Guide développeur de Devine-Mot

08/01/2025
Développement Mobile

MONNIER Sohail
Morisseau Noa

Sommaire :

Table des matières

Contexte

L’application DevineMot : Quick, Draw! The Data est un projet novateur qui combine amusement, créativité et interaction avec une technologie moderne. Inspirée par l’API de Quick, Draw! de Google, cette application transforme des esquisses simples, souvent absurdes et rudimentaires, en un jeu interactif où les joueurs doivent deviner ce que représentent ces dessins. Le jeu exploite une base de données publique alimentée par des millions d'utilisateurs du monde entier, permettant ainsi une variété infinie de contenus.

Le cœur du projet repose sur une idée simple mais efficace : transformer des dessins choisis de manière aléatoire en une expérience ludique grâce à des choix de réponses humoristiques et parfois totalement décalés. L’ambition principale est de proposer une plateforme qui transcende le simple divertissement en intégrant également une dimension sociale et communautaire. La galerie des "Horrible Drawings" joue un rôle central dans cette dimension, permettant aux joueurs de partager, commenter et voter pour les dessins les plus drôles ou les plus incompréhensibles.

Cependant, nous avons rencontré certaines limitations techniques et financières. L'API de traduction de Google que nous avions prévue pour internationaliser le jeu est payante, ce qui a empêché son intégration complète dans le projet. Par ailleurs, la fonctionnalité de galerie communautaire, bien qu'implémentée côté back-end, ne fonctionne pas encore comme prévu pour enregistrer les dessins qui plaisent le plus aux utilisateurs. Malgré cela, les scores sont correctement sauvegardés pour chaque utilisateur dans la base de données, mais ne peuvent pas encore être récupérés et affichés correctement sur le front-end.

Malgré ces défis, l’accessibilité a été une priorité tout au long du développement. L’application est conçue pour fonctionner aussi bien en ligne qu'en mode hors-ligne, garantissant une expérience utilisateur optimale quelles que soient les conditions de connexion.

Fonctionnalités

L'application DevineMot propose une expérience interactive unique, où le joueur est directement impliqué dans le processus de devinette. Contrairement aux jeux traditionnels de quiz, l'utilisateur n'est pas limité à choisir parmi des réponses prédéfinies. Lorsqu'un dessin s'affiche à l'écran, c'est à lui de proposer sa propre réponse en la saisissant au clavier. Ce système favorise la réflexion et la créativité, tout en ajoutant un élément de défi. L'utilisateur doit interpréter l'image souvent simplifiée ou rudimentaire pour deviner ce qu'elle représente. Cette approche rend chaque partie différente, car les dessins générés varient et exigent des réponses personnalisées.

L'un des aspects ambitieux du projet est la galerie communautaire, nomée "Horrible Drawings". Cette fonctionnalité, bien qu'encore partiellement réalisée, incarne l'esprit social et interactif de l'application. L'objectif est de permettre aux utilisateurs de voter pour leurs dessins favoris parmi une collection regroupant les esquisses les plus drôles, étranges ou incompréhensibles. Actuellement, le système de vote est opérationnel côté serveur, et les données sont correctement enregistrées dans la base de données. Cependant, la récupération et l'affichage de ces dessins dans l'interface utilisateur restent en développement. Une fois finalisée, cette galerie deviendra un espace central de l'application, favorisant l'échange et l'engagement communautaire, où les joueurs pourront explorer, commenter et valoriser les créations des autres utilisateurs.

En parallèle, l'application offre un système d'authentification conçu pour être à la fois flexible et sécurisé. Les utilisateurs peuvent jouer en tant qu'invités, permettant une entrée rapide dans le jeu sans passer par un processus de connexion. Pour ceux qui souhaitent sauvegarder leurs performances, accéder au classement général, ou profiter de fonctionnalités avancées, une option d'inscription est disponible. Le système repose sur l'utilisation de JSON Web Tokens (JWT), qui assurent une gestion sécurisée des sessions utilisateurs. Les informations sensibles, comme les mots de passe, sont hachées avant d'être stockées dans la base de données, garantissant une protection élevée des données personnelles.

Enfin, l'application intègre un mode hors-ligne pour garantir une expérience fluide même en l'absence de connexion Internet. Les dessins sont préchargés localement sur l'appareil, permettant ainsi aux joueurs de continuer à deviner et s'amuser dans toutes les conditions, qu'ils soient en déplacement ou dans des zones à faible connectivité. Cette fonctionnalité reflète l'engagement du projet à offrir une accessibilité universelle, tout en maintenant une qualité d'expérience élevée.

Les classes et interfaces utilisées

L’architecture de l’application repose sur une séparation claire entre le front-end et le back-end, permettant ainsi une conception modulaire, maintenable et évolutive. Le front-end, développé en Flutter (Dart), assure une interface utilisateur moderne, fluide et réactive. Cette technologie a été choisie pour sa capacité à générer des applications multiplateformes tout en offrant une expérience utilisateur homogène.

Le back-end, quant à lui, est basé sur Node.js avec le framework Express.js. Ce choix technologique garantit une gestion optimale des requêtes API et une communication rapide avec la base de données. La base de données, quant à elle, repose sur MongoDB, un système NoSQL particulièrement adapté pour stocker et organiser les données hétérogènes générées par les utilisateurs et l’API Quick, Draw!.

Les échanges entre le front-end et le back-end se font via une série d’API REST, qui assurent une communication rapide et sécurisée entre les différentes couches de l'application. Cela inclut des appels pour récupérer des dessins, soumettre des votes, ou encore gérer l’authentification des utilisateurs. L’utilisation de JWT (JSON Web Tokens) permet également une gestion sécurisée des sessions utilisateurs et des autorisations d’accès aux ressources protégées.

Cette architecture modulaire permet non seulement une meilleure maintenabilité du code, mais aussi une flexibilité accrue pour intégrer de nouvelles fonctionnalités ou s'adapter à des évolutions futures.

L'architecture du projet repose sur une organisation rigoureuse des classes et des interfaces. Chaque module respecte les principes de **séparation des responsabilités** et les bonnes pratiques de conception logicielle.

Frontend (Flutter/Dart)

Classe principale : GameScreen
La classe GameScreen est responsable de l’affichage principal du jeu. Elle orchestre les interactions utilisateur, telles que la soumission de réponses et l’affichage des résultats. Elle intègre également les appels API pour récupérer les dessins et gérer les scores.

Classe AuthService
Cette classe est dédiée à la gestion de l’authentification. Elle contient les méthodes suivantes :

- login(email, password) : Authentifie un utilisateur et stocke le token JWT.
- register(name, email, password) : Crée un nouvel utilisateur.
- getProfile() : Récupère les informations du profil utilisateur.

Classe DrawingService
La classe DrawingService gère les appels API liés aux dessins. Elle propose les méthodes :

- getRandomDrawing() : Récupère un dessin aléatoire avec ses options.
- voteForDrawing(drawingId) : Soumet un vote pour un dessin spécifique.
- getGallery() : Retourne les dessins les plus votés.

Classe ScoreService
Cette classe est responsable de l'envoi et de la récupération des scores.

- submitScore(score) : Envoie le score obtenu après une partie.
- getLeaderboard() : Récupère le classement général des joueurs.

Interface IUser
L'interface IUser définit la structure des objets utilisateur.

Interface IDrawing
L'interface IDrawing spécifie les données attendues pour les objets dessin.

Backend (Node.js / Express.js)

Classe AuthController
Cette classe contient la logique métier pour gérer l'authentification :

- register(req, res) : Crée un nouvel utilisateur après validation des données.
- login(req, res) : Vérifie les identifiants et génère un token JWT.

Classe DrawingController
La logique principale des dessins est centralisée ici :

- getRandomDrawing(req, res) : Sélectionne et retourne un dessin aléatoire depuis la base de données.
- voteForDrawing(req, res) : Incrémente le compteur de votes pour un dessin.
- getGallery(req, res) : Retourne une liste des dessins les plus votés.

Classe ScoreController
Cette classe gère les scores des joueurs :

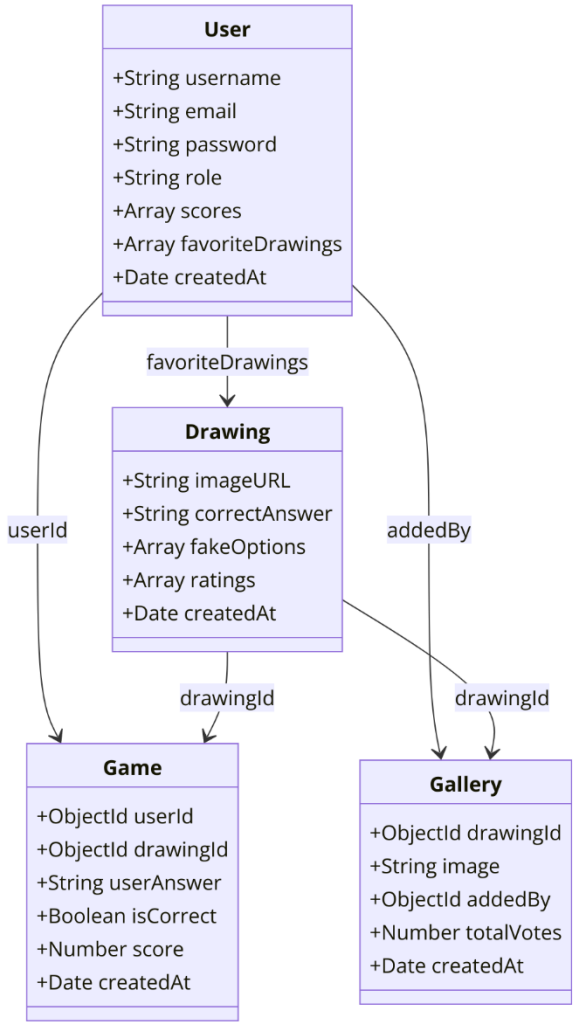
- submitScore(req, res) : Enregistre un score dans la base de données.
- getLeaderboard(req, res) : Récupère le classement global.

Le modèle de données manipulées par l’application

Le modèle de données repose sur une organisation structurée pour répondre aux besoins spécifiques de l’application. Les informations des utilisateurs, les dessins et les parties jouées sont stockés sous forme de collections distinctes dans MongoDB.

La collection Utilisateur contient les informations personnelles essentielles, telles que l'identifiant unique, le nom, l’email, le mot de passe et le score global. Cette structure garantit une gestion efficace des profils utilisateurs et permet de suivre les performances individuelles.

La collection Dessin regroupe les informations relatives aux esquisses récupérées depuis l’API Quick, Draw!. Chaque dessin est accompagné d’une URL d’accès, d’une description textuelle et d’un compteur de votes permettant de classer les plus populaires.
Enfin, la collection Partie Joueur permet d’enregistrer les performances spécifiques des utilisateurs pour chaque session de jeu, incluant le score obtenu et le temps passé.



Une description des algorithmes utilisés

L'un des algorithmes principaux de l'application est l'algorithme de sélection aléatoire de dessins, qui garantit une expérience diversifiée à chaque session de jeu. Cet algorithme puise dans la base de données et sélectionne de manière aléatoire un dessin unique, tout en générant deux fausses réponses humoristiques pour compléter le quiz.

Un autre algorithme clé est le système de vote pour la galerie Horrible Drawings. Chaque utilisateur a la possibilité de voter pour ses dessins préférés, mais une vérification est effectuée pour s'assurer qu'un même utilisateur ne peut pas voter plusieurs fois pour le même dessin.

Enfin, l’authentification sécurisée par JWT permet de garantir la sécurité des connexions et des sessions utilisateurs. Les mots de passe sont hachés avant stockage, et chaque session active est vérifiée à chaque requête API.

Routes API

Les routes API constituent le cœur de la communication entre le **front-end** et le **back-end** de l'application. Elles permettent aux différentes couches de l'application d'interagir de manière fluide et sécurisée. Chaque route est soigneusement définie pour répondre à des besoins spécifiques, qu'il s'agisse de l'authentification des utilisateurs, de la gestion des dessins, ou encore du stockage des scores.

Authentification et Gestion des Utilisateurs
L'authentification est gérée via des tokens **JWT (JSON Web Token)**, garantissant une connexion sécurisée pour chaque utilisateur enregistré.

- **POST /auth/register**
  Permet aux nouveaux utilisateurs de s'inscrire sur la plateforme en fournissant un nom, un email et un mot de passe. Une vérification de l'unicité de l'email est effectuée avant d'ajouter l'utilisateur à la base de données.
- **POST /auth/login**
  Gère la connexion des utilisateurs existants. Les identifiants sont vérifiés et un token JWT est généré pour les sessions sécurisées.
- **GET /auth/profile**
  Retourne les informations du profil utilisateur connecté grâce au token d'authentification.

Gestion des Dessins et Quiz
Les routes liées aux dessins permettent d'interagir directement avec la base de données **Quick, Draw!** et d'alimenter le jeu avec des contenus dynamiques.

- **GET /drawings/random**
  Retourne un dessin sélectionné aléatoirement avec trois options de réponses : une correcte et deux absurdes.
- **POST /drawings/vote**
  Permet aux utilisateurs de voter pour un dessin dans la galerie des **"Horrible Drawings"**. Un système de contrôle empêche les doublons de votes par utilisateur.
- **GET /drawings/gallery**
  Récupère une liste des dessins les plus populaires en fonction du nombre de votes reçus.

Gestion des Scores et Classements
Pour encourager la compétition amicale, des routes spécifiques sont mises en place pour gérer les scores et les classements.

- **POST /score/submit**
  Enregistre le score obtenu par un joueur après une partie. Chaque session est liée à un utilisateur identifié.
- **GET /leaderboard**
  Retourne le classement général des meilleurs joueurs, trié par score décroissant.

Conclusion

Le projet DevineMot **est** une base solide pour une application ludique et innovante. Malgré les défis techniques, notamment autour de l'API de traduction payante et des fonctionnalités incomplètes de la galerie et du classement, les fondations techniques sont robustes et prometteuses. Des améliorations futures incluront la résolution des problèmes de récupération des scores, le raffinement de la galerie communautaire et l'intégration complète des traductions pour une expérience utilisateur encore plus accessible.