Dujardin Xavier - Demeure Corentin - Verhaeghe Martin

Projet SmartCity - Développement mobile

Rappel du projet

Le projet choisi consiste en une centralisation du domaine des brocantes / vides-greniers,.. Le but de la partie mobile étant de permettre à des utilisateurs de pouvoir retrouver et participer facilement à des brocantes.

Lien github du projet

https://github.com/etu51424/appBrocante

Rappel des fonctionnalités

- Liste de brocante à partir d'un lieu, d'une délimitation de zone, d'un intervalle entre deux dates : La fonctionnalité existe via la carte interactive et les filtres.
- Description d'une brocante choisie par l'utilisateur : La fonctionnalité existe via les écran de détails en cliquant sur les brocantes dans la liste.
- Liste des brocanteurs d'une brocante : La fonctionnalité existe via l'écran d'affichage des intérêts d'une brocante.
- Liste des objets vendus par un brocanteur : La fonctionnalité existe via l'écran d'affichage des articles d'un brocanteur.
- Carte interactive utilisant le capteur de localisation et magnétomètre : La fonctionnalité existe partiellement, la carte n'est malheureusement pas totalement interactive étant donné qu'il faudra passer par la liste pour afficher les détails d'une brocante.

Fonctionnalités notées comme supplémentaires dans le pré dossier

- Envoi d'un mail de bienvenue lors de la création d'un compte : La fonctionnalité existe (API)
- Envoi d'un mail en cas de mot de passe oublié &
- Envoie de mail en cas de nom d'utilisateur oublié : Les fonctionnalités existent (API) et sont utilisables via le bouton d'aide l'écran Login (Mobile) bien que des pistes d'améliorations sont à prévoir.
- Envoi d'un mail en cas de bannissement d'un utilisateur : La fonctionnalité existe (API)
- Mode sombre/clair : La fonctionnalité n'est pas totalement implémentée

- Multi langage : La fonctionnalité existe, il est possible qu'il y ait des trous de traductions à certains endroits par faute de temps pour le projet.

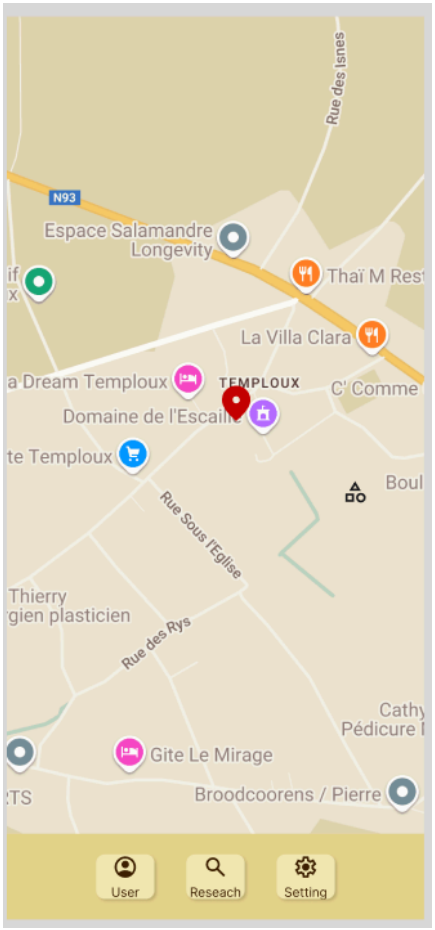
Note sur les fonctionnalités en général

Nous avons remarqué après la remise de l'API que certaines fonctionnalités non définies dans le pré dossier mais présentes sur les maquettes étaient impossible à créer dans les délais imposés (nécessité de modifier / ajouter des parties à l'API) donc nous avons dévié de ces objectif pour fournir une alternative un peu plus accès sur l'interaction avec le système global. Il est donc possible d'insérer, modifier, supprimer des articles et des intérêts.

Maquette

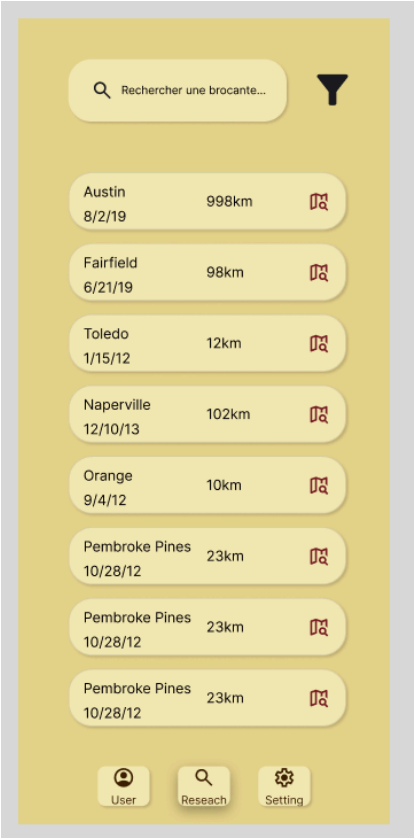
Lien vers le document Figma :
https://www.figma.com/design/zNcjjQjp9AJZrr4keDj80t/Brocante?node-id=0-1&p=f&t=kbj2hF1GOKhzxpSB-0
Si vos accès sont manquants, contactez etu51971@henallux.be (Dujardin Xavier).

Ecran recherche

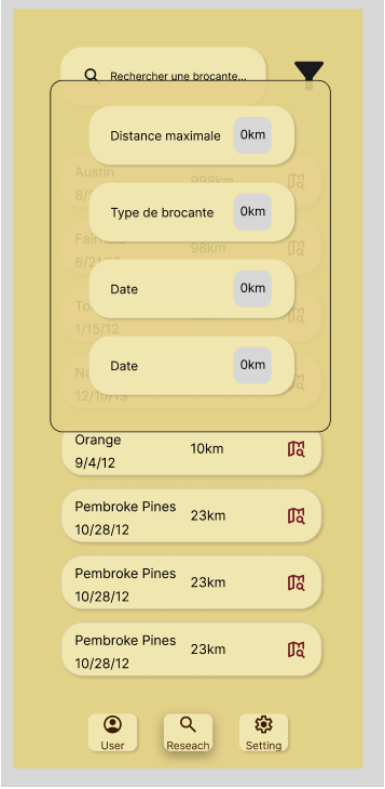


Liste des brocantes

Dujardin Xavier - Demeure Corentin - Verhaeghe Martin

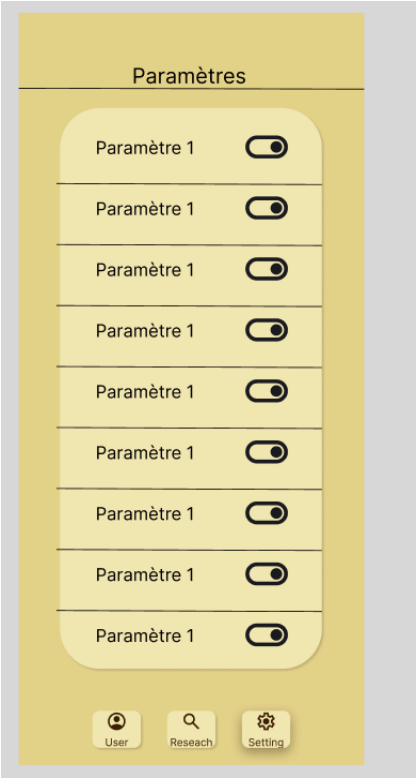


Filtres pour la recherche

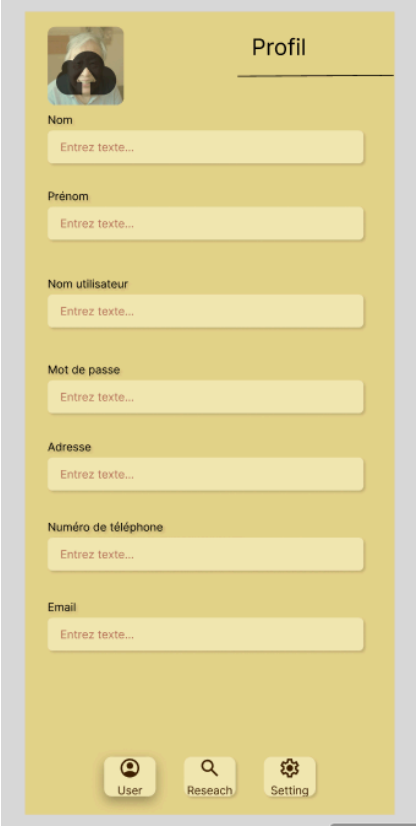


Paramètres

Dujardin Xavier - Demeure Corentin - Verhaeghe Martin

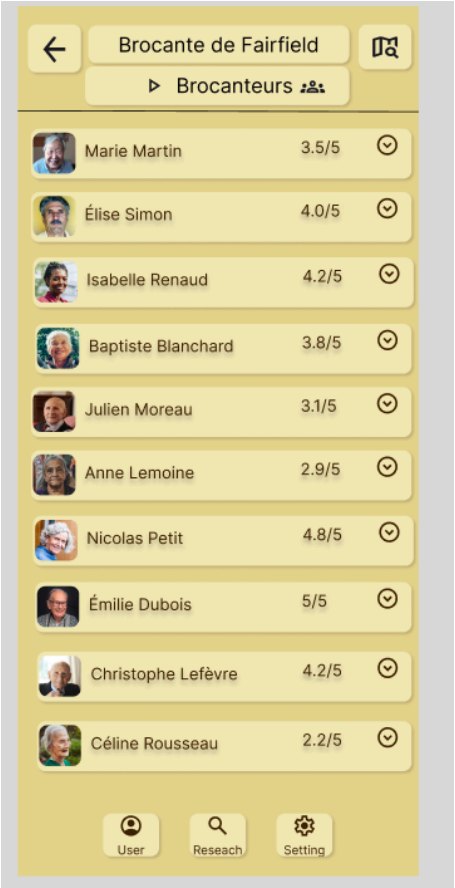


Profil utilisateur

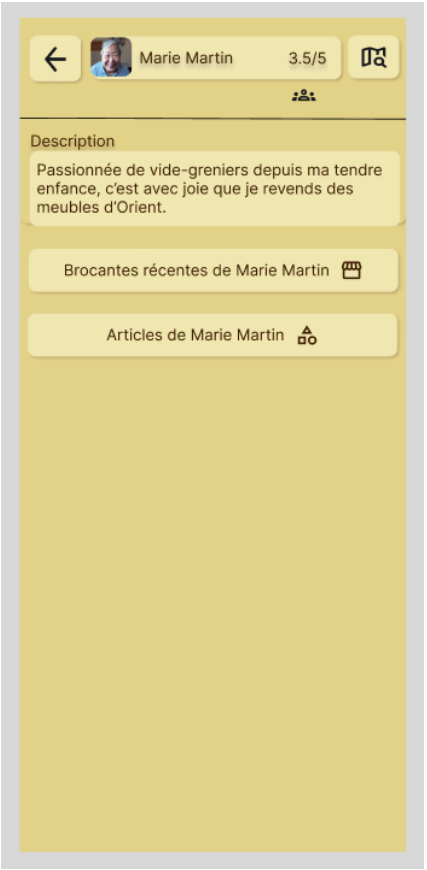


liste des intérets

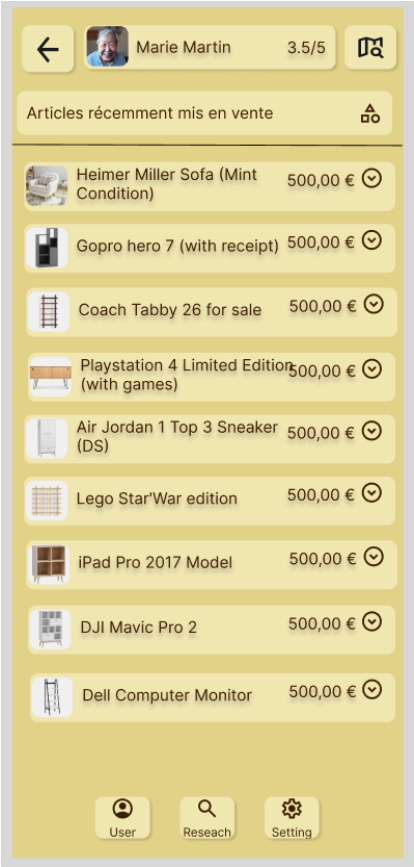
Dujardin Xavier - Demeure Corentin - Verhaeghe Martin



Détails d’un profil



Articles d’un dealer :



Description du code du prototype utilisé

Structure

L’arborescence s’organise comme ceci :
Appbrocante/
├── assets/ : stockage des images
├── fetchAPI/ : fichiers permettant la communication avec l’API
├── screens/ : fichiers contenant le code des écrans
├── store/ : fichiers concernant le stockage des context redux
├── translations/ : fichiers contenant les différentes traductions sous forme de .json
├── utils/ : fichiers contenant du code javascript utile à certaines fonctionnalités
└── App.js

Page d’accueil

Arrivé sur l’application, l’utilisateur doit attendre un temps de chargement des composants et du bundling. Le logo personnalisé de l’application tient compagnie à l’utilisateur sur l’écran de chargement.

Ensuite l'utilisateur atterrit directement sur l'écran « Rechercher », mis par défaut sur la vue carte.

L'application demande au système de l'appareil, qui lui demande à l'utilisateur, si l'application peut accéder au capteur de localisation. L'utilisateur peut refuser ou accepter.

S'il accepte, la carte est affichée autour de la position utilisateur.

Sous la carte, sont en tout temps facile d'accès les 3 écrans principaux : Utilisateur, Rechercher et Paramètres.

Utilisateur

Par défaut, cet onglet permet à l'utilisateur de se connecter. L'utilisateur est invité à entrer son nom d'utilisateur et son mot de passe dans leurs champs respectifs. S'il n'a pas de compte, l'utilisateur peut en créer un.

Il arrive alors sur un nouvel écran. Il peut remplir différents champs. Ceux-ci incluent le prénom, le nom, le nom d'utilisateur, l'adresse et le numéro de téléphone, adresse e-mail et un mot de passe sécurisé. Comme pour tout autre champ, des placeholders guident l'utilisateur sur le type d'info qu'il doit intuitivement introduire.

Une fois les champs complétés, il peut créer un compte.

Une touche retour est également présente, comme sur la totalité des écrans.

S'il avait essayé de se connecter mais avait échoué à cette tâche, l' utilisateur peut signaler qu'il n'a pas réussi à se connecter. Une fois la touche pertinente cliquée, il est redirigé vers un nouvel écran d'aide où il est invité à soumettre un identifiant et un code de récupération.

Rechercher

L'écran Rechercher a deux vues : carte et liste des brocantes.

La vue carte affiche la carte avec la position utilisateur. La vue liste des brocantes liste les brocantes. Un troisième bouton sur la droite permet de rafraîchir la carte.

L'utilisateur peut entrer une adresse et l'appli doit pouvoir lui proposer une brocante ayant une adresse correspondante.

La carte devrait également afficher la localisation des brocantes recherchées par l'utilisateur.

Filtre

Un modal (petite fenêtre amovible) permet à l'utilisateur de choisir plus précisément ue brocante dans un certain intervalle de dates, et/ou dans une certaine distance autour de sa position courante. Les filtres sont appliqués sur la recherche utilisateur.

Sélectionner une brocante

La vue « liste des brocantes » affiche différentes brocantes à l'utilisateur sous formes de petites vues. Les cliquer révèle les informations de la brocante : l'adresse exacte, les dates de début et de fin de cette brocante, sa note moyenne actuelle et le nombre d'avis.

À terme, l'utilisateur doit pouvoir ajouter lui aussi un avis. L'utilisateur peut voir quels autres utilisateurs se sont montrés intéressés par cette brocante, et peut lui-même placer son intérêt en cliquant sur « Placer un intérêt ». Un modal s'ouvre. L'utilisateur peut alors y préciser s'il est brocanteur ainsi que son numéro de participation. Il peut ensuite soumettre ou annuler son intérêt.

L'utilisateur doit pouvoir également voir les emplacements déjà réservés pour cette brocante en cliquant sur « Voir les emplacements ». Ceux-ci devraient s'afficher sous forme de listes.

Paramètre

Le 3ème écran principal. Permet à l'utilisateur de sélectionner la langue d'affichage de l'application (Anglais, Français ou Néerlandais) et le mode de l'application (sombre ou clair). D'autres paramètres sont prévus à terme.

Description des packages

React Native et Navigation

1. @react-native-community/datetimepicker
   Permet d'afficher un sélecteur de date et d'heure natif dans les applications React Native.
2. @react-native-community/slider
   Fournit un composant de slider natif (barre coulissante) pour React Native, utile pour sélectionner une valeur dans une plage.
3. @react-navigation/bottom-tabs
   Module pour créer une navigation par onglets en bas de l'écran (bottom tabs) dans une application React Navigation.
4. @react-navigation/native
   Composant de base de React Navigation pour gérer la navigation dans les applications React Native.
5. @react-navigation/native-stack
   Fournit une navigation en pile (stack navigation) basée sur des animations natives pour une meilleure fluidité.
6. @react-navigation/stack
   Offre une alternative au module native-stack pour gérer la navigation en pile, avec des animations spécifiques.

Gestion d'état et outils

7. @reduxjs/toolkit
   Une bibliothèque officielle pour Redux qui simplifie la gestion de l'état global, avec des outils intégrés pour réduire le code boilerplate.
8. react-redux
   Fournit une intégration entre React et Redux, permettant de connecter vos composants React au store Redux.

Expo et Extensions

9. expo
   Framework pour développer facilement des applications React Native avec des fonctionnalités prêtes à l’emploi (accès à la caméra, GPS, etc.).
10. expo-location
    Module Expo pour accéder aux fonctionnalités de géolocalisation de l'appareil, comme obtenir les coordonnées GPS.
11. expo-status-bar
    Permet de contrôler la barre de statut (status bar) dans une application Expo (apparence, couleur, visibilité).

Internationalisation et installation

12. i18n-js
    Bibliothèque pour gérer la traduction et la localisation des textes dans une application.
13. install
    Outil basique utilisé pour gérer l'installation de dépendances manuellement.

UI et Design

14. react-native
    Le framework principal pour construire des applications mobiles en React, avec des composants natifs pour Android et iOS.
15. react-native-maps
    Fournit des composants pour afficher et interagir avec des cartes (Google Maps ou Apple Maps).
16. react-native-paper
    Une bibliothèque de composants UI basée sur Material Design, pour des interfaces modernes et élégantes.
17. react-native-safe-area-context
    Gère les zones sûres (safe areas) dans les appareils modernes avec des encoches ou des bords incurvés.
18. react-native-screens
    Optimise la gestion des écrans pour améliorer la performance des applications React Native.
19. react-native-vector-icons
    Fournit une vaste bibliothèque d’icônes personnalisables pour les applications React Native.

React et autres

20. react
    La bibliothèque principale pour créer des interfaces utilisateur.

21. react-redux
    Intégration officielle entre React et Redux, pour connecter facilement le store Redux aux composants React.