de 3 à 5 heures par semaine et par personne. Cette organisation permet d'assurer une progression régulière et la bonne coordination de l'équipe.

Critères d'acceptation (définis et testables)

L'application démarre et affiche le menu principal

Le mode Solo et Duel fonctionnent
Possibilité de configurer le nombre de manches et la durée d'extrait

Les extraits sont récupérés et lus via iTunes Search API
Les requêtes doivent renvoyer un extrait jouable

Le scoring suit les règles (+10 / +5 / +15)
Les résultats apparaissent correctement à la fin d'un match

Le classement persiste entre les exécutions

Les paramètres sont persistés et rechargés

Import des playlists JSON fonctionnel

Couverture de tests de la logique métier

Documentation fournie : README, manuel d'exécution, diagrammes

Organisation du projet & conventions

Conventions de code (Java)

- Nommage CamelCase pour les classes (ex : GameController)
- Nommage lowerCamelCase pour les méthodes et variables
- Constantes en UPPER_SNAKE_CASE
- Anglais pour noms d'API et variables publiques
- Commentaires en français
- Limite de 120 colonnes par ligne
- Javadoc pour les classes publiques

Branching Git

- main : livrable stable
- dev : intégration continue
- feature/<nom> : branche par tâche

Architecture

Packages proposés:

- app : point d'entrée (MainApp)
- app.ui : vues JavaFX (MainMenuView, GameView, SettingsView, LeaderboardView, PlayerSetupView)
- app.controller : contrôleurs (GameController, UIController)
- app.model : classes métier (Player, Track, Round, Score, Playlist)
- app.service : AppleAPIClient, AudioService, PlaylistService, PersistenceService
- app.util : utilitaires (StringComparator, Config)
- test : tests unitaires

Détails techniques — logique métier

Comparaison titre / artiste

- Comparaison insensible à la casse
- Suppression des accents et des espaces en début/fin
- Implémentée dans StringComparator

Gestion des extraits

- Requête iTunes : https://itunes.apple.com/search?term=<query>&limit=<n>&media=music
- Extraire previewUrl
- Vérifier HTTP 200 et Content-Type supporté par JavaFX
- Si previewUrl absent, prendre un autre résultat
- Fallback prévu : playlist locale
- La recherche des extraits peut être filtrée selon le genre sélectionné par le joueur.

Scoring & Round flow

- Le round démarre, MediaPlayer lance le preview (durée configurable)
- Timer UI affiche le temps restant
- À la fin du temps, réponse verrouillée
- Calcul du score selon les règles (+ bonus/malus éventuels)
- Passage au round suivant

Persistance JSON

- scores.json → tableau d'objets {pseudo, score, date}
- playlists/*.json → {name, tracks:[{trackId, title, artist, previewUrl, duration}]}

Tests

Les tests sont conçus pour garantir la stabilité, la robustesse et la fiabilité du jeu, aussi bien sur la logique métier que sur les interactions utilisateur.

Tests unitaires (JUnit 5)

- AppleAPIClient : simulation de requêtes HTTP

- StringComparator : contrôle de la gestion des accents, des fautes de frappe et des lettres manquantes.

- ScoreManager : test de l'enregistrement, du calcul et du tri des scores.

- PlaylistService : validation du chargement des playlists au format JSON.

- GameController : test de la logique de round et de l'attribution des points.

Tests manuels

- Tests d'intégration : vérification du bon fonctionnement de l'API iTunes et du MediaPlayer (connexion réseau nécessaire).

- Tests UX : contrôle de la navigation, des clics et du comportement des boutons.

Documentation et livrables

La documentation du projet est structurée de manière à faciliter la compréhension, la maintenance et la réutilisation du code :

- README.md : informations sur les membres de l'équipe, lien vers le dépôt GitHub, instructions de build et d'exécution (Gradle), prérequis Java/JavaFX et guide de démarrage rapide.

- doc/cahier_des_charges.pdf : document de référence détaillant les spécifications du projet.

- doc/classes.png : diagrammes de Classes

- doc/gantt.png : diagramme de Gantt

- doc/javadoc/ : documentation technique générée automatiquement.

- doc/tests/ : rapport des tests (succès/échecs).

- src/ : code source complet et commenté.

Gestion des risques et solutions

- API inaccessible ou extrait manquant → utilisation d'une playlist locale de secours et affichage d'un message clair à l'utilisateur.

- Problèmes de compatibilité JavaFX → documentation des versions compatibles et proposition d'un fichier JAR empaqueté incluant JavaFX.

- Déséquilibre dans la charge de travail → mise en place de commits hebdomadaires et revues de pull requests.

- Manque de temps pour l'intégration audio → priorité donnée à la logique métier, avec mock audio en Sprint 1 et audio réel intégré au Sprint 2.

Répartition hebdomadaire et estimation d’effort

La charge de travail estimée est de 3 à 5 heures par semaine et par personne.
Chaque membre doit :

- réaliser au moins un commit par semaine,
- documenter un ticket par sprint,
- et contribuer activement à la revue du code et à la documentation.

12. Annexes

Diagramme de classes :

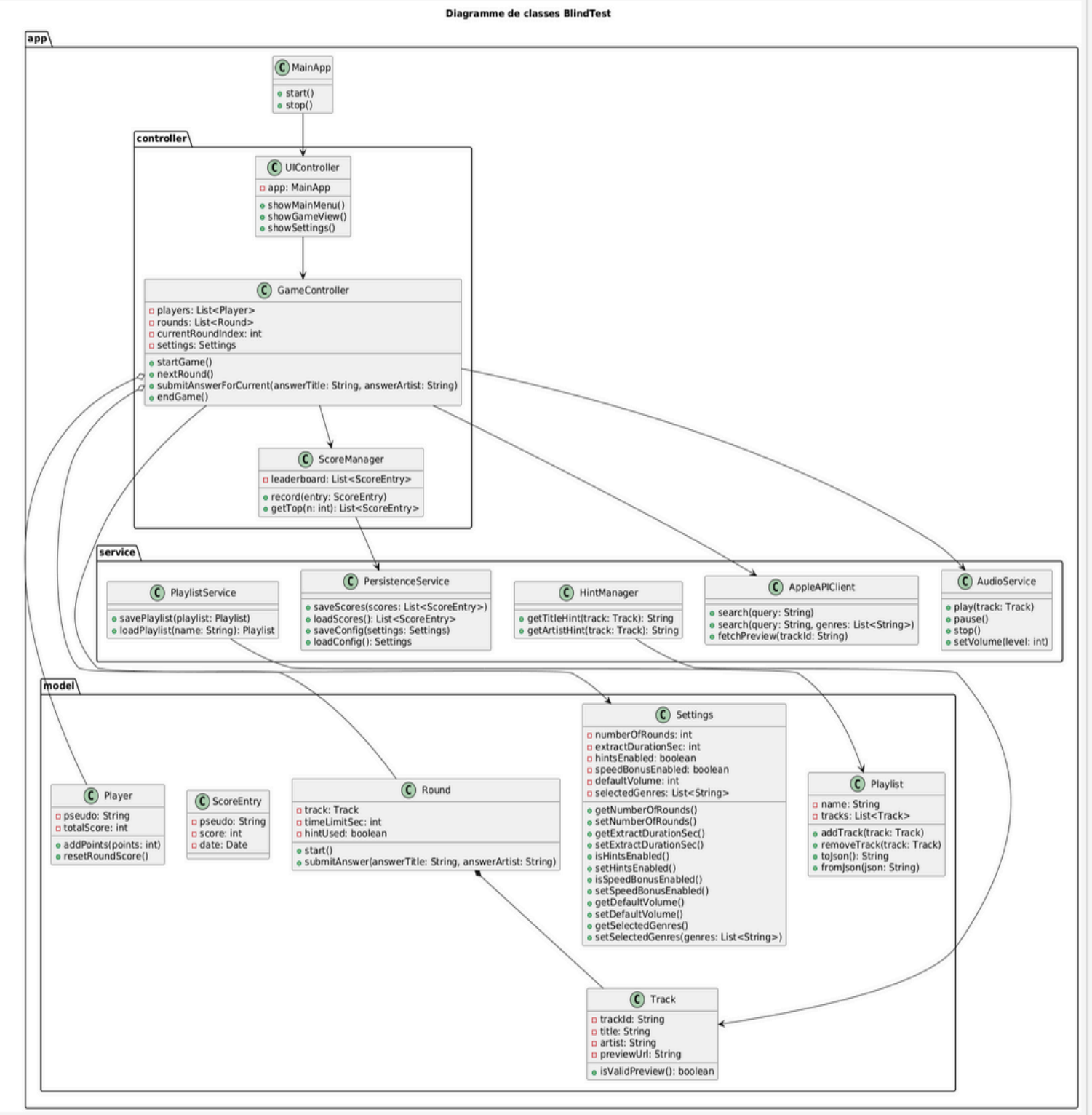
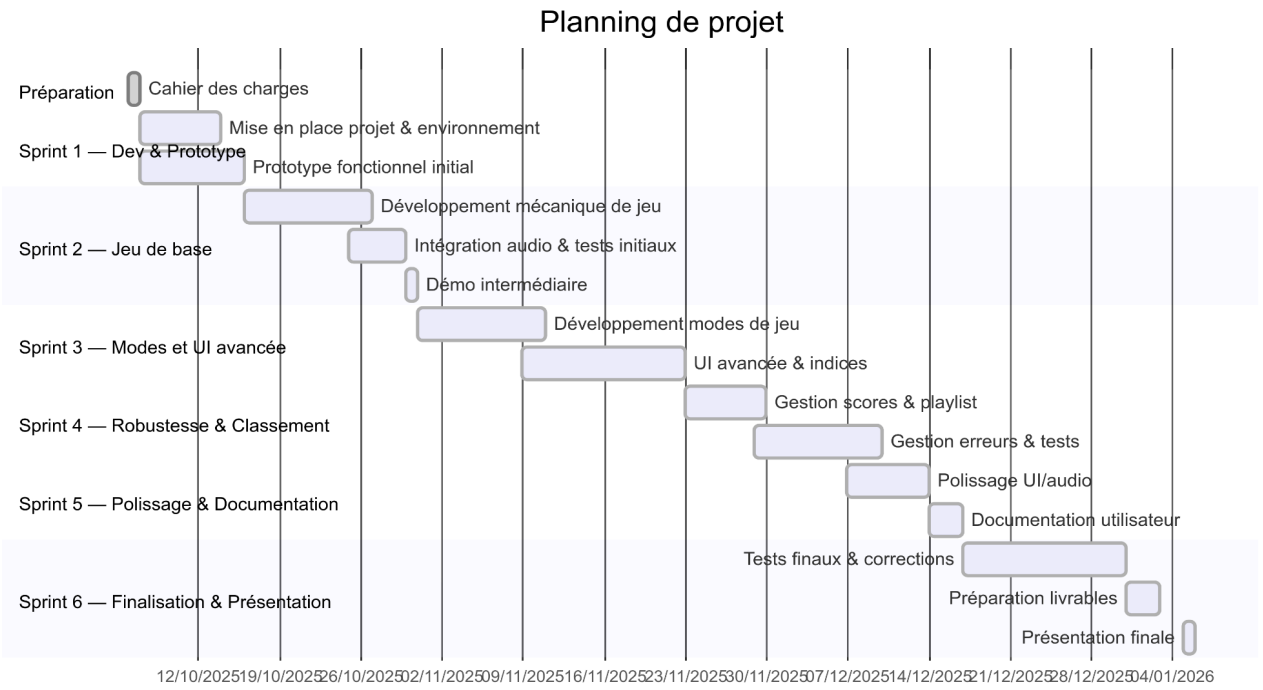


Diagramme de Gantt :



Sprint 1 — Développement & Prototype (06/10/2025 → 16/10/2025)

Objectif général : Mise en place du projet et réalisation d'un prototype fonctionnel initial.

Répartition des tâches pour trois personnes :

1. **Mise en place du projet & environnement**
    - **Iskander :**
        - Créer le dépôt Git local et initialiser le projet Java/JavaFX
        - Configurer Gradle ou Maven pour la gestion des dépendances
        - Vérifier la compilation et l'exécution minimale de l'application
    - **Achraf :**
        - Configurer Visual Studio Code avec les extensions nécessaires (Java, Git, JavaFX)
        - Configurer le projet pour les tests unitaires (JUnit 5)

        - Créer la structure initiale des packages et des classes vides
    - **Léo :**
        - Préparer l'architecture des dossiers (src/, doc/, data/, test/)
        - Créer les premiers fichiers README et .gitignore
        - Vérifier l'intégration Git pour que tous les membres puissent pousser et récupérer le code
2. **Prototype fonctionnel initial**
    - **Iskander :**
        - Implémenter l'écran d'accueil avec menu principal
        - Préparer les classes Player et Track avec attributs principaux
    - **Achraf :**
        - Créer le GameController avec méthodes startGame(), nextRound()
        - Implémenter Round avec gestion de l'extrait audio simulé (mock)
    - **Léo :**
        - Développer Playlist et PlaylistService pour charger/sauvegarder des playlists JSON
        - Tester la communication entre GameController et les classes modèle

Durée estimée pour Sprint 1 : 06/10/2025 → 16/10/2025

Livrable attendu :

- Projet initial compilable avec structure des packages prête
- Prototype minimal fonctionnel : écran d'accueil + simulation d'une manche de jeu
- Dépôt Git partagé avec commits réguliers pour chaque tâche