ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR

Membre de

HONORIS UNITED UNIVERSITIES

Rapport de Projet

Sujet:

Développement d'une application web de gestion des sondages

Réalisé par:

M. Lhoussaine GHALLOU

Encadré par

M. Khalid NAFIL

Introduction

Dans l'ère numérique actuelle, la collecte d'opinions et de feedbacks constitue un élément essentiel pour la prise de décisions éclairées, que ce soit dans un contexte académique, professionnel ou social. Les sondages en ligne représentent un moyen efficace et moderne de recueillir des données auprès d'un large public de manière structurée et organisée. Cependant, de nombreuses solutions existantes présentent des limitations en termes de simplicité d'utilisation, de flexibilité ou d'accessibilité.

Le présent projet s'inscrit dans cette problématique en proposant le développement d'une application web moderne de gestion de sondages. Cette solution vise à offrir une plateforme intuitive permettant aux utilisateurs de créer facilement des sondages personnalisés, de les partager via des liens ou des codes d'accès, et de consulter les résultats en temps réel.

L'application repose sur une architecture moderne séparant clairement le frontend du backend. Le choix technologique s'est porté sur Django REST Framework pour l'API backend, reconnu pour sa robustesse et sa sécurité, couplé à React pour l'interface utilisateur, garantissant une expérience utilisateur fluide et responsive. Cette architecture permet une grande modularité, facilitant la maintenance et l'évolution future de l'application.

La solution développée permet aux utilisateurs authentifiés de créer des sondages avec des choix simples ou multiples, de gérer leurs publications, et de suivre les statistiques de participation. Les votants peuvent participer aux sondages via des liens partagés ou des codes d'accès, sans nécessiter obligatoirement de création de compte. L'authentification par token JWT assure la sécurité des données et la gestion des sessions utilisateur.

Ce projet a été mené selon une approche méthodique, débutant par l'analyse des besoins et la conception de l'API REST, suivie du développement du backend avec Django, puis de la création de l'interface utilisateur avec React et les composants shadcn/ui. L'intégration des différents modules et les tests ont constitué les phases finales du développement.

L'objectif de ce rapport est de présenter de manière détaillée le processus de développement, les choix techniques effectués, l'architecture mise en place, ainsi que les fonctionnalités implémentées dans l'application finale.

Objectifs du projet

L'objectif principal de ce projet est de concevoir et développer une application web complète de gestion de sondages, basée sur une architecture moderne REST API et une interface utilisateur interactive. Il s'agit de fournir un outil accessible, performant et sécurisé permettant la création, la gestion et la participation à des sondages en ligne.

Objectifs fonctionnels :

• **Gestion des sondages** : permettre la création, modification, suppression et consultation de sondages avec questions personnalisables et choix multiples • **Système de vote** : faciliter la participation aux sondages via des liens partagés ou des codes d'accès • **Authentification sécurisée** : gérer l'inscription, la connexion et les sessions utilisateur via JWT • **Partage et accessibilité** : permettre le partage facile des sondages et la participation sans barrières techniques • **Suivi des statistiques** : fournir des données de participation et des résultats en temps réel

Objectifs techniques :

• **Architecture REST** : développer une API robuste avec Django REST Framework respectant les standards REST • **Interface moderne** : créer une interface utilisateur responsive avec React et les composants shadcn/ui • **Sécurité** : implémenter une authentification JWT sécurisée avec gestion des tokens d'accès et de rafraîchissement • **Base de données relationnelle** : structurer efficacement les données avec un modèle relationnel optimisé • **Séparation des responsabilités** : maintenir une architecture découplée entre frontend et backend

Objectifs pédagogiques :

• **Maîtrise des technologies web modernes** : approfondir React, Django REST Framework et l'écosystème JavaScript/Python • **Conception d'API REST** : comprendre et appliquer les principes de conception d'APIs RESTful • **Gestion de l'état et routage** : manipuler l'état des applications React et le routage côté client • **Sécurité web** : implémenter des mécanismes d'authentification et d'autorisation robustes

---

Technologies utilisées

La réalisation de l'application de sondages s'appuie sur un stack technologique moderne et éprouvé, combinant des outils backend et frontend performants pour assurer scalabilité, maintenabilité et expérience utilisateur optimale.

Backend - API REST

Django 5.2 Framework web Python mature et sécurisé, choisi pour sa robustesse, sa documentation exhaustive et son écosystème riche. Django fournit une base solide avec son ORM intégré, son système d'authentification et ses mécanismes de sécurité.

Django REST Framework 3.16.0 Extension de Django spécialisée dans la création d'APIs REST. Elle offre des sérialiseurs puissants, un système de

permissions granulaire et une interface de navigation automatique pour les APIs.

djangorestframework_simplejwt 5.5.0 Implémentation JWT (JSON Web Tokens) pour Django REST Framework, permettant une authentification stateless sécurisée avec gestion des tokens d'accès et de rafraîchissement.

Base de données Utilisation de la base de données par défaut de Django (SQLite en développement, PostgreSQL recommandé en production) avec migrations automatiques et ORM intégré.

Frontend - Interface utilisateur

React 19.1.0 Bibliothèque JavaScript moderne pour la construction d'interfaces utilisateur interactives, choisie pour sa performance, sa communauté active et son écosystème riche.

Vite 6.3.5 Outil de build rapide et moderne remplaçant les solutions traditionnelles comme Create React App, offrant un développement plus fluide avec rechargement à chaud optimisé.

TypeScript 5.8.3 Superset de JavaScript ajoutant un typage statique, améliorant la qualité du code et la productivité de développement.

Interface utilisateur et styling

Tailwind CSS 4.1.5 Framework CSS utility-first permettant un développement rapide avec des classes utilitaires, garantissant une cohérence visuelle et une maintenance facilitée.

shadcn/ui Collection de composants React réutilisables basés sur Radix UI, offrant des composants accessibles et personnalisables (boutons, formulaires, modales, etc.).

Radix UI Bibliothèque de composants primitifs non-stylés, garantissant l'accessibilité et les bonnes pratiques UX.

Gestion des données et routing

Axios 1.9.0 Client HTTP pour JavaScript, facilitant les appels API avec intercepteurs, gestion d'erreurs et transformation automatique des données.

React Router 7.5.3 Bibliothèque de routage déclaratif pour React, permettant la navigation côté client et la gestion des URLs.

Recharts 2.15.3 Bibliothèque de graphiques React pour la visualisation des résultats de sondages et statistiques.

Outils de développement

ESLint & TypeScript ESLint Outils de linting pour maintenir la qualité du code et respecter les conventions de développement.

Vite Plugin React Plugin Vite optimisé pour le développement React avec Fast Refresh et optimisations spécifiques.

Architecture de l'application

L'application de sondages suit une architecture moderne séparée en deux parties distinctes : un backend API REST développé avec Django et un frontend SPA (Single Page Application) développé avec React. Cette séparation permet une meilleure scalabilité, maintenabilité et possibilité d'évolution.

Vue d'ensemble de l'architecture

L'architecture du projet repose sur les couches suivantes :

1. Couche API Backend (Django REST Framework) Modèles de données

- **User** : modèle utilisateur Django étendu pour l'authentification
- **Poll** : représente un sondage avec question, description, options et métadonnées
- **Choice** : représente les choix disponibles pour chaque sondage
- **Vote** : enregistre les votes des utilisateurs avec horodatage

Vues API (ViewSets et APIViews)

- **AuthViewSet** : gestion de l'authentification (login, signup, logout, refresh token)
- **PollViewSet** : CRUD des sondages avec permissions appropriées
- **VoteAPIView** : gestion des votes avec validation
- **UserStatsAPIView** : statistiques utilisateur

Système d'authentification JWT

- Tokens d'accès à durée limitée pour les requêtes API
- Tokens de rafraîchissement pour renouveler l'accès
- Blacklist des tokens lors de la déconnexion

2. Couche Frontend React Composants de présentation Composants réutilisables basés sur shadcn/ui pour l'interface utilisateur (formulaires, boutons, cartes, etc.)

Pages et routage

- Page d'accueil avec liste des sondages
- Pages d'authentification (connexion/inscription)
- Page de création de sondage
- Page de vote et résultats
- Dashboard utilisateur

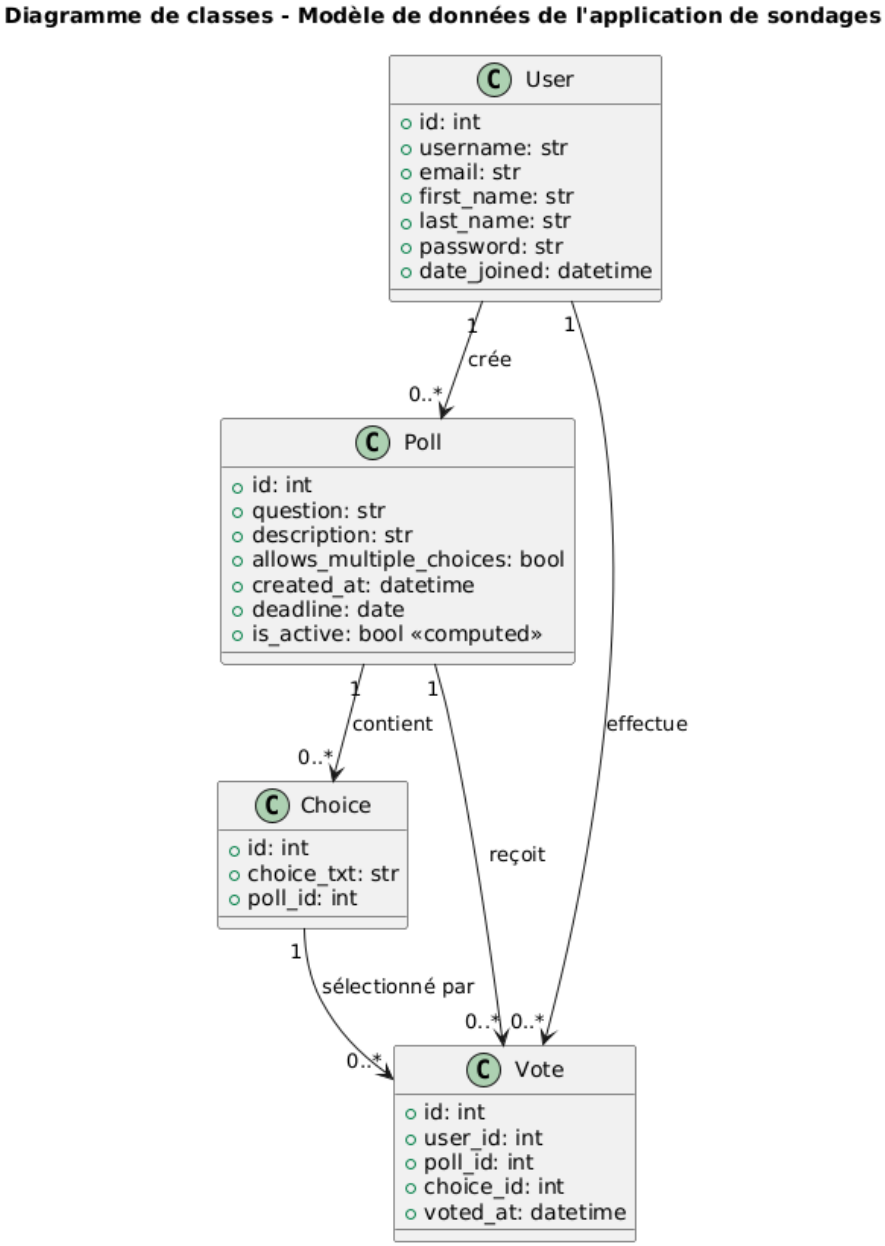

Figure 1: diagramme de classe

Gestion d'état

- État local avec React hooks (useState, useEffect)
- Contexte React pour l'authentification globale
- Gestion des appels API avec Axios

3. Couche de communication (API REST) Endpoints principaux

```
POST /polls/api/auth/login/ - Connexion utilisateur
POST /polls/api/auth/signup/ - Inscription utilisateur
GET /polls/api/polls/ - Liste des sondages
POST /polls/api/polls/create/ - Création de sondage
POST /polls/api/polls/vote/ - Soumission de vote
GET /polls/api/user_stats/ - Statistiques utilisateur
```

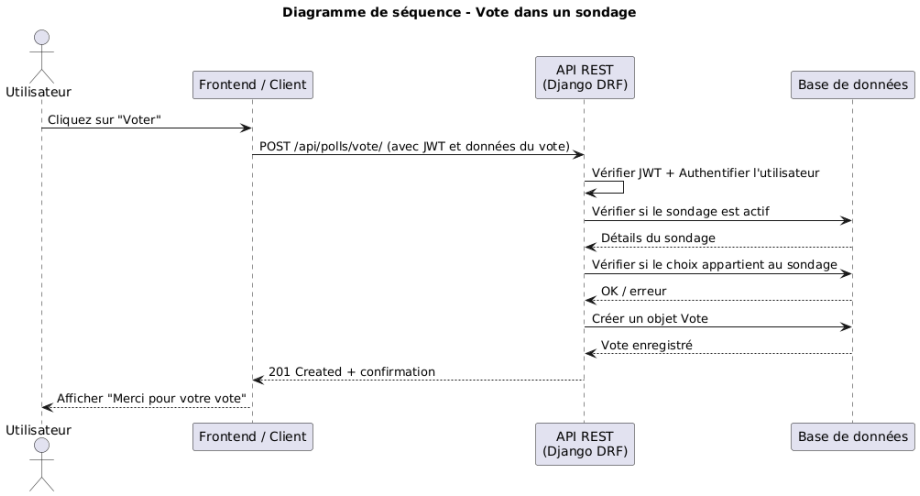

Figure 2: diagramme séquentielle

Sérialisation des données Utilisation des sérialiseurs Django REST Framework pour la validation et transformation des données entre JSON et modèles Django.

Avantages de cette architecture

- **Séparation des responsabilités** : API backend indépendante du frontend
- **Scalabilité** : possibilité de développer des clients mobiles ou autres interfaces
- **Sécurité** : authentification centralisée avec JWT
- **Performance** : SPA React pour une expérience utilisateur fluide
- **Maintenabilité** : code organisé en modules réutilisables

---

Modélisation des données

La structure de données de l'application repose sur un modèle relationnel simple mais efficace, centré autour des entités principales : utilisateurs, sondages, choix et votes.

Modèle de données principal

Entité User (Utilisateur) Utilise le modèle User de Django avec les champs standards :

- `username` : nom d'utilisateur unique
- `email` : adresse email
- `first_name`, `last_name` : nom et prénom
- `password` : mot de passe hashé
- `date_joined` : date d'inscription

Entité Poll (Sondage)

```
class Poll:
    - id: clé primaire auto-incrémentée
    - question: texte de la question (max 90 caractères)
    - description: description détaillée (max 220 caractères)
    - user: clé étrangère vers User (créateur)
    - allows_multiple_choices: booléen pour choix multiples
    - created_at: timestamp de création
    - deadline: date limite de vote
    - is_active: statut actif/inactif
```

Entité Choice (Choix)

```
class Choice:
    - id: clé primaire
    - choice_txt: texte du choix (max 50 caractères)
    - poll: clé étrangère vers Poll
    - vote_count: nombre de votes (calculé)
```

Entité Vote (Vote)

```
class Vote:
    - id: clé primaire
    - user: clé étrangère vers User
    - poll: clé étrangère vers Poll
    - choice: clé étrangère vers Choice
    - created_at: timestamp du vote
```

Relations entre entités

- **User → Poll** : relation 1-N (un utilisateur peut créer plusieurs sondages)
- **Poll → Choice** : relation 1-N (un sondage a plusieurs choix possibles)
- **User → Vote** : relation 1-N (un utilisateur peut voter plusieurs fois)
- **Poll → Vote** : relation 1-N (un sondage reçoit plusieurs votes)
- **Choice → Vote** : relation 1-N (un choix peut être voté plusieurs fois)

Contraintes et validations

- **Unicité des votes** : un utilisateur ne peut voter qu'une fois par sondage (sauf choix multiples)
- **Validation des dates** : la deadline doit être future à la création
- **Validation des choix** : au moins 2 choix requis par sondage
- **Permissions** : seul le créateur peut modifier/supprimer son sondage

---

Fonctionnalités principales

L'application de sondages offre un ensemble complet de fonctionnalités permettant la création, gestion et participation à des sondages en ligne de manière intuitive et sécurisée.

1. Authentification et gestion des utilisateurs

Inscription d'utilisateurs

- Formulaire d'inscription avec validation côté client et serveur
- Vérification de l'unicité du nom d'utilisateur et email
- Hashage sécurisé des mots de passe avec Django

Connexion sécurisée

- Authentification par JWT avec tokens d'accès et de rafraîchissement
- Gestion automatique de l'expiration des tokens
- Déconnexion avec blacklist des tokens

Profil utilisateur

- Consultation des informations personnelles via endpoint `/whoami/`
- Statistiques utilisateur (nombre de sondages créés, votes effectués)

2. Gestion des sondages

Création de sondages

- Interface intuitive pour créer des sondages personnalisés
- Saisie de question, description et choix multiples
- Configuration des options (choix simple/multiple, date limite)
- Validation des données avant soumission

Liste et consultation des sondages

- Affichage paginé de tous les sondages actifs
- Consultation détaillée d'un sondage spécifique
- Accès public aux sondages sans authentification requise

Administration des sondages

- Gestion complète via l'interface d'administration Django
- Modification et suppression des sondages par leurs créateurs
- Suivi des statistiques de participation

3. Système de vote

Participation aux sondages

- Vote via interface web responsive
- Validation des votes côté serveur
- Prévention des votes multiples par utilisateur
- Support des choix multiples selon configuration

Résultats en temps réel

- Affichage automatique des résultats après vote
- Graphiques de visualisation avec Recharts
- Statistiques détaillées par choix

4. API REST complète

Endpoints d'authentification

```
POST /polls/api/auth/login/ - Connexion
POST /polls/api/auth/signup/ - Inscription
POST /polls/api/auth/logout/ - Déconnexion
POST /polls/api/auth/refresh_token/ - Renouvellement token
GET /polls/api/auth/whoami/ - Informations utilisateur
```

Endpoints de sondages

```
GET /polls/api/polls/ - Liste des sondages
GET /polls/api/polls/{id}/ - Sondage spécifique
POST /polls/api/polls/create/ - Création sondage
POST /polls/api/polls/vote/ - Soumission vote
```

5. Interface utilisateur moderne

Design responsive

- Interface adaptée mobile/desktop avec Tailwind CSS
- Composants accessibles basés sur Radix UI
- Thème sombre/clair avec next-themes

Navigation intuitive

- Routage côté client avec React Router
- Breadcrumb et navigation contextuelle
- Messages de feedback avec Sonner (notifications)

Formulaires interactifs

- Validation en temps réel des saisies
- Messages d'erreur contextuels
- États de chargement et feedback utilisateur

6. Sécurité et permissions

Authentification JWT

- Tokens sécurisés avec expiration automatique
- Refresh tokens pour sessions longues
- Blacklist des tokens lors déconnexion

Permissions granulaires

- Accès public aux sondages en lecture
- Création restreinte aux utilisateurs connectés
- Modification limitée aux propriétaires

Validation des données

- Sérialisation stricte avec Django REST Framework
- Validation côté client et serveur
- Protection CSRF et headers CORS configurés

Interface utilisateur

L'interface de l'application a été conçue avec une approche mobile-first, privilégiant la simplicité d'utilisation et l'accessibilité. Elle repose sur des composants modernes et des interactions fluides pour offrir une expérience utilisateur optimale.

Design et ergonomie

Système de design cohérent

- Palette de couleurs harmonieuse avec support du mode sombre
- Typographie claire avec hiérarchie visuelle définie
- Espacement et proportions basés sur Tailwind CSS
- Composants réutilisables avec shadcn/ui

Navigation principale

- Header fixe avec logo et menu utilisateur
- Navigation responsive avec menu hamburger sur mobile
- Breadcrumb pour orienter l'utilisateur
- Footer avec liens utiles et informations

Pages principales

Page d'accueil

- Liste des sondages actifs avec aperçu
- Système de recherche et filtrage
- Pagination pour grandes listes
- Call-to-action pour créer un sondage

Interface de création de sondage

- Formulaire en étapes (wizard) pour guider l'utilisateur
- Ajout dynamique de choix avec boutons +/-
- Prévisualisation en temps réel
- Validation instantanée des champs

Page de vote

- Affichage clair de la question et description
- Choix présentés avec radio buttons ou checkboxes
- Bouton de soumission avec état de chargement
- Transition fluide vers les résultats

Tableau de bord utilisateur

- Vue d'ensemble des sondages créés
- Statistiques personnelles
- Gestion des sondages (édition/suppression)
- Historique des votes

Composants interactifs

Formulaires

- Champs de saisie avec validation en temps réel
- Messages d'erreur contextuels
- États de chargement avec spinners
- Sauvegarde automatique des brouillons

Graphiques et visualisations

- Graphiques circulaires et barres avec Recharts
- Animations fluides lors du chargement des données
- Couleurs cohérentes avec le thème de l'application

- Tooltips informatifs

Notifications et feedback

- Toasts pour les actions réussies/échouées
- Messages de confirmation pour actions importantes
- Indicateurs de progression pour les opérations longues
- États vides avec illustrations et appels à l'action

Accessibilité et responsive design

Accessibilité

- Contraste de couleurs respectant les standards WCAG
- Navigation au clavier complète
- ARIA labels sur les éléments interactifs
- Support des lecteurs d'écran

Responsive design

- Interface adaptée du mobile (320px) au desktop (1920px+)
- Grille flexible avec breakpoints Tailwind
- Images et médias adaptatifs
- Touch-friendly sur appareils tactiles

Conclusion

Ce projet de développement d'une application de sondages a permis de créer une solution web complète et moderne, répondant aux besoins de création, gestion et participation à des sondages en ligne. L'approche méthodique adoptée, combinant une API REST robuste avec Django et une interface utilisateur interactive avec React, a abouti à une application performante et évolutive.

Réalisations techniques

L'architecture mise en place respecte les bonnes pratiques du développement web moderne avec une séparation claire entre backend et frontend. L'API REST développée avec Django REST Framework offre une base solide avec authentification JWT sécurisée, validation des données et gestion fine des permissions. L'interface React, enrichie des composants shadcn/ui et du styling Tailwind CSS, propose une expérience utilisateur fluide et accessible.

Le modèle de données relationnel simple mais efficace permet de gérer les entités principales (utilisateurs, sondages, choix, votes) avec des relations cohérentes et des contraintes appropriées. L'utilisation de l'ORM Django facilite les opérations de base de données tout en maintenant la sécurité.

Fonctionnalités implémentées

L'application couvre l'ensemble du cycle de vie des sondages, depuis la création jusqu'à la consultation des résultats. Les utilisateurs peuvent s'inscrire, se connecter de manière sécurisée, créer des sondages personnalisés avec choix simples ou multiples, et partager leurs sondages. Les participants peuvent voter facilement et consulter les résultats en temps réel via des graphiques intuitifs.

Le système d'authentification JWT garantit la sécurité des sessions utilisateur, tandis que les permissions granulaires protègent l'intégrité des données. L'interface responsive assure une utilisation optimale sur tous types d'appareils.

Perspectives d'évolution

Cette base solide ouvre plusieurs possibilités d'amélioration et d'extension. L'ajout de fonctionnalités temps réel avec WebSockets permettrait une mise à jour instantanée des résultats. L'intégration d'un système de notifications push enrichirait l'engagement utilisateur. Des fonctionnalités avancées comme l'analyse statistique approfondie, l'export de données, ou la personnalisation des thèmes pourraient être développées.

L'architecture modulaire facilite l'ajout de nouvelles fonctionnalités sans impacter l'existant. Le passage à une base de données PostgreSQL en production permettrait de gérer des volumes de données plus importants.

Apports pédagogiques

Ce projet a permis d'approfondir la maîtrise des technologies web modernes, particulièrement l'écosystème React et Django REST Framework. La conception d'une API REST complète, l'implémentation d'un système d'authentification sécurisé, et le développement d'une interface utilisateur moderne constituent des compétences transférables vers d'autres projets.

La gestion d'un projet full-stack, de la conception à la réalisation, a développé une vision globale du développement web et une compréhension des enjeux de scalabilité, sécurité et expérience utilisateur.

Cette application de sondages représente ainsi une réalisation complète et fonctionnelle, démontrant la maîtrise des technologies web contemporaines et constituant une base solide pour de futurs développements.