**Project – Mobile App**

Lucile Pelou

Student n°74526

**Git repository:** [**https://github.com/Clarisse78/project\_74526**](https://github.com/Clarisse78/project_74526)

Build a shopping mobile app fulfilling the following requirements. You are allowed to make your own decisions on implementing the requirements, such as deciding layouts, data types, etc. Use common assumptions based on the real-world online shopping apps you have used e.g Cart total should have float values rounded to 2 decimals; Product quantity in cart can't be zero or negative. In case you could not decide between two design choices, check the best practices and see what's recommended. Log important design decisions in README.md

**I. Introduction**

Les consignes du projet étaient de faire une application mobile de vente avec les requirements demandés. Voici un rappel des requirements :

1. Authentication
   1. Allow User to Signup ✔️
   2. Log In using username and password ✔️
   3. Store userID once logged in to keep the user logged in (even after restarting the app) ✔️
2. Product Listing
   1. List Product Categories ✔️
   2. On clicking a Category, list Products in that Category ✔️
   3. On clicking a Product, show Product description, show buy button and controls to change quantity. ✔️
3. Cart
   1. Show cart summary ✔️
   2. Show total amount ✔️
   3. Purchase button to place an order, show order notification ✔️
4. Show order history
   1. List users orders ✔️
   2. On clicking an Order, show Order details and Products ordered ✔️
   3. On clicking a Product, take them to Product description page created for 2.3 ✔️
5. Show User details
   1. Use the stored userID to show user details ✔️
   2. Show a random circular profile image ✔️
   3. Show Logout button, on click take back to Signup / Log In page (Restart should not auto login after logout) ✔️
6. UI/Implementational Requirements
   1. Lazy lists to be used for all Lists: Categories, Products, Orders ✔️
   2. Add a small "About this app" button in the profile page, that shows a page on click with your copyright details and credits ✔️

Pour faire cette application et atteindre tous les objectifs demandés j’ai décidé d’utiliser Firebase. C’est facile d’utilisation et tout est déjà bien mis en place tel que l’authentification de Firebase qui permet de login, logout et sign up l’utilisateur.

**II. Conception et Architecture de l'Application**

Avant de me lancer dans quoique ce soit en termes de code ou de création de projet Firebase, j’ai fait un wireframe de basse qualité et j’ai noté les différentes collections avec les champs dont j’aurai besoin pour mener à bien le projet.

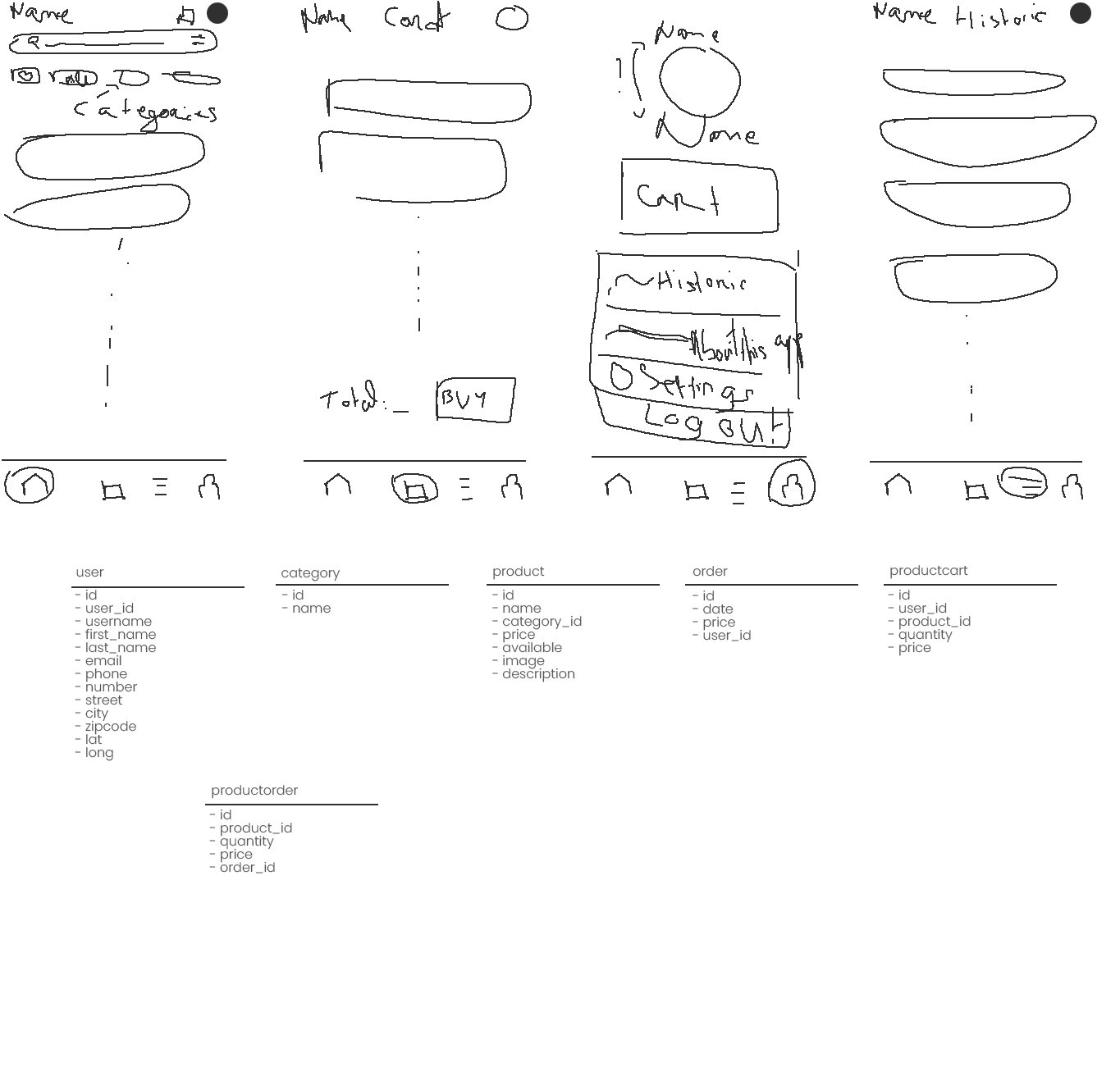


Figure 1: Wireframe & Collections

Ceci n’était bien sur qu’une ébauche du résultat final et des choses allait être amener à être changé. Au final voici l’UML de mes différentes collections :

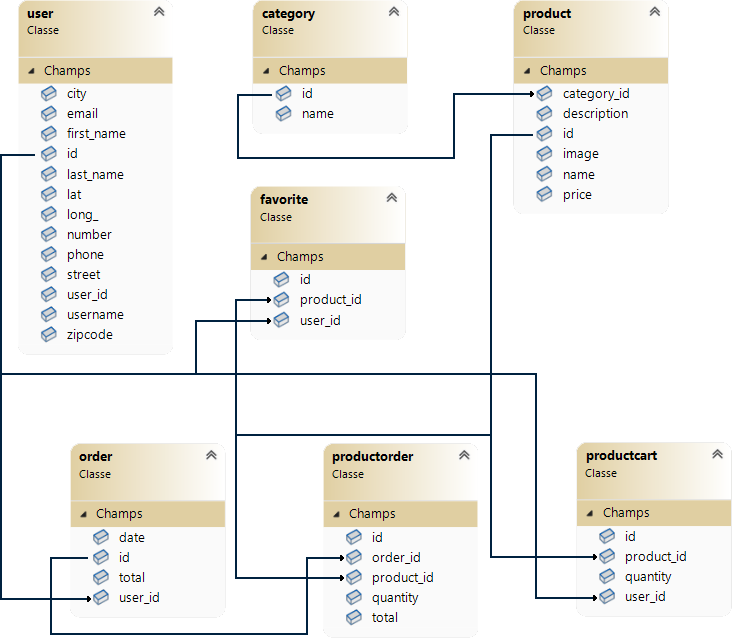


Figure 2: UML des différentes collections (classe)

J’ai donc fait 7 collections pour faire cette application mobile. La première celle des utilisateurs qui stock toutes leurs données. Le mot de passe n’est pas géré par la collection utilisateur mais par l’authentification de Firebase, qui stock également son email. Une collection category qui va permettre de stocker les différents noms des catégories et de pouvoir lier chaque produit à une d’entre elle. La collection produit possède un champ category\_id pour ce faire. Il y a également la collection order qui possède le champ user\_id car chaque order est relié à un utilisateur. Nous avons la collection productorder qui va permettre de récupérer un produit avec une quantité définie, tout cela rattaché à une order avec le champ order\_id. Il y a la collection productcart qui possède un champ user\_id et product\_id ainsi qu’une quantité, cela va permettre de stocker les produits qui sont dans le panier de l’utilisateur à l’heure actuelle. Et enfin nous avons la collection favorite qui va nous permettre de savoir si un utilisateur a ou non en favoris un produit avec les champs user\_id et product\_id.

Avec toutes ces collections et champs, l’application est désormais faisable.

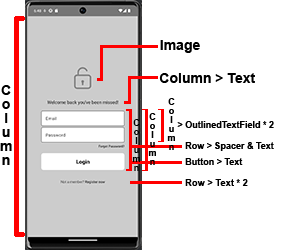
**III. Implémentation des Fonctionnalités**

Avant de commencer le projet sur Android Studio, j’ai créé un projet Firebase. Je l’ai ensuite relié à un projet sur Android Studio. Pour cela j’ai utilisé la documentation jetpack compose : <https://firebase.google.com/docs/android/setup?hl=en>.

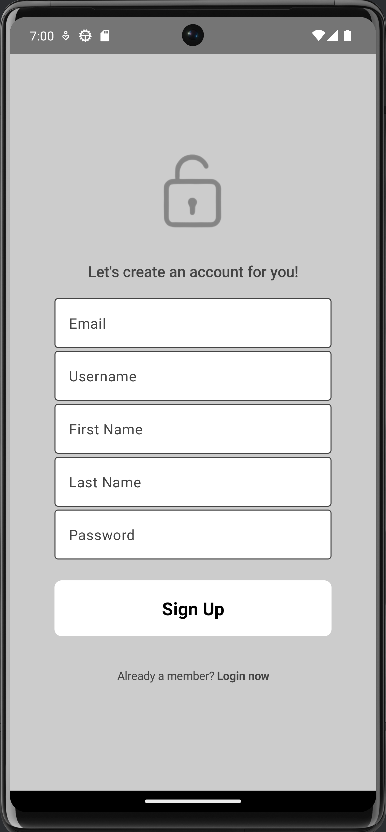
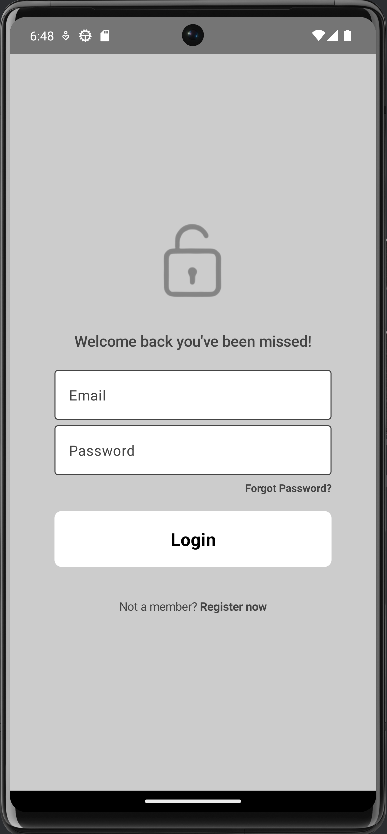
1. **Authentication : Page Login et Register**

J’ai débuté l’application par les pages login et register. Pour cela j’ai repris les pages effectuées dans l’assignement 1. J’ai ensuite ajouté la logique d’authentification de Firebase en m’aidant de la documentation jetpack compose : <https://firebase.google.com/docs/auth/android/start?hl=en>. J’ai aussi implémenté la navigation pour pouvoir naviguer entre la page login et la page register.

Voila un schéma concernant la mise en page pour la page login (c’est sensiblement la même que pour la page Register, je n’ai donc pas trouvé nécessaire d’en faire une pour cette seconde page) :



Et voilà les deux pages finales :



Je n’ai pas changé le design par rapport à l’assignement 1 parce que je trouvais que cela collait bien avec mon application.

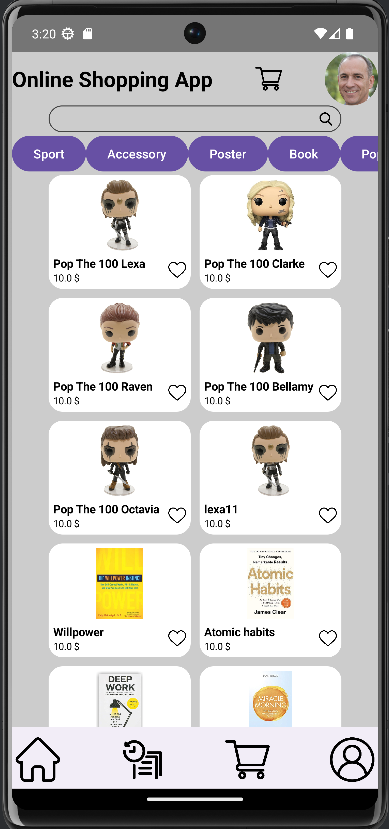
A l’inscription les utilisateurs ne peuvent pas encore rentrer toutes leurs informations, ils pourront les ajouter par la suite.

1. **Category & Product Listing**

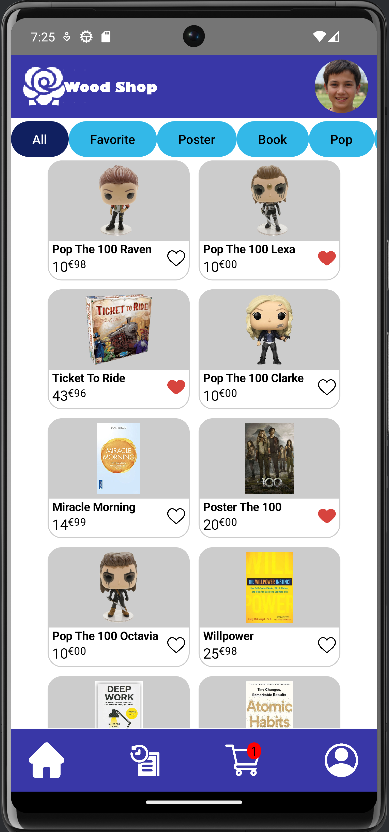
Après la page Login & Register, je me suis attaqué à la page home ou sont listé les différentes catégories et produits. Au tout départ je pensais faire une barre de recherche mais j’ai abandonné cette idée car je trouvais que cela ne rendait pas le tout très beau. Pour lister les différentes catégories, j’ai utilisé une lazy row et je les ai récupérés avec les méthodes données par Firebase Android Studio. Pour les produits, c’est fait de la même manière mais c’est une lazy grid qui est utilisé. J’ai eu beaucoup de mal à récupérer les données « au bon moment ». Le problème majeur que j’ai rencontré est la recomposition des composants concernés après la récupération de TOUTES les données. Il arrivait parfois que cela n’affiche pas les différentes catégories ou les différents produits car cela ne se recomposait pas une fois la récupération des données terminés. J’ai finalement réussi en utilisant la méthode LaunchedEffect, des méthodes suspend pour récupérer les données et un booléen mutable qui permet de savoir quand toutes les données ont été récupérées.

J’ai utilisé cette documentation pour la récupération des données : <https://cloud.google.com/firestore/docs/query-data/get-data?hl=en> ainsi que celle-ci pour les lazy lists: <https://developer.android.com/develop/ui/compose/lists>.

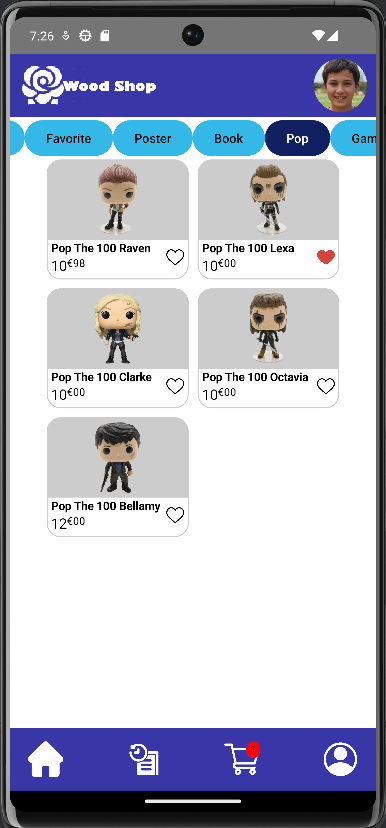
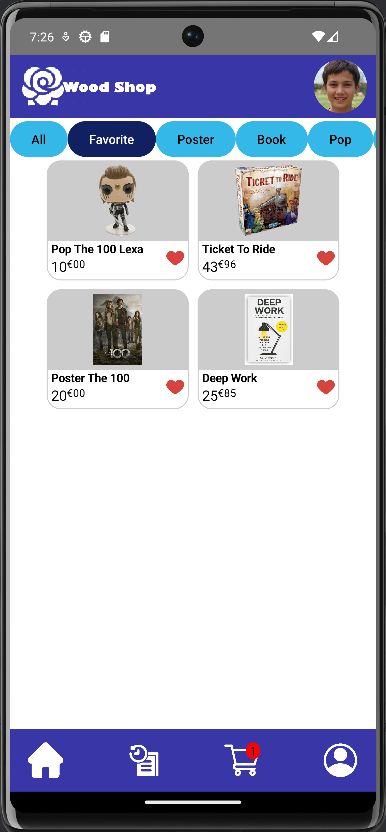
Voici mon résultat au départ :

****

Voilà la page finale :



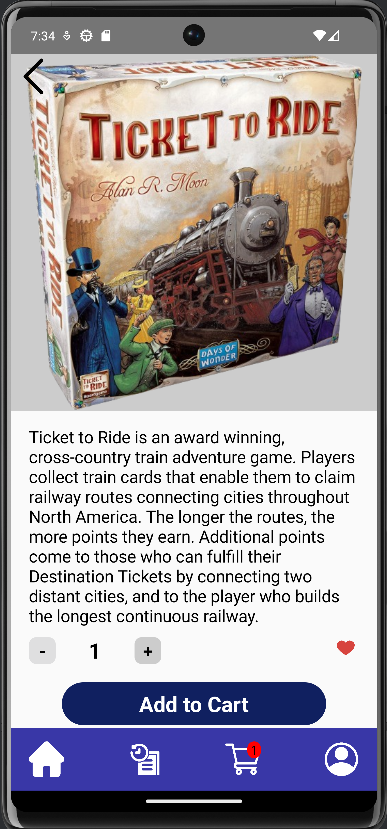
L’utilisateur peut cliquer sur les différentes catégories pour n’afficher que les produits de celle-ci.



L’utilisateur peut cliquer sur les produits pour afficher plus d’informations à leur sujets ou pour les ajouter à son panier.

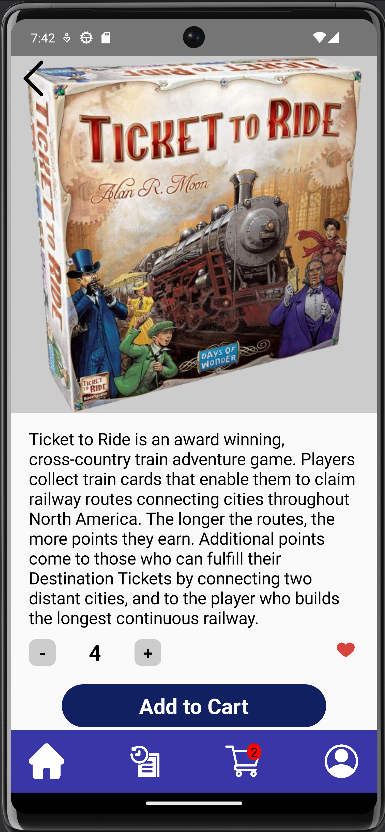
En cliquant sur la description il va pouvoir afficher la suite et il pourra scroller vers le bas pour afficher le reste de la page comme dans le screen ci dessous :



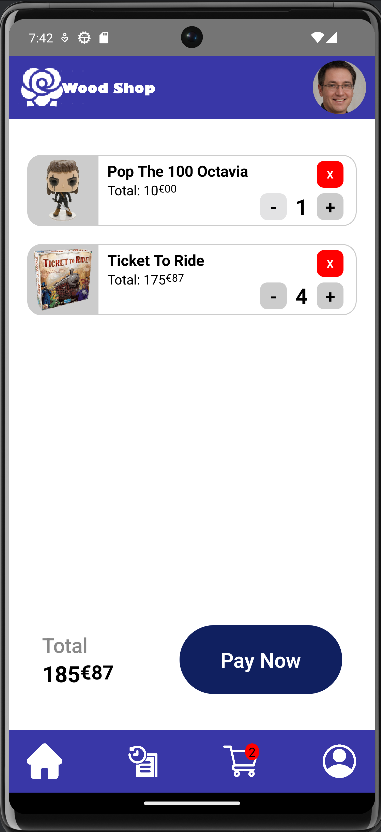
L’utilisateur peut également ajouter en favoris ses produits préférés en cliquant sur le cœur dans la page de la liste des produits ou en cliquant sur le cœur dans la page du produit concerné. Il peut aussi ajouter ou diminuer la quantité qu’il veut ajouter à la carte en cliquant sur les boutons plus et moins. Quand la quantité est à 1, le bouton moins est désactivé, sinon il s’active comme dans le screen ci-dessus :

1. **Cart**

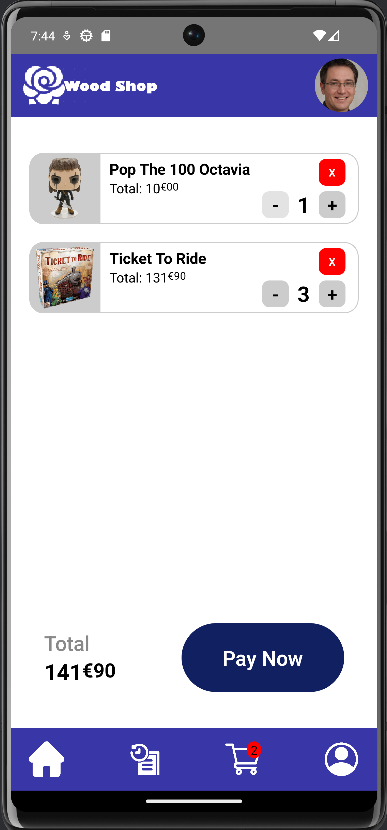
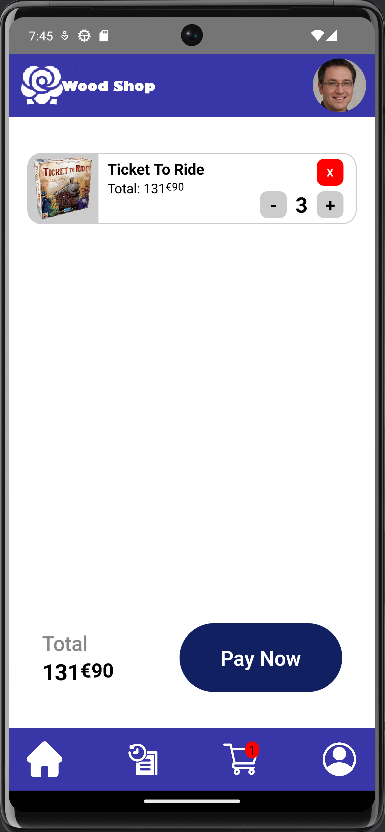
L’utilisateur peut donc ajouter des objets à son panier, il faut donc faire en sorte qu’il puisse voir le contenu de son panier. Tout d’abord j’ai ajouté un petit icone rouge pour indiquer le nombre de produit ajoutés dans celui-ci. Cela compte le nombre de produit différents et non le nombre de produits avec la quantité. Si par exemple j’ajoute à la carte 4 jeu ticket to ride :



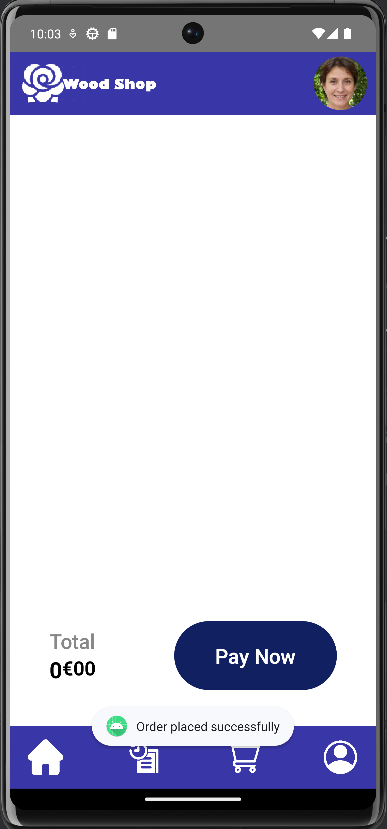
Le nombre est passé de 1 à 2. Et si nous allons voir dans le panier, nous retrouvons bien nos produits :



De cette page l’utilisateur va pouvoir contrôler son panier. Tout d’abord il peut supprimer et modifier la quantité de produits voulue. Par exemple si je ne veux plus que trois jeu ticket to ride car je trouve cela trop chère je peux cliquer sur le bouton -. Et si finalement je ne veux plus de figurine Pop Octavia je peux la supprimer en cliquant sur le bouton rouge « X ».

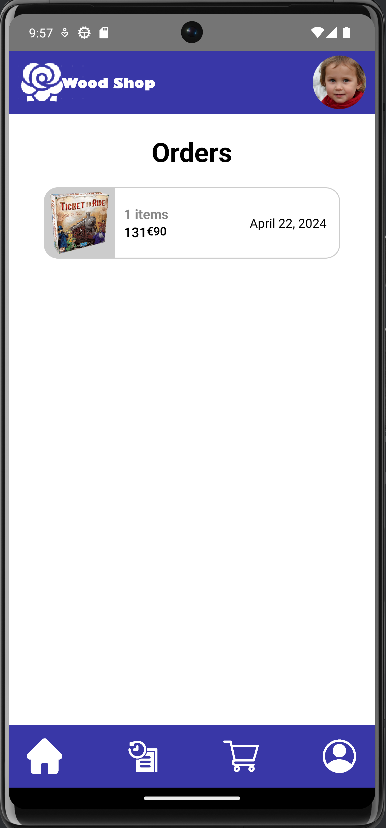
Maintenant si la commande me convient je vais pouvoir commander en appuyant sur le bouton « Pay Now ».



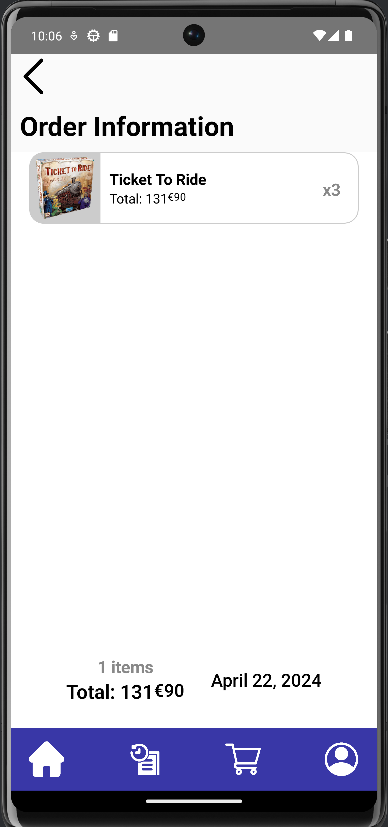
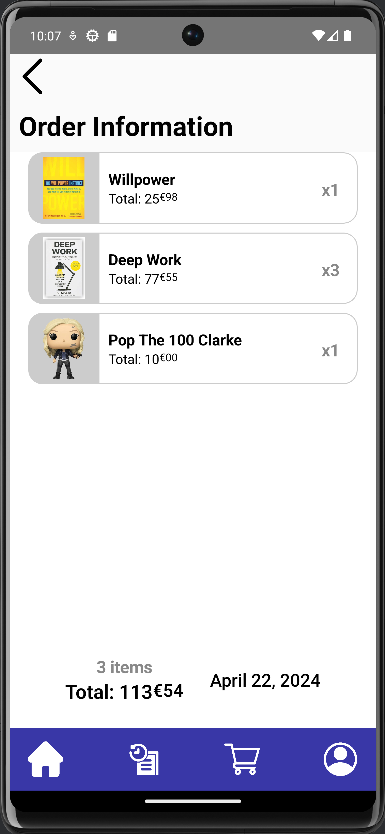
La commande a été passé avec succès.

1. **Order History**

Nous pouvons maintenant aller dans les orders pour retrouver nos commandes passées.



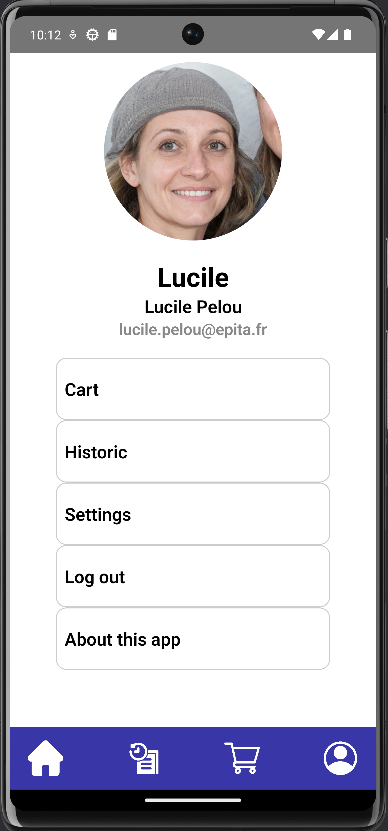
Nous pouvons voir ici la commande que l’on vient de passer. L’image montré dans l’aperçu est celui d’un des items de la commande. Si nous cliquons sur la commande, nous accédons aux détails de celle-ci. Voila un exemple avec la commande passée précédemment et une autre avec plus d’items.

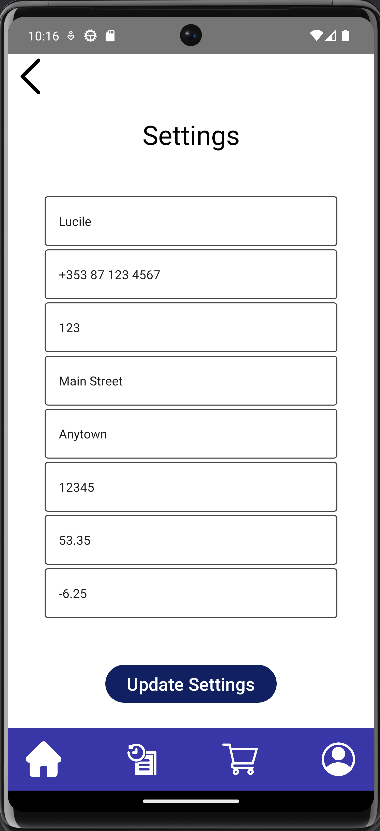
En cliquant sur les produits nous retournons à leur page de détails déjà montré précédemment.

1. **User Details**

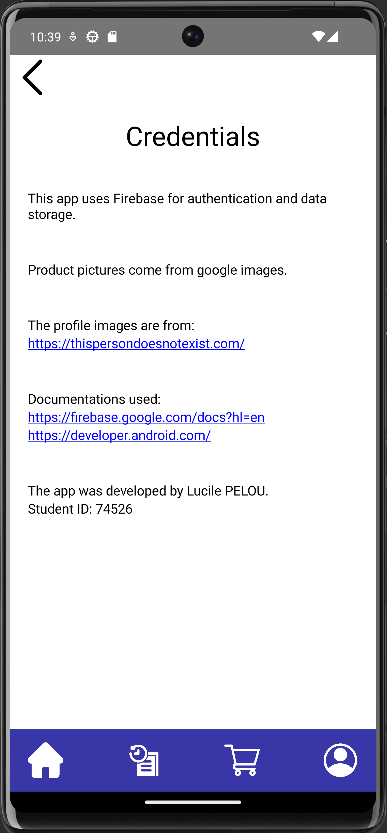
L’utilisateur peut également accéder à ses informations sur la page profile.



En cliquant sur « Cart » ou « Historic » il retombera sur les pages précédemment décrites. En cliquant sur « Settings » il pourra mettre à jour ses informations.



En appuyant sur « Update Settings » ces données vont être mis à jour. La photo de profil est générée par ce lien : <https://thispersondoesnotexist.com/>. L’utilisateur peut également se déconnecter en appuyant sur « Log out » ou voir plus d’informations sur l’application en appuyant sur « About this app ».



1. **User Details App bar & Bottom navigation bar**

Oui.

**V. Défis Rencontrés**

* Difficultés techniques
* Décisions de conception difficiles

**VI. Conclusion**

* Récapitulation des principales fonctionnalités implémentées
* Réflexion sur l'expérience globale du projet

**VIII. Références**

* Liens vers les ressources utilisées
* Documentation Firebase, jetpack compose, etc