AVIGNON UNIVERSITÉ

Projet : Application de suivi des rappels de produits

Groupe TD3
GENTI Anthony
BOUDOUNT Youssef

15/05/2024

Licence Informatique L3
Parcours LININGLOG : INGENIERIE LOGICIELLE
UE Applications Mobiles

Responsable
MERLIN Eric
VERNET Mathilde

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Introduction

1.1 Contexte du projet

Le projet de fin de semestre, intitulé "Application de suivi des rappels de produits," s'inscrit dans le cadre de l'UE Applications Mobiles de la Licence 3. Ce cours a pour objectif de nous fournir, nous les étudiants, une compréhension approfondie et pratique du développement d'applications mobiles sur la plateforme Android. Les concepts enseignés durant ce cours, ainsi que les travaux pratiques réalisés précédemment, ont permis de bâtir une base solide pour aborder ce projet. Contrairement aux autres TPs, où des squelettes de projets étaient fournis, ce projet doit être développé entièrement 'from scratch', offrant ainsi une opportunité unique de mettre en œuvre l'ensemble des nos compétences acquises.

1.2 Objectif du projet

L'objectif principal de cette application est de gérer des alertes de rappels de produits en utilisant l'API du gouvernement sur les rappels aux consommateurs. Les rappels de produits, qu'ils concernent des produits alimentaires, des jouets, ou d'autres biens de consommation. Ils informent le public des risques potentiels associés à certains produits et les mesures à prendre pour éviter des problèmes de santé ou de sécurité. Cette application permettra aux utilisateurs de rechercher des alertes de rappels de produits, de lire en détail les informations sur ces rappels, et de sauvegarder leurs rappels favoris pour un accès ultérieur facile.

L'application vise également à fournir une interface utilisateur intuitive et efficace, facilitant la navigation entre les différentes fonctionnalités. Les utilisateurs pourront configurer l'application selon leurs préférences, notamment en définissant des critères spécifiques pour la recherche et en limitant le nombre de résultats affichés. Cette flexibilité permettra à l'application de répondre aux besoins individuels des utilisateurs de manière optimale.

2 Description de l'application

2.1 Fonctionnalités principales

2.1.1 Recherche des rappels de produits

La fonctionnalité de recherche des rappels de produits est au cœur de notre application. Elle permet aux utilisateurs de rechercher facilement les produits rappelés via une barre de recherche intuitive. Les résultats de la recherche sont présentés sous forme de cartes contenant une image du produit rappelé, le nom du produit, ainsi qu'une information supplémentaire choisie pour chaque carte, comme la date de rappel ou la catégorie du produit. Par défaut, lors de l'ouverture de l'application, une liste des derniers rappels de produits est affichée afin de fournir immédiatement une information pertinente sans nécessiter une recherche initiale. Cela assure que les utilisateurs sont toujours informés des rappels les plus récents et potentiellement dangereux.

Chaque carte dans la liste des résultats comporte un menu contextuel permettant d'ajouter le produit aux favoris. En cliquant sur une carte, l'utilisateur est redirigé vers une page détaillée du rappel du produit, où toutes les informations pertinentes sont affichées, y compris les raisons du rappel, les actions recommandées, et d'autres détails importants. Sur cette page, un bouton permet également d'ajouter le produit aux favoris pour un accès facile ultérieur.

2.1.2 Lecture détaillée des rappels

La lecture détaillée des rappels est une extension directe de la fonctionnalité de recherche. Lorsqu'un utilisateur sélectionne un rappel de produit dans la liste de résultats, il est dirigé vers une nouvelle page où il peut lire en détail toutes les informations relatives au rappel. Cette page inclut des descriptions complètes, des images supplémentaires, les dates pertinentes, et les actions recommandées par les autorités pour gérer le rappel du produit. Un bouton permet de revenir facilement à la page de recherche, garantissant ainsi une navigation fluide et sans friction.

2.1.3 Gestion des favoris

La gestion des favoris est une fonctionnalité cruciale qui permet aux utilisateurs de sauvegarder les rappels de produits les plus pertinents pour eux. Les utilisateurs peuvent ajouter des produits à leur liste de favoris depuis la page de recherche ou depuis la page de détails du rappel. Une page dédiée à la gestion des favoris affiche tous les produits enregistrés sous forme de cartes. Chaque carte contient plusieurs éléments d'information sur le produit rappelé, permettant une vue d'ensemble rapide et efficace. Un menu contextuel sur chaque carte permet aux utilisateurs de supprimer un produit de la liste des favoris ou de copier le lien de l'offre dans le presse-papiers. En cliquant sur une carte, l'utilisateur peut accéder à une page détaillée du rappel similaire à celle de la recherche, avec un bouton supplémentaire pour retirer le produit des favoris.

De plus, un menu de filtrage permet de trier les rappels de produits selon plusieurs critères, tels que la catégorie de produit, la date de rappel, et le niveau de dangerosité. Cela aide les utilisateurs à organiser et à accéder facilement aux rappels qui les concernent le plus.

2.1.4 Configuration de l'application

La page de configuration offre aux utilisateurs la possibilité de personnaliser le comportement de l'application selon leurs préférences. Les utilisateurs peuvent choisir de limiter le nombre de résultats renvoyés par les requêtes pour éviter une surcharge d'informations. Ils

peuvent également fixer certains critères de recherche par défaut, comme n'afficher que les rappels d'une certaine catégorie de produits ou selon les dates de commercialisation. La page de configuration permet ainsi d'adapter l'application aux besoins spécifiques de chaque utilisateur, améliorant ainsi l'expérience utilisateur globale.

2.2 Navigation

La navigation dans l'application est organisée de manière intuitive et logique pour garantir une expérience utilisateur fluide. L'application est structurée autour de trois sections principales : la recherche des rappels, la gestion des favoris, et la configuration. La navigation entre ces sections se fait via un menu d'action accessible depuis toutes les pages de l'application.

2.2.1 Page de recherche des rappels

Accessible dès l'ouverture de l'application, elle permet de lancer des recherches de produits rappelés et d'afficher les résultats sous forme de cartes. Chaque carte est cliquable pour accéder aux détails du rappel.

2.2.2 Page de détails du rappel

Accédée en cliquant sur une carte de rappel dans la page de recherche ou la page des favoris. Elle affiche toutes les informations détaillées sur le rappel sélectionné et permet d'ajouter ou de retirer le produit des favoris.

2.2.3 Page des favoris

Contient la liste des produits ajoutés aux favoris. Les utilisateurs peuvent gérer leurs favoris, accéder aux détails des rappels, et utiliser les options de filtrage pour organiser les produits rappelés.

2.2.4 Page de configuration

Permet aux utilisateurs de personnaliser l'application selon leurs besoins, en configurant les paramètres de recherche et les limites de résultats.

3 Conception et implémentation

3.1 Architecture

L'architecture générale de notre application repose sur une structure modulaire et flexible, visant à séparer les différentes responsabilités en composants distincts. Cette approche facilite la maintenance et l'évolution future de l'application. Les principaux composants de l'application sont les suivants :

Activités et Fragments : L'application utilise une activité principale qui agit comme un conteneur pour divers fragments. Les fragments sont utilisés pour représenter les différentes sections de l'application, telles que la recherche des rappels, la gestion des favoris, et la configuration. Chaque fragment gère une partie spécifique de l'interface utilisateur et de la logique associée.

Services : Les services sont utilisés pour les opérations en arrière-plan, comme les appels API pour récupérer les données de rappel depuis l'API gouvernementale. Ils assurent que les opérations longues n'interfèrent pas avec l'expérience utilisateur en bloquant l'interface.

ViewModel et LiveData : Pour une gestion efficace des données et pour séparer la logique d'affaires de l'interface utilisateur, nous avons utilisé le modèle ViewModel et LiveData de l'architecture Android. Cela permet de persister les données même lors des changements de configuration, comme la rotation de l'écran.

3.2 Base de données

Nous avons utilisé Room pour gérer la base de données locale des favoris. Room est une bibliothèque de persistance qui fournit une abstraction au-dessus de SQLite, simplifiant ainsi l'accès aux bases de données tout en garantissant des opérations sécurisées et performantes. Voici comment nous avons implémenté la gestion des favoris :

Entité : Nous avons défini une entité "Favoris" qui représente les produits ajoutés aux favoris par l'utilisateur. Chaque entité correspond à une table dans la base de données SQLite.

DAO (Data Access Object) : Nous avons créé un DAO pour définir les méthodes d'accès aux données de la base de données. Cela inclut des opérations telles que l'insertion, la suppression et la requête des favoris.

Database : Nous avons implémenté la classe de base de données qui étend RoomDatabase. Cette classe utilise le DAO pour effectuer les opérations sur la base de données.

3.3 Appels API

Pour interagir avec l'API de Rappel Conso, nous avons utilisé Retrofit, une bibliothèque puissante pour effectuer des requêtes HTTP en Java. Retrofit facilite la gestion des appels réseau et la conversion des réponses JSON en objets Java. Voici les principales étapes de notre implémentation :

Interface API : Nous avons défini une interface décrivant les différentes requêtes HTTP que notre application peut effectuer, telles que la récupération de la liste des rappels récents et la recherche de rappels spécifiques.

Moshi : Nous avons utilisé Moshi pour la sérialisation et la désérialisation des données JSON. Moshi est une bibliothèque moderne qui fonctionne bien avec Retrofit pour convertir les réponses JSON en objets Java.

3.4 Interface utilisateur

L'interface utilisateur de notre application est conçue pour être intuitive et conviviale. Nous avons utilisé les composants modernes d'Android pour créer une interface attrayante et fonctionnelle. Voici un aperçu des principales interfaces utilisateur :

Page de recherche des rappels : Cette page comporte une barre de recherche en haut, suivie d'une liste de cartes représentant les rappels de produits. Chaque carte affiche une image du produit, son nom et une information supplémentaire. Un menu contextuel sur chaque carte permet d'ajouter le produit aux favoris.

Page de détails du rappel : Lorsqu'un utilisateur clique sur une carte de rappel, il est redirigé vers une page détaillant toutes les informations relatives au rappel. Cette page inclut des descriptions, differantes information et les actions recommandées. Un bouton permet d'ajouter ou de retirer le produit des favoris.

Page des favoris : Cette page affiche tous les produits ajoutés aux favoris sous forme de cartes. Chaque carte inclut des informations détaillées sur le produit et un menu contextuel pour supprimer le produit des favoris ou copier le lien de l'offre.

Page de configuration : Cette page permet aux utilisateurs de personnaliser les paramètres de l'application, comme la limitation du nombre de résultats de recherche et la définition des critères de recherche par défaut.

4 Difficultés rencontrées et solutions

4.1 Problèmes techniques

4.1.1 Problèmes rencontrés par Genti Anthony

Intégration de l'API avec Retrofit et Moshi

Problème : Initialement, la récupération des données de l'API de Rappel Conso via Retrofit a posé des problèmes, notamment des erreurs de sérialisation des réponses JSON avec Moshi. Les réponses contenaient des champs inattendus ou absents, causant des erreurs lors de la conversion en objets Java.

Solution : Pour surmonter ce défi, j'ai d'abord exploré la documentation complète de l'API de Rappel Conso pour comprendre les structures de réponse possibles. J'ai ensuite ajusté les classes de données et utilisé des annotations @Json sur les champs pour gérer les noms de champs JSON variables. Enfin, j'ai ajouté des blocs try-catch autour des appels API pour gérer les erreurs de sérialisation et afficher des messages d'erreur utiles à l'utilisateur.

Gestion des fragments et navigation

Problème : La configuration des fragments et de la navigation entre eux était initialement complexe, surtout pour maintenir un état cohérent lors des changements de configuration (comme la rotation de l'écran).

Solution : J'ai utilisé Android ViewModel et LiveData pour séparer la logique de la gestion des données de l'interface utilisateur. Cela a permis de persister l'état des fragments lors des changements de configuration. De plus, j'ai adopté la navigation par composants d'Android pour simplifier et centraliser la gestion de la navigation entre les fragments.

Base de données Room

Problème : La mise en place de la base de données Room a été difficile en raison des erreurs de migration de base de données lorsque de nouvelles colonnes ou tables étaient ajoutées.

Solution : J'ai résolu ce problème en définissant clairement les versions de la base de données et en écrivant des scripts de migration pour chaque changement de structure. Cela a assuré que les données de l'utilisateur restaient intactes lors des mises à jour de l'application.

4.1.2 Problèmes rencontrés par Boudount Youssef

Gestion des favoris et filtrage avancé

Problème : La gestion des favoris nécessitait de mettre en place une structure flexible pour ajouter, supprimer et filtrer les favoris selon plusieurs critères. Le filtrage en particulier posait des défis de performance lorsque le nombre de favoris augmentait.

Solution : J'ai optimisé la gestion des favoris avec Room pour effectuer le filtrage.

4.2 Collaboration

4.2.1 Problèmes de collaboration communs

Coordination et gestion des versions

Problème : La coordination du travail entre les deux membres du binôme a parfois été difficile, notamment pour éviter les conflits de fusion lors des commits simultanés sur des parties interdépendantes du projet.

Solution : Nous avons adopté une stratégie de développement basée sur des branches, où chaque fonctionnalité majeure était développée dans une branche distincte. Les branches étaient fusionnées dans la branche principale après révision et approbation mutuelle.

Communication et gestion du temps

Problème : La gestion du temps et la communication ont posé des défis, surtout lorsque des réunions étaient nécessaires pour discuter des problèmes complexes ou des décisions de conception.

Solution : Nous avons planifié des réunions régulières en utilisant Discord pour discuter des progrès, des problèmes rencontrés et des prochaines étapes.

Partage des connaissances et apprentissage mutuel

Problème : Étant donné que chaque membre du binôme se concentrait sur des aspects différents du projet, il était crucial de partager les connaissances acquises et de s'assurer que chaque membre comprenait les autres parties de l'application.

Solution : Nous avons partagé nos connaissances et chaque membre expliquait les composants sur lesquels il avait travaillé. Cela a non seulement facilité la compréhension mutuelle, mais a également permis d'identifier et de résoudre des problèmes potentiels plus rapidement. Nous avons aussi documenté nos solutions et nos choix de conception avec des commentaires dans le code.

5 Conclusion

5.1 Bilan du projet

Le projet de développement de l'application de suivi des rappels de produits a été une expérience enrichissante et formatrice à plusieurs égards. En respectant les exigences fixées par le sujet du projet, nous avons pu concevoir une application mobile fonctionnelle et intuitive, capable de gérer efficacement les alertes de rappels de produits.

5.2 Points forts

Fonctionnalités Complètes et Fonctionnelles

Nous avons réussi à implémenter toutes les fonctionnalités demandées, y compris la recherche des rappels de produits, la lecture détaillée des rappels, la gestion des favoris, et la configuration de l'application. Chaque fonctionnalité fonctionne de manière fluide, répondant aux attentes des utilisateurs cibles.

Architecture Modulaire

L'architecture de l'application, basée sur des activités, des fragments, des services, et l'utilisation de ViewModel et LiveData, nous a permis de structurer le code de manière modulaire. Cette approche facilite la maintenance et l'évolutivité de l'application, en plus d'assurer une séparation claire des responsabilités entre les composants.

Intégration Efficace des Technologies

L'utilisation de Retrofit pour les appels API, combinée avec Moshi pour la sérialisation JSON, a permis une interaction efficace avec l'API de Rappel Conso. De plus, l'implémentation de Room pour la gestion de la base de données des favoris a garanti des opérations de persistance performantes et sécurisées.

Interface Utilisateur Intuitive

L'interface utilisateur a été conçue pour être intuitive et conviviale, avec une navigation fluide entre les différentes sections de l'application. L'ajout de fonctionnalités telles que le mode sombre et les notifications a amélioré l'expérience utilisateur globale.

Gestion de Projet Efficace

L'utilisation de Git pour le versionnement du code, ainsi que des outils de communication et de planification, a permis une collaboration efficace entre les membres du binôme. Les commits réguliers, les pull requests et les réunions de suivi ont contribué à une gestion de projet structurée et organisée.

5.3 Points à améliorer

Optimisation des Performances

Bien que l'application fonctionne de manière satisfaisante, il existe des marges d'amélioration en termes de performances. Par exemple, la gestion des favoris avec des filtres avancés pourrait être encore optimisée pour réduire les temps de chargement et améliorer la réactivité.

Gestion des Erreurs

La gestion des erreurs, notamment les erreurs réseau et les erreurs de sérialisation JSON, pourrait être améliorée pour offrir une meilleure expérience utilisateur. Une approche plus robuste de gestion des exceptions et des messages d'erreur plus détaillés permettraient aux utilisateurs de comprendre et de résoudre les problèmes plus facilement.

Tests et Validation

Bien que l'application ait été testée de manière extensive, une couverture de tests automatisés plus complète, incluant des tests unitaires et des tests d'intégration, garantirait une stabilité accrue et faciliterait les futures évolutions de l'application.

5.4 Apprentissages

5.4.1 Apprentissages techniques

Maîtrise des Technologies Android

Le projet nous a permis de renforcer nos compétences en développement Android, en particulier l'utilisation de technologies telles que Retrofit, Moshi, Room, et les composants d'architecture Android (ViewModel, LiveData). Nous avons acquis une compréhension approfondie de la gestion des fragments, des services, et des interactions réseau dans une application mobile.

Gestion de Base de Données

L'implémentation de Room pour la gestion de la base de données des favoris nous a appris à concevoir des schémas de base de données efficaces et à gérer les migrations de base de données de manière sécurisée.

Sérialisation et Désérialisation JSON

L'utilisation de Moshi pour la sérialisation et la désérialisation des données JSON nous a permis de comprendre les nuances de la gestion des formats de données complexes et de la conversion entre les structures JSON et les objets Java.

5.4.2 Apprentissages en gestion de projet

Planification et Suivi de Projet

La gestion du projet nous a enseigné l'importance de la planification rigoureuse, de la définition des jalons, et du suivi constant de l'avancement. L'utilisation de GitLab pour la gestion des versions et la collaboration a été cruciale pour assurer la transparence et l'organisation du travail.

Résolution de Conflits

La gestion des conflits de fusion et la résolution des divergences de code nous ont appris à adopter des pratiques de développement collaboratif, telles que les branches de fonctionnalité et les pull requests. Cela a permis de maintenir un codebase propre et cohérent.

5.5 Conclusion

En conclusion, ce projet de fin de semestre a été une expérience précieuse, tant sur le plan technique que sur le plan de la gestion de projet. Nous avons non seulement réussi à développer une application fonctionnelle et utile, mais nous avons également acquis des compétences clés qui nous seront bénéfiques dans nos futures carrières de développeurs. Les défis rencontrés et surmontés ont enrichi notre compréhension du développement d'applications mobiles et de la collaboration en équipe, faisant de ce projet un jalon significatif dans notre parcours académique et professionnel.

6 Annexes

6.1 Captures d'écran

6.1.1 Page d'accueil et recherche des rappels de produits

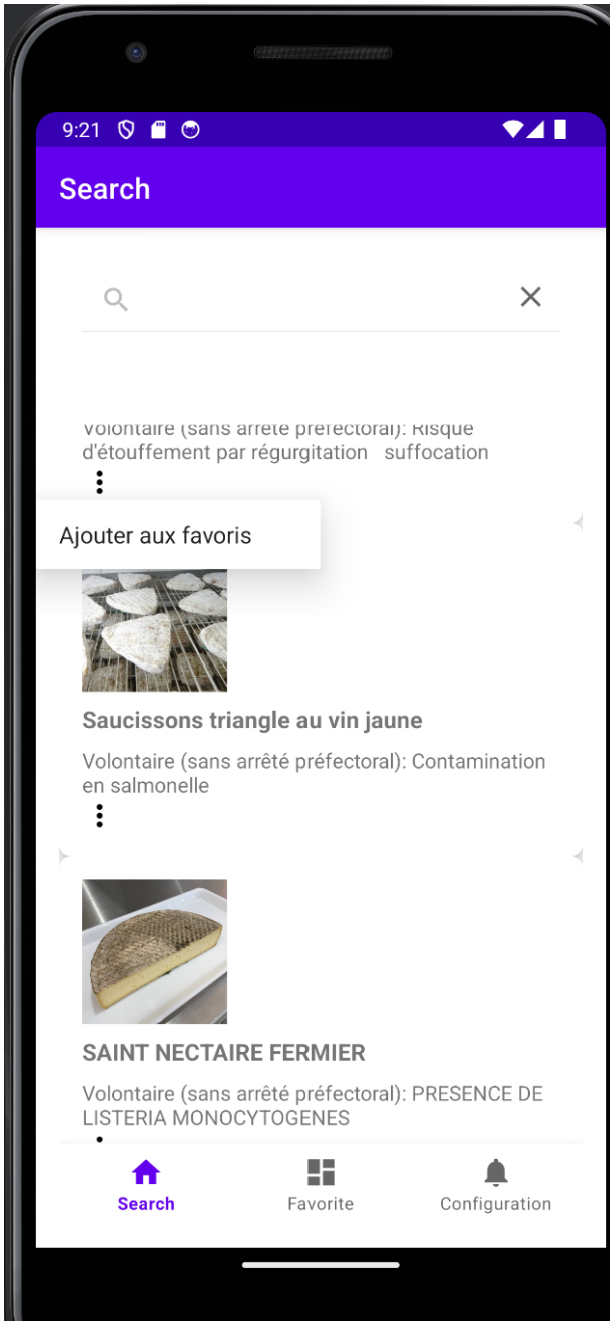

Figure 1. Page d'accueil avec la barre de recherche. Les utilisateurs peuvent saisir des mots-clés pour rechercher des rappels de produits spécifiques.

6.1.2 Résultats de la recherche

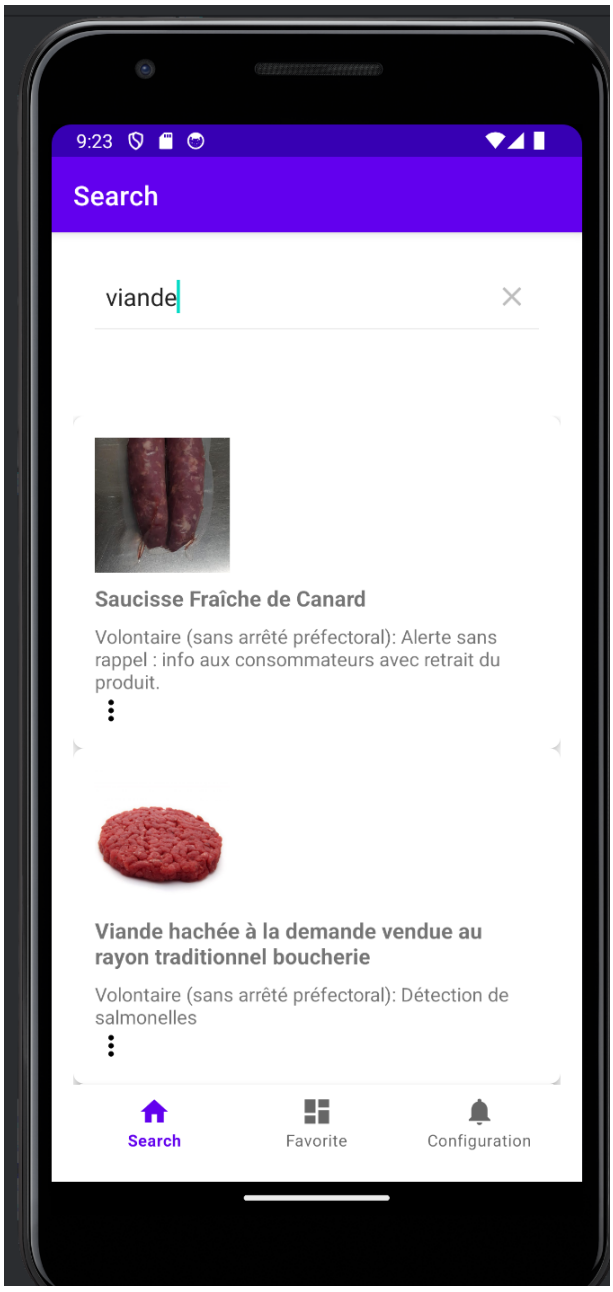

Figure 2. Liste des résultats de recherche affichée sous forme de cartes. Chaque carte contient une image du produit rappelé, son nom, et la date de rappel. Un menu contextuel permet d'ajouter le produit aux favoris.

6.1.3 Détails d'un rappel de produit

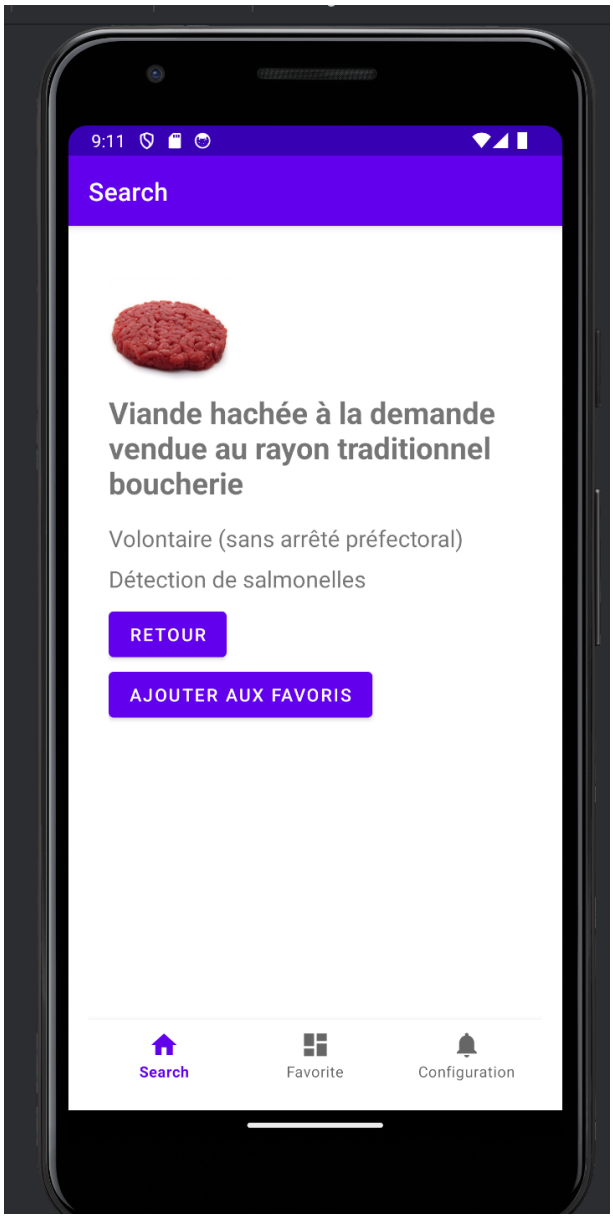

Figure 3. Page de détails d'un rappel de produit. Cette page fournit des informations complètes sur le produit rappelé, y compris les raisons du rappel, les actions recommandées, et des images supplémentaires. Un bouton permet d'ajouter le produit aux favoris.

6.1.4 Gestion des favoris

Figure 4. Page de gestion des favoris. Les produits ajoutés aux favoris sont affichés sous forme de cartes. Les utilisateurs peuvent supprimer des produits de la liste des favoris ou copier le lien de l'offre. Un menu de filtrage permet de trier les produits selon divers critères.

6.1.5 Détails d'un produit favori

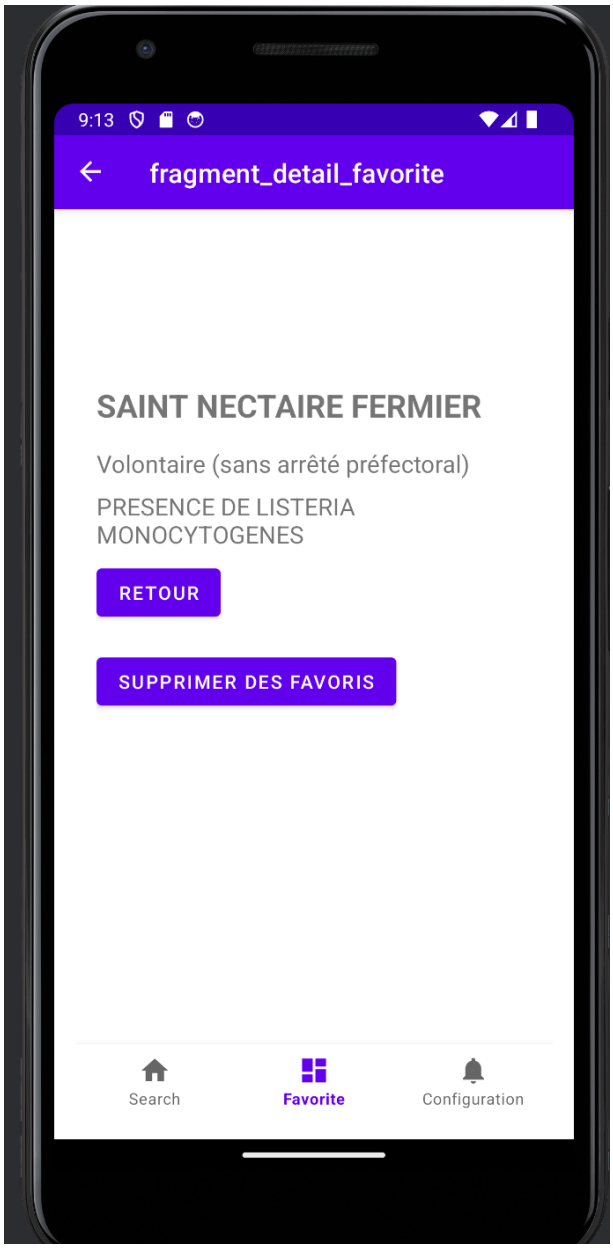

Figure 5. Page de détails d'un produit favori, similaire à la page de détails de la recherche. Elle inclut un bouton pour retirer le produit des favoris.

6.1.6 Page de configuration

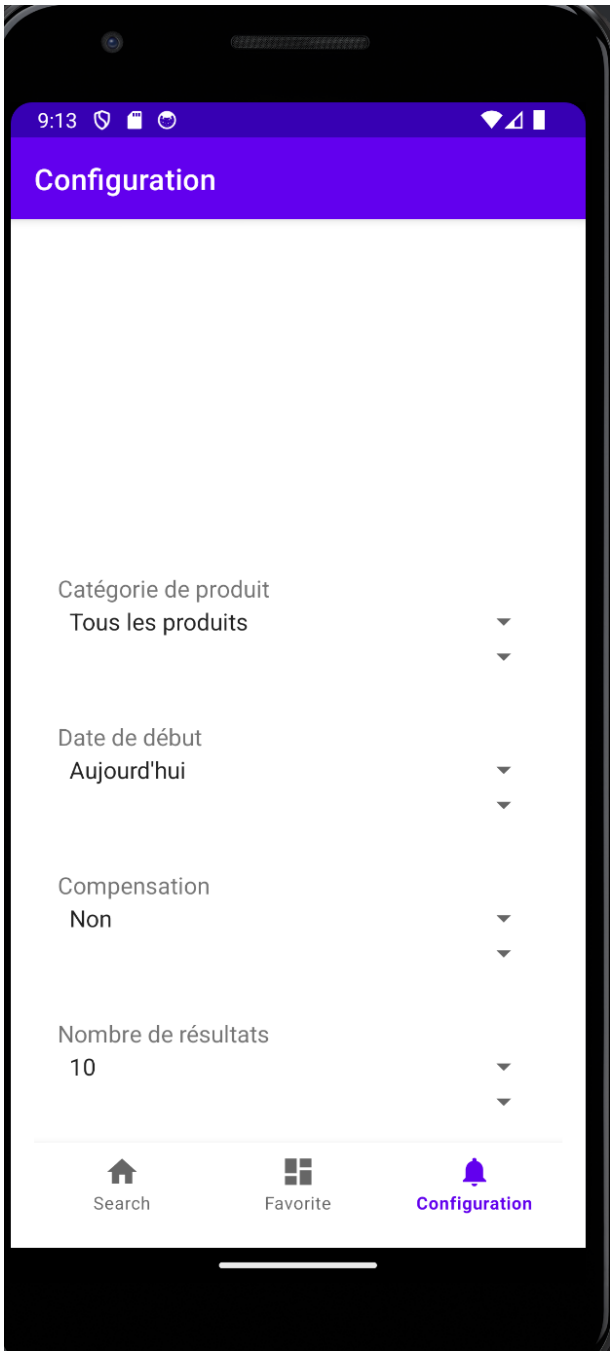

Figure 6. Page de configuration de l'application. Les utilisateurs peuvent personnaliser les critères de recherche, limiter le nombre de résultats affichés, et activer ou désactiver le mode sombre.