Safia ALIOUCHE
Feriel BOUDEKHANI
Jeanne DEBETTE

Application Design – Projet n°2

SOMMAIRE :

Notre projet

Notre idée initiale était de pouvoir optimiser les courses en grande surface. Pour ce faire, il était prévu que l'utilisateur puisse entrer sa liste de course et que l'application puisse trier les différents éléments pour permettre à l'utilisateur de ne passer qu'une seule fois dans chaque rayon nécessaire à ses besoins.

Il nous fallait également une des fonctionnalités imposées. Nous avons fait le choix de faire l'authentification.

Toutefois, nous avons fait face à plusieurs problèmes. Une des ressources que nous avions repérées pour faire la liste de course n'était pas compatible avec la dernière version d'Android Studio. Nous avons donc changé nos plans et décidé d'intégrer un chat similaire à ChatGPT (cf. les parties de Feriel et Safia). Nous avons gardé l'authentification : cette fonctionnalité n'a pas été impactée par ce changement de dernière minute. S'ajoutent à cela des difficultés liées à GitHub. Suite à des problèmes et des confusions, nous avons fait le choix de nous répartir les fonctionnalités et de déposer notre travail au format .zip pour éviter toute confusion de version tout en pouvant distinguer le travail de chacune d'entre nous.

Finalement, après les rebondissements et changements de fonctionnalités, nous avons : une authentification ainsi qu'un chat semblable à ChatGPT

Nos participations individuelles

Safia

Initialement, je devais récupérer le travail de Feriel sur la liste de course afin d'implémenter une fonction de tri. Comme cela a été annulé tardivement, je me suis penchée sur une intégration de ChatGPT, étant quelque chose que j'ai toujours voulu essayer mais que je n'avais pas encore eu l'occasion de faire. Pour cela, je me suis aidée de quelques tutoriels ainsi que du site OpenAI.

L'application Android permet à l'utilisateur de saisir une question dans un champ de texte. En appuyant sur un bouton envoyer, la question est envoyée à l'API OpenAI. L'API génère ensuite une réponse basée sur la question et renvoie la réponse à l'application. La réponse est affichée dans une interface utilisateur sous forme de message.

Cela implique la création de requêtes HTTP POST avec des données JSON et la gestion des réponses de l'API. Comme les requêtes vers l'API OpenAI sont asynchrones, la gestion des

réponses nécessite des callbacks pour traiter les réponses lorsqu'elles sont reçues. Cela implique la mise en œuvre de l'interface Callback d'OkHttp et la gestion des réponses réussies ou en cas d'échec. Une fois que la réponse de l'API est reçue, elle est au format JSON. La partie complexe consiste à analyser cette réponse JSON pour extraire les informations pertinentes, telles que le texte de la réponse générée par l'API. Cela implique de naviguer à travers les structures JSON et d'extraire les données requises à l'aide des classes JSONObject et JSONArray de la bibliothèque JSON.
La gestion des erreurs de réseau, telles que les erreurs de connectivité ou les erreurs de serveur est également complexe.

Feriel

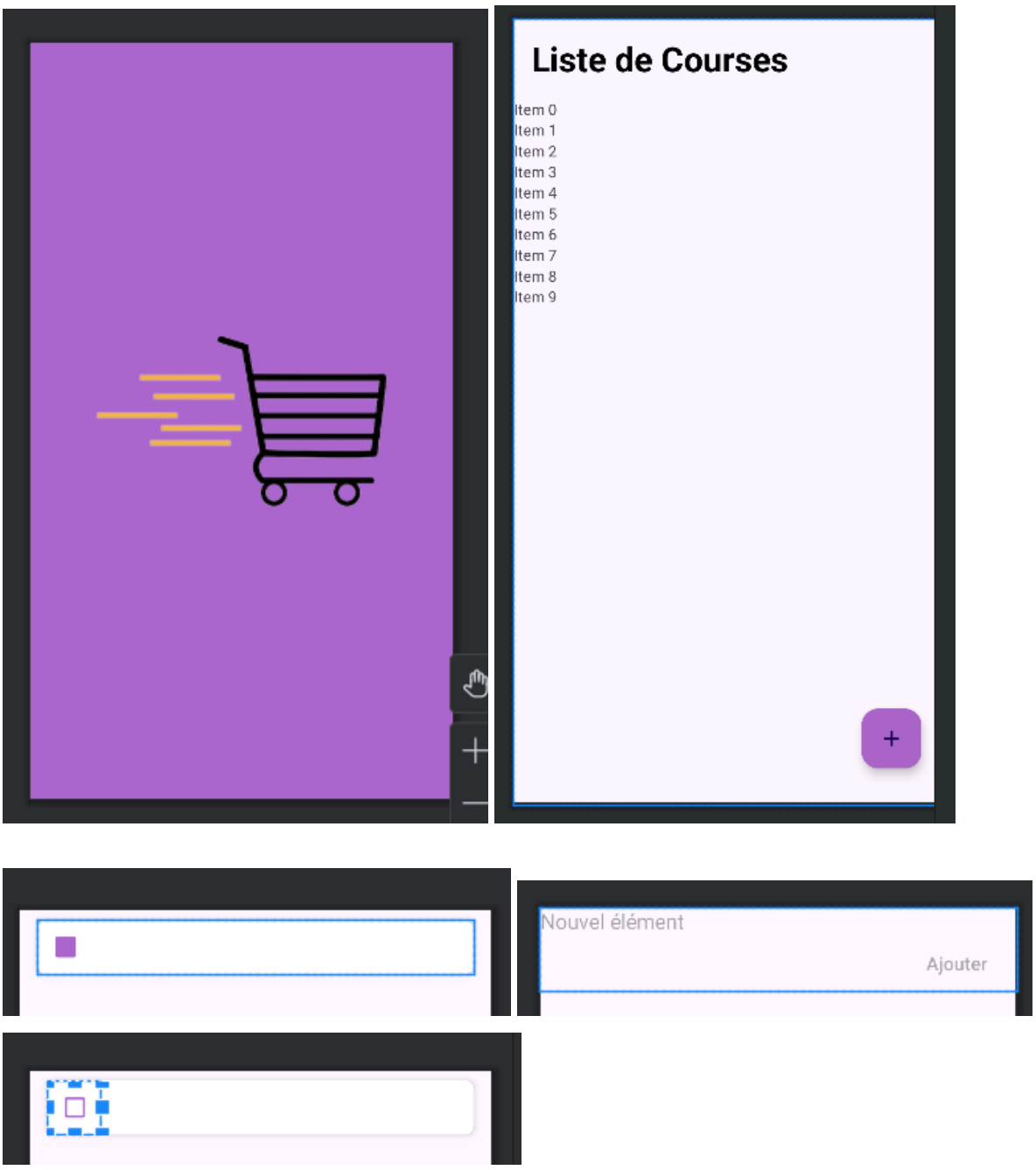
Lors de notre première idée, j'ai été chargée de faire la liste de course et de permettre à l'utilisateur de la remplir. La liste devait être réorganisée en fonction d'une base de données de tri. Par exemple, dans la colonne fruits on aurait retrouvé pomme, banane, fraise. Lorsqu'un utilisateur ajoute des bananes et des fraises à la liste, la liste créerait une catégorie fruits qui contiendrait ces deux objets. Bien que cette première idée n'ait pas pu aller jusqu'au bout, une version (non-fonctionnelle) de ce que j'ai pu faire en ce sens est sur notre repository GitHub. Elle contient les prémices de fonctionnalités. J'ai implémenté un bouton "ajouter", une zone de texte pour écrire le nom de l'élément à ajouter, la liste globale, ainsi que les différentes variables et classes nécessaires. Pour réaliser cela, j'ai suivi un tutoriel malheureusement obsolète.

https://www.youtube.com/playlist?list=PLzEWSvaHx_Z2MeyGNQeUCEktmnJBp8136

Le tuto consistait en :

- Vidéo 1: Interface utilisateur: Créer l'interface de base avec un bouton "Ajouter une tâche" et une liste vide.
- Vidéo 2: Ajouter des tâches: Implémenter la logique pour ajouter des tâches à la liste et les afficher dans une RecyclerView.
- Vidéo 3: Persistance des données: Implémenter la base de données SQLite et les opérations CRUD pour stocker et gérer les tâches.
- Vidéo 4: Amélioration de l'interface utilisateur: Améliorer l'apparence de la liste et ajouter des animations pour les tâches ajoutées ou supprimées.

Ci-dessous nous avons des visuels des différentes Views de notre application initiale :



Notre idée de départ étant abandonnée en cours de route, nous avons dû réorganiser les tâches entre Safia et moi (Jeanne n'a pas été impactée).

Jeanne

J'ai réalisé l'ensemble de la fonctionnalité d'authentification (XML+Java). J'ai donc conçu les deux Activities associées : l'inscription (Register) et le login (main activity). Il est possible de passer d'une page à une autre en fonction de si on a déjà un compte ou non. Chaque zone d'input est associée à la nature de l'input. Par exemple, pour remplir son mail à l'inscription, la zone d'input ne peut recevoir que des textes de nature "email address". De même, le mot de passe peut être invisible (avec des points remplaçant les caractères) ou visible, au choix de l'utilisateur, avec l'icône d'œil sur la droite de la zone d'input.

J'ai créé différents layout pour avoir une page de fond réutilisable et un layout contenant les fonctions d'authentification. Les éléments sont similaires entre eux et diffèrent par leur icône et la nature de l'input. Un dernier layout permet de positionner le texte en bas de l'écran qui redirige vers l'autre page (inscription si on est sur la page Login ; login si on est sur la page d'inscription).

Le plus difficile pour moi a été le design de la page (utilisation d'icônes, design fixe si on change d'appareil, relation entre les différents composants de la page), ainsi que de trouver les bonnes fonctions associées aux différentes zones d'input. La liaison entre les pages (fonctions en Java) n'ont pas été très difficiles puisque j'ai rapidement trouvé comment faire.

Je n'ai pas eu le temps de réaliser une base de données test et de programmer une vérification lors de l'authentification. C'est la piste d'amélioration principale de ce que j'ai réalisé pour ce projet.