

Projet : Application de suivi des rappels de produits

| | |
|------------------------------|----|
| 1/ Page Recherche..... | 2 |
| 2/ Page rappel détaillé..... | 5 |
| 3/ Page favoris..... | 7 |
| 4/ Page Configuration..... | 10 |
| 5/ Conclusion..... | 15 |

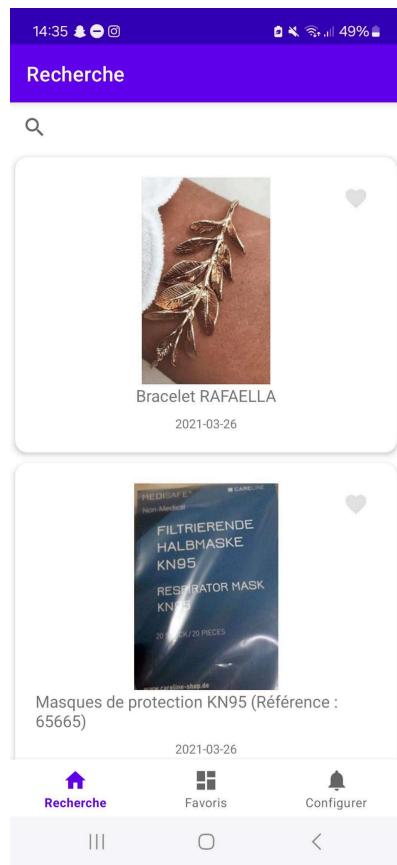
1. Page Recherche

La page (Home) recherche permet à l'utilisateur de rechercher et de visualiser rapidement des rappels spécifiques à partir de l'API de RappelConso. Les principales fonctionnalités incluent :

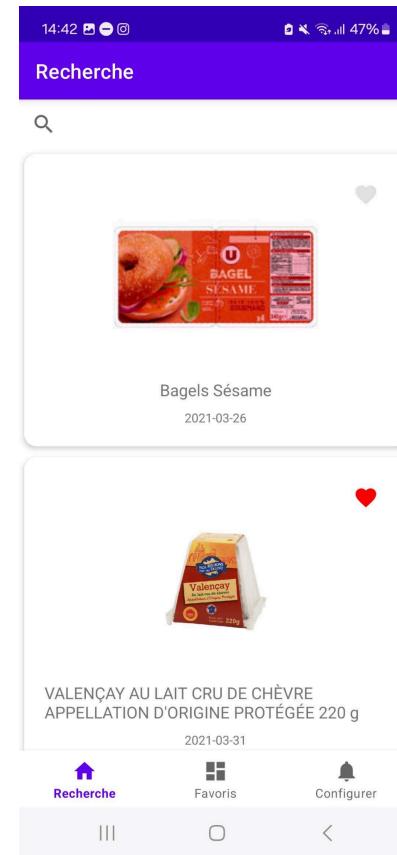
1. Affichage de la liste des rappels : La page affiche une liste de rappels obtenus à partir de l'API. Chaque élément de la liste est représenté par un card_layout, affichant les informations principales du produit telles que son nom, sa date et son image.
2. Affichage des détails des rappels : L'utilisateur peut cliquer sur un rappel pour afficher tous ses détails. Cela le redirige vers la page SecondFragment.
3. Gestion des favoris : L'utilisateur peut ajouter ou retirer un rappel de ses favoris en cliquant sur un bouton favori associé à chaque élément de la liste. Le bouton est symbolisé par un cœur vide si le produit n'est pas dans les favoris, et un cœur rouge s'il y est déjà.
4. Recherche de produits : Une barre de recherche est disponible en haut de l'écran pour permettre à l'utilisateur de rechercher un produit spécifique dans la liste affichée actuellement, en utilisant son nom.

L'implémentation des fonctionnalités de la page Recherche se fait à travers trois fichiers principaux :

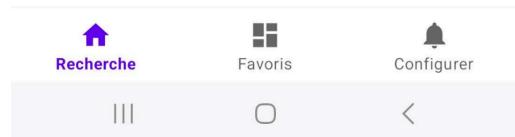
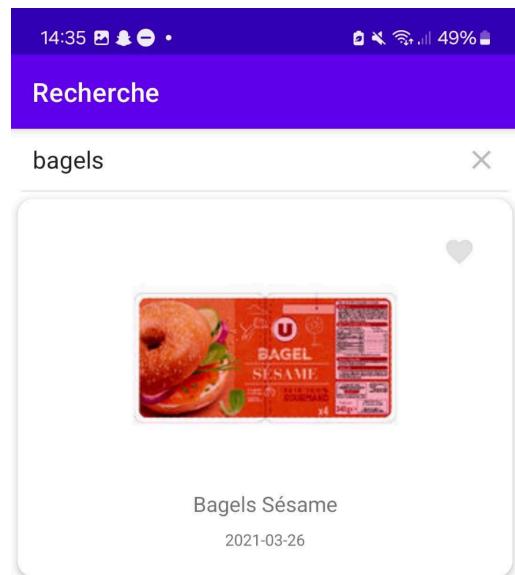
- HomeFragment: Ce fragment gère l'affichage de la liste des rappels et la barre de recherche. Il utilise l'API de RappelConso pour récupérer les données de rappel et les afficher dans un RecyclerView. Il permet également de gérer les interactions utilisateur telles que le clic sur un rappel pour afficher ses détails et l'ajout ou le retrait de rappels des favoris.
- HomeRecyclerAdapter : Cet adaptateur gère l'affichage des données de rappel dans le RecyclerView du fragment HomeFragment. Il est responsable de la création des vues pour chaque élément de la liste et de la gestion des événements tels que le clic sur le bouton favori.
- HomeViewModel : Ce ViewModel gère la récupération des données à partir de l'API et les expose au fragment Home. Il utilise Retrofit pour effectuer des appels à l'API et récupérer les données de rappel. Les préférences de l'utilisateur, telles que les filtres de recherche, sont prises en compte via SharedPreferences lors de la récupération des données. Les rappels récupérés sont stockés dans une liste LiveData, permettant ainsi de les observer et de mettre à jour l'affichage en temps réel.



Page de Recherche



Page de Recherche avec un favoris



Test de la bar de recherche avec le mot bagels

2. Page rappel détaillé

Le fragment SecondFragment est conçu pour afficher les détails d'un rappel sélectionné par l'utilisateur, ainsi que pour permettre la gestion des favoris. Voici un aperçu de ses fonctionnalités :

1. Affichage des détails du rappel : Le fragment affiche les détails d'un rappel sélectionné, notamment son nom, sa date, sa catégorie, sa sous-catégorie, sa marque, son risque, son motif et sa compensation.
2. Récupération des détails du rappel : Les détails du rappel sont récupérés à partir d'un Bundle transmis depuis le fragment précédent (généralement HomeFragment ou DashboardFragment).
3. Gestion des favoris : L'utilisateur peut ajouter ou retirer le rappel de ses favoris en cliquant sur le bouton "Favori". Le bouton se met à jour en conséquence, affichant le texte approprié pour ajouter ou retirer le rappel des favoris.

L'implémentation des fonctionnalités de la page Second se fait à travers deux principaux composants :

- SecondFragment : Ce fragment est chargé d'afficher les détails d'un rappel sélectionné. Il récupère les détails du rappel à partir du Bundle transmis depuis le fragment précédent. Il affiche ensuite ces détails dans des TextView appropriés. Le fragment permet également à l'utilisateur d'ajouter ou de retirer le rappel de ses favoris en cliquant sur un bouton "Favori".
- SecondViewModel : Ce ViewModel gère la logique métier associée à l'affichage des détails du rappel dans SecondFragment. Il expose un LiveData contenant le rappel sélectionné pour observation depuis le fragment. De plus, le ViewModel est responsable de l'ajout et de la suppression du rappel sélectionné dans la base de données, selon les actions de l'utilisateur.

Affichage du SecondFragment lorsque l'on clique sur l'un des rappel :



avec le bouton ajouter aux favoris....



et avec le bouton retirer des favoris

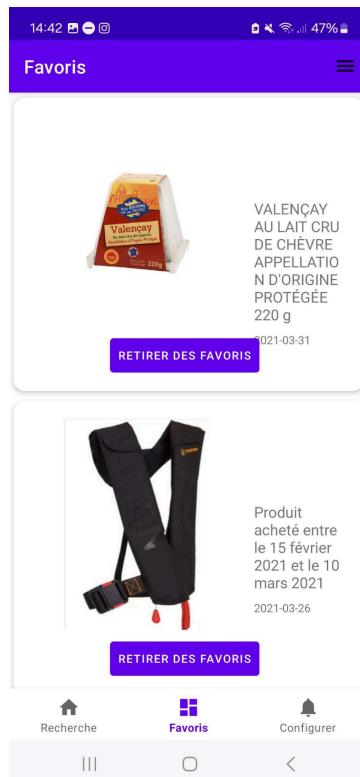
3. Page favoris

La page Dashboard (Favoris) affiche une liste de rappels favoris provenant de la base de données Room. Les fonctionnalités de cette page comprennent :

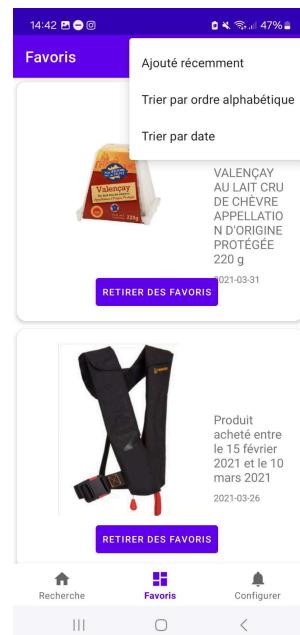
1. Affichage des rappels favoris : Les rappels sont affichés dans une liste déroulante à l'aide d'un RecyclerView.
2. Navigation vers SecondFragment : En cliquant sur un rappel, l'utilisateur est redirigé vers la page SecondFragment, où des détails plus spécifiques sur le rappel sélectionné sont affichés.
3. Suppression d'un rappel favori : En appuyant sur le bouton "Retirer des Favoris" associé à chaque rappel, l'utilisateur peut supprimer ce rappel de la liste des favoris.
4. Tri des rappels : L'utilisateur peut trier les rappels par ordre alphabétique ou par la date du rappel ainsi que "ajouté récemment". Le tri par ordre alphabétique est effectué en fonction du nom du produit, le tri par date est basé sur la date de rappel associée à chaque élément, tant dis que "ajouté récemment" se base sur l'id du rappel, il affiche le dernier id en premier et le premier id en dernier.

L'implémentation de ces fonctionnalités est réalisée à l'aide de plusieurs composants :

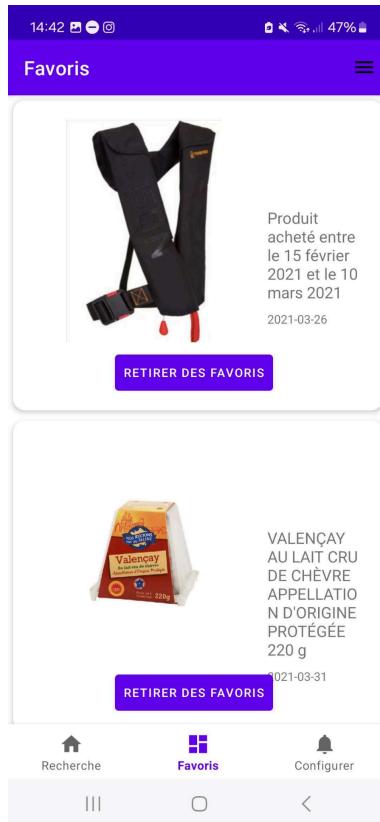
- DashboardFragment: Ce fragment est responsable de l'affichage de la liste des rappels favoris et de la gestion des interactions de l'utilisateur, telles que le tri et la navigation vers SecondFragment.
- DashboardRecyclerAdapter: Cet adaptateur est utilisé pour lier les données des rappels à la vue RecyclerView. Il gère également les interactions utilisateur telles que les clics sur les éléments de la liste et les clics sur les boutons "Retirer des Favoris".
- DashboardViewModel: Ce ViewModel gère la récupération des données des rappels depuis le repository et les expose à DashboardFragment. Il maintient également l'état des données même en cas de rotation de l'appareil.
- RappelRepository: Cette classe agit comme une couche d'abstraction entre la source de données (la base de données Room) et le ViewModel. Elle fournit des méthodes pour interagir avec la base de données, telles que la récupération de tous les rappels favoris.



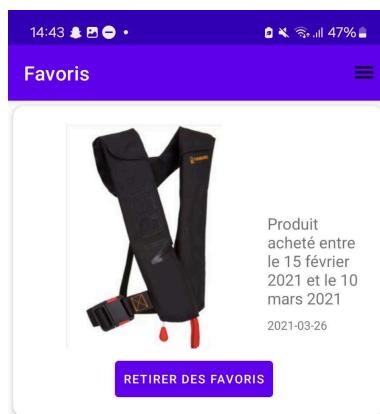
Page des favoris



ainsi que son menu de tri



après le tri par ordre alphabétique



après avoir retiré un élément des favoris

4. Page Configuration

La page de configuration (NotificationFragment) vous permet de définir différents paramètres pour filtrer les résultats des rappels. Cinq filtres de recherche sont possibles les voici :

1. Le nombre maximum de résultats. Cliquer dessus permet de choisir le nombre de résultats à afficher.
2. Le type de catégories de résultats, soit : Alimentaire, Vêtements ou Automobile.
3. Le type de modalité de compensation, soit : Remboursement ou Echange.
4. Les risques encourus, soit : Blessures externes ou dommage interne.
5. Le distributeur, soit Décathlon, Aldi ou Leclerc.

Il est important de noter qu'il est possible d'ajouter ces filtres de recherche (exemple: catégorie = Alimentaire et distributeur = Leclerc va afficher uniquement les rappels alimentaire du distributeur Leclerc).

L'implémentation de ces fonctionnalités est réalisée à l'aide de plusieurs composants :

- NotificationFragment : Ce fragment affiche les préférences de configuration à l'utilisateur, telles que le nombre maximum de résultats, la catégorie de produit, la modalité de compensation, le risque encourus, et le distributeur. Il utilise un ViewModel (NotificationsViewModel) pour récupérer les données à afficher dans les préférences.
- NotificationsViewModel : Ce ViewModel est responsable de fournir les données nécessaires à l'affichage des préférences de configuration dans NotificationFragment. Il expose un LiveData contenant le texte à afficher dans les préférences.
- arrays.xml : Ce fichier contient les tableaux de chaînes utilisés pour les différentes listes déroulantes dans les préférences de configuration, tels que les catégories de produits, les modalités de compensation, les risques encourus et les distributeurs.

Ensemble, ces fichiers forment l'interface utilisateur et la logique associée à la page de configuration, permettant à l'utilisateur de personnaliser les paramètres selon ses préférences.



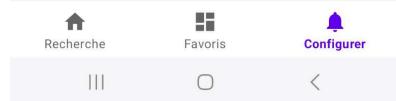
Nombre maximum de résultats
5

Catégorie
Toutes catégories

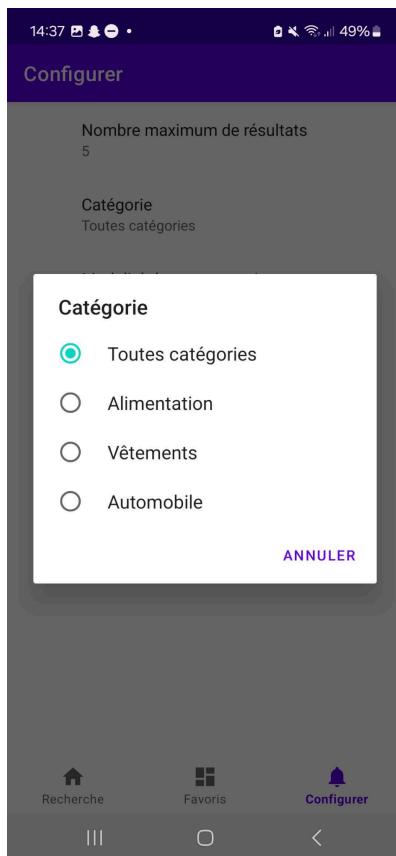
Modalité de compensation
Tout type

Risque encourus
Tout type

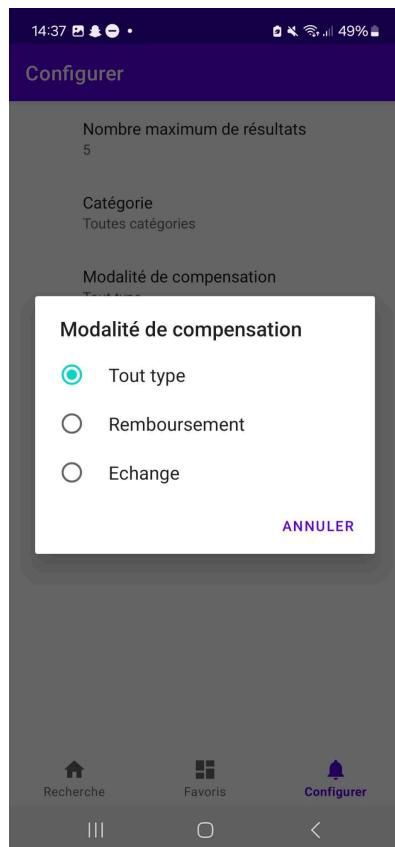
Distributeur
Tout distributeur



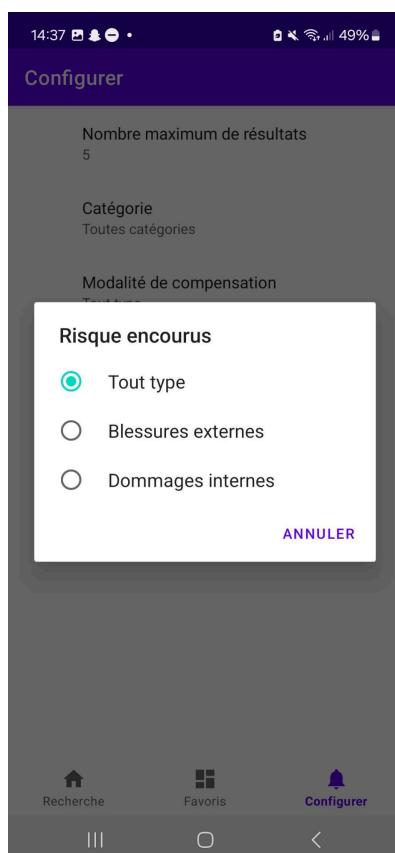
Avec les options par défaut



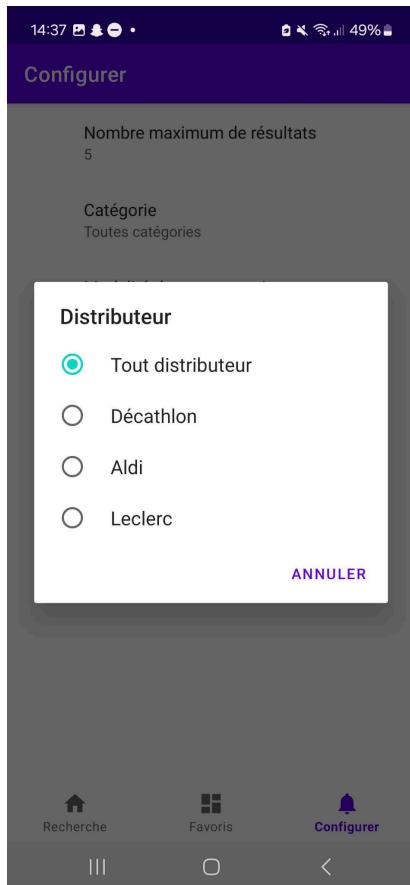
Choix des options de catégorie



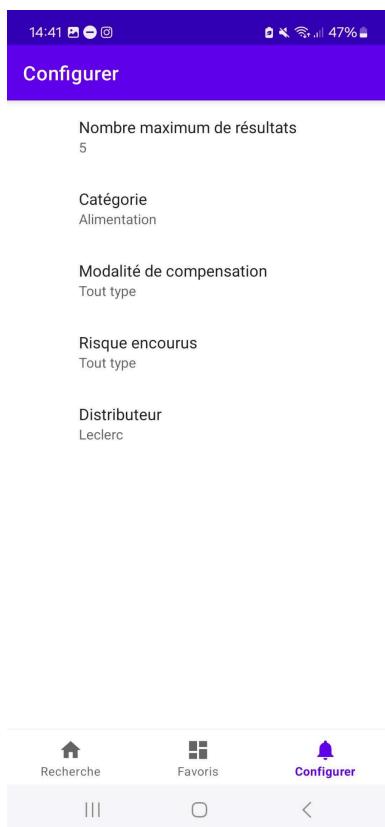
Choix des options de compensation



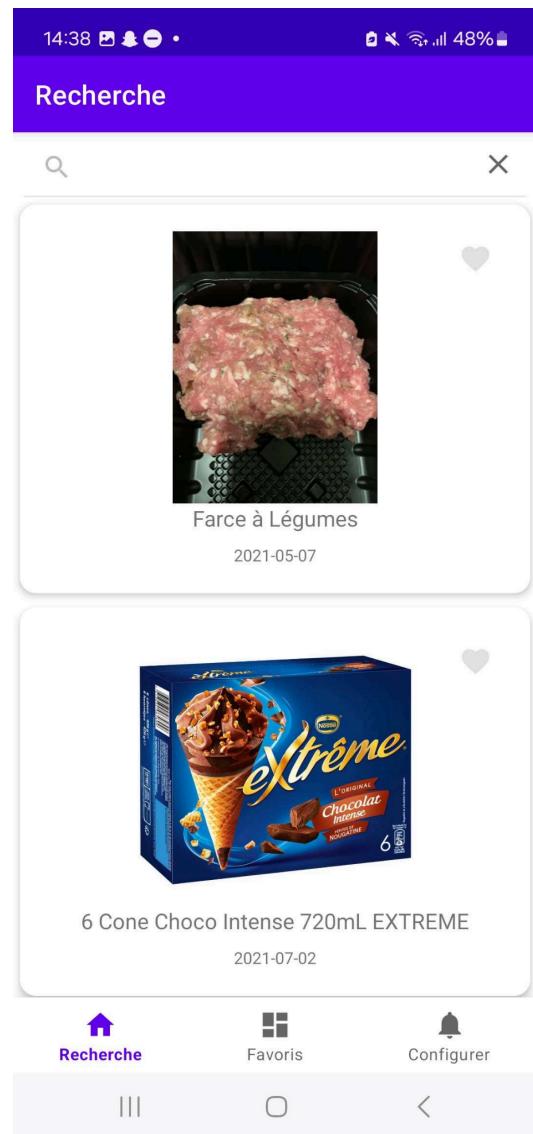
Choix des options de risque



Choix des options de distributeur



Avec des options personnalisées



Résultat de la recherche avec les options personnalisées

5. Appel à l'API

- Appel à l'API :

L'application utilise une API externe pour récupérer les données de rappel de produits. L'API est appelée à l'aide de la bibliothèque Retrofit, qui facilite la communication entre l'application et l'API RESTful.

- Configuration de l'API :

Pour configurer l'API, l'application utilise une instance de la classe api qui implémente l'interface apilmplement. Cette instance est créée à l'aide d'un objet Retrofit.Builder qui définit l'URL de base de l'API et le convertisseur Gson pour parser les réponses JSON.

- Requêtes API :

L'application effectue des requêtes GET à l'API pour récupérer les données de rappel de produits. La méthode getRappels() de l'interface apilmplement est utilisée pour effectuer ces requêtes. Cette méthode prend deux paramètres :

limit (entier) - Le nombre maximum de résultats à renvoyer.
where (chaîne de caractères) - La clause WHERE de la requête, qui est utilisée pour filtrer les résultats en fonction des critères sélectionnés par l'utilisateur.

- Traitement des réponses :

Lorsqu'une réponse est reçue de l'API, l'application extrait les données pertinentes des enregistrements et met à jour la liste des rappels de produits (rappelList) dans le HomeViewModel. La liste mise à jour est ensuite affichée dans le RecyclerView de la page d'accueil.

- Construction de la clause WHERE :

La clause WHERE de la requête API est dynamiquement construite en fonction des préférences de l'utilisateur. Les préférences sont récupérées à partir des SharedPreferences dans le fichier HomeViewModel.java.

- Les préférences suivantes sont utilisées pour construire la clause WHERE :

max_results (entier) - Le nombre maximum de résultats à renvoyer.
compensation (chaîne de caractères) - La compensation pour le rappel de produit.
categorie (chaîne de caractères) - La catégorie du produit rappelé.
risque (chaîne de caractères) - Le risque associé au rappel de produit.
date (chaîne de caractères) - La date de publication du rappel de produit.
La clause WHERE est construite en concaténant les valeurs des préférences avec les noms de champs appropriés et les opérateurs de comparaison.
Les clauses WHERE pour chaque préférence sont ensuite concaténées à l'aide de l'opérateur logique AND pour former la clause WHERE finale de la requête API.

6. Conclusion

En conclusion, en plus d'avoir énormément appris de ce projet dans le milieu de création d'application mobile, nous avons tous deux pris plaisir à collaborer sur ce projet, avec un partage équitable des tâches entre les deux membres de l'équipe, favorisé par une excellente entente et une communication fluide.

SEGHIR Samy a pris en charge la gestion de l'API, ce qui a permis de récupérer les données des rappels et de les afficher efficacement sur la page de recherche. Son travail sur la liste des rappels et la mise en place de la page de configuration ont été essentiels pour offrir une expérience utilisateur fluide et personnalisée.

D'autre part, RIVES-LEHTINEN Louis a joué un rôle clé dans la création de la base de données et l'implémentation de la fonctionnalité des favoris. Son travail sur la page des favoris, ainsi que la mise en place de tous les systèmes de gestion des favoris, tels que les boutons d'ajout et de retrait, le bouton cœur etc.., a contribué à enrichir l'expérience utilisateur en offrant une manière pratique de sauvegarder et d'accéder rapidement aux rappels préférés.

En outre, Louis a également réalisé le SecondFragment (la page de détails d'un rappel), qui permet d'afficher en détail les informations sur un rappel sélectionné, ainsi que de gérer son ajout ou sa suppression des favoris.

Cependant, en raison de notre départ en Erasmus lors du semestre 5, nous avons rencontré quelques difficultés avec GitHub, expliquant le faible taux de commit sur le projet.

En combinant nos compétences et nos efforts, nous avons réussi à créer une application fonctionnelle, offrant aux utilisateurs la possibilité de rechercher, d'afficher et de gérer facilement une liste de rappels de produits, tout en bénéficiant de fonctionnalités avancées telles que la gestion des favoris.