3259.2 Développement mobile II– Rapport technique – ISC3il-b

|  |
| --- |
| **Shake ToDo** |

|  |
| --- |
| Étudiants participant à ce travail :  **Titus Abele, ISC3il-a Nicolas Aubert, ISC3il-b Tim Peck, ISC3il-a**  Présenté à :  **Aïcha Rizzotti Guillaume Digier**  Restitution du rapport : **12.05.2023**  Période : **2022 – 2023**  École : **HE-Arc, Neuchâtel** |

**Application de gestion des tâches, réalisée avec Android Studio**

Table des matières

[1 - Introduction 2](#_Toc134541942)

[2 - Analyse 3](#_Toc134541943)

[2.1 - Objectifs 3](#_Toc134541944)

[2.1.1 - Principaux 3](#_Toc134541945)

[2.1.2 - Secondaires 3](#_Toc134541946)

[2.2 - Maquettes 3](#_Toc134541947)

[3 - Conception 5](#_Toc134541948)

[3.1 - Base de données 5](#_Toc134541949)

[3.2 - Capteurs 5](#_Toc134541950)

[3.2.1 - Prise de photos 5](#_Toc134541951)

[3.2.2 - Accéléromètre 5](#_Toc134541952)

[3.3 - Settings 6](#_Toc134541953)

[3.4 - Notifications 6](#_Toc134541954)

[4 - Résultats 7](#_Toc134541955)

[4.1 - Visualisation des ToDos 7](#_Toc134541956)

[4.2 - Ajout d’un ToDo 7](#_Toc134541957)

[4.3 - Suppression d’un ToDo 7](#_Toc134541958)

[4.4 - Modification d’un ToDo 7](#_Toc134541959)

[4.5 - Compléter un ToDo 7](#_Toc134541960)

[5 - Limitations et perspectives 8](#_Toc134541961)

[6 - Conclusion 9](#_Toc134541962)

[7 - Annexes I](#_Toc134541963)

[7.1 - Table des illustrations I](#_Toc134541964)

[7.2 - Bibliographies et références II](#_Toc134541965)

[7.2.1 - Sites Web II](#_Toc134541966)

1. Introduction

L'application a été construite en utilisant le langage de programmation Kotlin et Android Studio comme environnement de développement. L'application est construite sur l'Android Framework, qui fournit une base solide pour la fonctionnalité et la performance de l'application. Le cadre Android comprend une variété d'outils et d'API qui sont essentiels pour le développement de toute application Android, y compris le SDK Android, le Runtime Android et la bibliothèque de support Android. Cette présentation fournira un aperçu détaillé des fonctionnalités, de la conception et de la mise en œuvre de l'application, ainsi que de tous les défis rencontrés au cours du processus de développement et de leurs solutions. En outre, il inclura les résultats des tests utilisateurs et les plans futurs pour l'application.

Le développement de l'application s'est appuyé sur l'environnement de développement intégré (IDE) Android Studio. Kotlin, le langage de programmation utilisé pour l'application, est un langage moderne, concis et expressif qui est totalement interopérable avec Java et qui offre plusieurs fonctionnalités telles que la sécurité des nullités, les fonctions d'extension et les coroutines qui améliorent la lisibilité et la maintenabilité du code. Android Studio, quant à lui, est l'IDE officiel pour le développement d'Android. Il propose une variété d'outils et de fonctionnalités qui simplifient le processus de développement, notamment un éditeur de mise en page visuelle, un éditeur de code et un débogueur. Ensemble, Kotlin et Android Studio constituent une plateforme de développement puissante et efficace qui permet de créer des applications mobiles de haute qualité.

L'application, ShakeToDo, est conçue pour fournir aux utilisateurs un moyen simple et efficace de gérer leurs tâches. La principale fonction de l'application est de permettre à l'utilisateur de créer des "ToDo", c'est-à-dire des tâches qui doivent être accomplies avant une date limite. L'utilisateur peut facilement ajouter des images à une tâche pour la visualiser et la rendre plus mémorable. En outre, l'application offre une fonction unique qui permet à l'utilisateur de secouer son téléphone lorsqu'une tâche est sélectionnée pour la marquer comme "terminée". Cette fonction constitue un moyen rapide et pratique pour l'utilisateur de suivre ses progrès et de rester au fait de ses tâches. Globalement, ShakeToDo offre une interface conviviale et visuellement attrayante qui rend la gestion des tâches simple et efficace pour l'utilisateur.

1. Analyse
   1. Objectifs
      1. Principaux

* Gestion des ToDos,
  + Ajout, modification et suppression d’un ToDo
* Interface utilisateur simple et intuitive
* Intégration caméra et accéléromètre pour ajouter des interactions avancées
  + Secouer pour le téléphone pour un ToDo comme complété,
  + Ajout d’une image à un ToDo.
    1. Secondaires
* Ajout de fonctionnalité "alarme" ou "notification" pour rappel à l'utilisateur.
  1. Maquettes



Figure 1 - Maquettes de l'application

L'interface de l'application ShakeToDo est conçue pour être intuitive et facile à utiliser pour les utilisateurs. Elle présente une interface visuellement attrayante qui permet à l'utilisateur de visualiser facilement ses tâches et de les gérer efficacement. La principale fonction de l'application est accessible à partir de l'écran principal, où l'utilisateur peut voir la liste de ses tâches en cours, avec des options pour ajouter une nouvelle tâche, marquer une tâche comme terminée ou supprimer une tâche existante.

Les tâches sont affichées de manière claire et concise, avec une description courte, une date limite et éventuellement une image associée.

Les éléments de l'interface sont disposés de manière logique et facile à comprendre pour l'utilisateur, avec des boutons et des icônes clairement étiquetés pour les différentes fonctions. Les menus et les sous-menus sont également bien organisés pour une navigation facile. La fonction unique de secouer le téléphone pour marquer une tâche comme terminée est également intégrée de manière transparente dans l'interface, avec des instructions claires pour l'utilisateur. En somme, l'interface de ShakeToDo est conçue pour offrir une expérience utilisateur fluide et agréable pour la gestion des tâches.

1. Conception
   1. Base de données

L'utilisation de Room pour créer la base de données d'une application Android offre une approche efficace et structurée pour la gestion des données. Room facilite la création et l'interaction avec la base de données en fournissant une couche d'abstraction. Les étapes clés pour utiliser Room consistent à ajouter les dépendances nécessaires dans le fichier build.gradle, définir les entités qui représentent les tables de la base de données, créer les interfaces DAO pour accéder aux données, et enfin, instancier et utiliser la base de données dans l'application.

En ajoutant les dépendances appropriées, Room offre une solution complète et fiable pour la gestion des données persistantes. Les entités permettent de définir la structure des tables et les annotations associées facilitent la gestion des contraintes et des relations entre les tables. Les interfaces DAO offrent une abstraction permettant d'effectuer des opérations de haut niveau sur la base de données, en utilisant des méthodes claires et intuitives qui masquent les détails de mise en œuvre des requêtes SQL. Enfin, l'instanciation et l'utilisation de la base de données se font de manière centralisée, ce qui garantit une gestion cohérente et efficace des ressources.

Lorsqu'une application effectue des appels à la base de données pour des opérations CRUD, il peut arriver que l'interface utilisateur se fige temporairement, empêchant ainsi les utilisateurs d'effectuer d'autres actions. Cette situation peut être frustrante et nuire à l'expérience utilisateur. Heureusement, il existe une solution pour éviter cela : l'utilisation de la classe ExecutorService en Kotlin.

L'ExecutorService est un composant puissant de la bibliothèque standard de Kotlin qui permet d'ordonnancer et d'exécuter des tâches de manière asynchrone. Dans le contexte de l'utilisation d'un ExecutorService pour les opérations CRUD, il est possible de lancer ces actions en arrière-plan, sans bloquer l'interface utilisateur et sans causer de frustration à l'utilisateur.

En utilisant un ExecutorService, les opérations CRUD peuvent être exécutées de manière indépendante sur des threads dédiés, tandis que l'interface utilisateur reste réactive et permet aux utilisateurs d'interagir sans aucun délai notable. Cela signifie que même lors de l'exécution de tâches complexes sur la base de données, les utilisateurs peuvent continuer à naviguer dans l'application, à consulter d'autres informations ou à effectuer d'autres actions sans aucun impact négatif sur leur expérience.

Pour lancer une action dans un executor, on fait appel à la classe Kotlin Executors qui contient la fonction newSingleThreadExecutor qui, lors de l’appel de execute, lance le code qui lui est fourni :



Ceci permet de lancer les tâches dans des threads séparés sans pour autant devoir créer des objets pour chaque appel.

* 1. Capteurs
     1. Prise de photos

La prise de photos est une fonctionnalité importante de l'application de gestion de tâches, car elle permet aux utilisateurs d'ajouter des images pour une meilleure organisation et une visualisation plus complète de leurs tâches. Cette fonctionnalité a été développée en utilisant le package androidx.camera, qui fournit une interface pour interagir avec la caméra du périphérique.

Pour utiliser la caméra du périphérique, l'application demande des permissions à l'utilisateur pour accéder à l'appareil photo et au micro. Une fois les permissions accordées, l'application ouvre une activité dédiée à la prise de photos, qui peut être réutilisée facilement dans d'autres parties de l'application. Lorsque l'utilisateur capture une image, celle-ci est enregistrée dans la galerie photos à l'aide de MediaStore, ce qui permet une gestion centralisée des images.

Cependant, une limitation de cette fonctionnalité est l'impossibilité de supprimer la photo après l'avoir capturée. Malgré les recherches effectuées, aucune méthode fiable n'a été trouvée pour supprimer l'image capturée.

* + 1. Accéléromètre

L'utilisation de l'accéléromètre est une fonctionnalité intéressante de l'application de gestion de tâches, car elle permet aux utilisateurs de marquer une tâche comme terminée en secouant leur téléphone. Cette fonctionnalité est intégrée dans l'activité qui affiche un ToDo, et utilise le SensorManager pour gérer l'accès aux capteurs.

Lorsque l'utilisateur accède à l'activité de visualisation d'un ToDo, l'application ajoute un listener sur le capteur d'accéléromètre. Dans ce listener, l'application récupère l'accélération du téléphone et la compare avec l'accélération précédente pour obtenir une valeur delta. Si cette valeur delta dépasse une certaine valeur, cela signifie que l'utilisateur a secoué son téléphone, et l'application marque la tâche comme terminée.

La valeur de la limite de delta à dépasser est récupérée à partir des Shared Preferences, qui sont présentés dans le chapitre suivant du rapport et qui parlent du menu des paramètres. Les Shared Preferences sont utilisées pour stocker des préférences utilisateur simples et peuvent être utilisées pour stocker des valeurs de seuil pour la fonctionnalité de secousse du téléphone.

* 1. Settings

La section "Paramètres de l'application" est un élément important de toute application mobile, car elle permet à l'utilisateur de personnaliser son expérience et de s'adapter à ses besoins spécifiques. Dans notre application de gestion de tâches, nous avons choisi d'ajouter un menu "paramètres" pour permettre à l'utilisateur de modifier les valeurs de sensibilité des capteurs utilisés pour détecter les secousses.

Nous avons implémenté un slider dans ce menu qui permet à l'utilisateur de définir la sensibilité de détection de secousses, avec une valeur minimale de 5 et une valeur maximale de 25. Lorsque l'utilisateur modifie la valeur du slider, celle-ci est enregistrée dans les Shared Preferences, qui sont un système de stockage local et persistant sous forme de clé-valeur.

Les Shared Preferences sont un moyen simple et pratique de stocker des données dans une application Android. Elles sont uniques à une activité et privées, mais peuvent être partagées entre toutes les activités de l'application. Dans notre cas, nous avons utilisé les Shared Preferences pour stocker la valeur de sensibilité de détection de secousses.

Cependant, nous avons noté que les Shared Preferences sont dépréciées et qu'il est conseillé d'utiliser DataStore à la place pour de meilleures performances

Enfin, il est important de noter que les capteurs utilisés pour détecter les secousses ont des valeurs différentes selon le téléphone. Par conséquent, le menu "paramètres" est un élément crucial pour permettre à l'utilisateur d'adapter la configuration de l'application à son téléphone.

* 1. Notifications
* Lorsqu’un todo est ajouté
* Background service
* ??

1. Résultats
   1. Graphical user interface, website

      Description automatically generatedVisualisation des ToDos

Après plusieurs itérations, l’affichage des ToDos a fini avec son état actuel. Pour la page affichante tous les ToDos, les images sont affichées à gauche du nom, avec le nom, la date, et la priorité du ToDo.

Le burger permet d’ouvrir un menu permettant le passage des ToDos aux paramètres de l’application.

Chaque ToDo est une listItem ajouté dans une ListView, comme ça les ToDos peuvent être ajoutés à la volée. Chaque ToDo a un onClickListener rattaché qui renvoie sur la page d’affichage d’un ToDo spécifique.

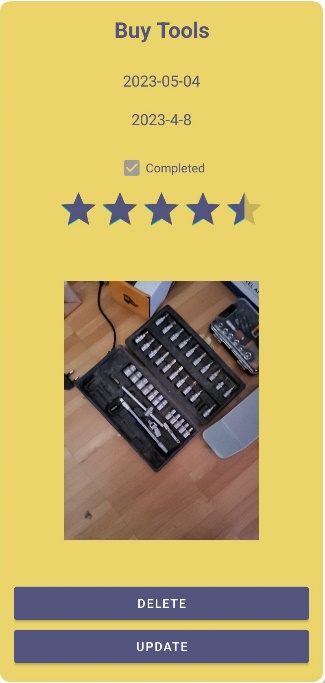
L’affichage des ToDos fonctionne parfaitement sur la majorité des tailles d’écrans, cependant cet affichage n’est pas idéal sur les écrans plus grands, par exemple une tablette.

Figure 4 Page d'affichage de tous les ToDos

A screenshot of a computer

Description automatically generated with low confidenceLa page d’affichage d’un ToDo utilise un mélange de layouts pour permettre un affichage hétérogène et compréhensible d’un ToDo. L’image est toujours redimensionnée pour l’afficher en entier.

Figure 3 Le menu burger ouvert

Figure 2 Affichage d'un ToDo spécifique

* 1. Ajout d’un ToDo

A screenshot of a phone

Description automatically generatedL’ajout d’un ToDo se fait via l’activity d’ajout d’un ToDo. Cette activité est faite avec une combinaison de layouts pour assurer le bon affichage.

Normalement, les couleurs suivent la charte graphique de l’application, cependant, il n’est pas possible de forcer la couleur du datePicker et du Spinner, qui suivent la couleur du téléphone. Il est donc possible que le texte soit blanc car l’utilisateur a son téléphone en mode sombre et donc l’application change la couleur automatiquement.

Les entrées textuelles ont tous des *placeholders* pour tirer l’utilisateur vers le bon contenu. Malheureusement, il est possible de créer un ToDo avec une date de complétion dans le passé.

Figure 5 Page de création d'un ToDo

* 1. Suppression d’un ToDo

La suppression d’un ToDo se fait depuis la page d’affichage d’un ToDo, une demande de confirmation est faite pour assurer que ceci est l’action voulue.

* 1. Modification d’un ToDo

Il est possible de modifier un ToDo, l’activité de modification est identique à celle de création, avec la précision que les champs sont déjà remplis. Il est toujours possible de rajouter une image, qui ouvre la caméra et renvoie ensuite sur la page avec la nouvelle image.

* 1. Compléter un ToDo

La completion d’un ToDo se fait sur la page du ToDo en lui-même en secouant le téléphone, la sensibilité de cette secousse peut être modifier dans les paramètres. Quand un ToDo est complété, son nom apparait barré dans la liste des ToDos. Sur la page du ToDo, la checkbox *completed* se remplit, il est possible d’annuler la complétion d’un ToDo en modifiant le ToDo.

1. Limitations et perspectives

La réalisation de l'application mobile de gestion de tâches à l'aide d'Android Studio a permis de créer une application pratique et fonctionnelle pour la gestion des todos. Cependant, il y a des limitations et des perspectives à considérer pour améliorer l'application.

Tout d'abord, une limitation importante est la visualisation des todos sous forme de calendrier. Actuellement, l'application ne permet pas de voir les tâches dans un calendrier, ce qui est une fonctionnalité très demandée par les utilisateurs. Pour améliorer l'application, il serait donc important d'implémenter une vue calendrier pour la gestion des tâches. De plus, il serait utile de permettre la synchronisation de l'application avec un autre calendrier, comme Google Calendar, pour une gestion plus globale et une synchronisation en temps réel.

Une autre limitation de l'application est liée à la version d'API utilisée. Actuellement, l'application utilise la version 26 d'API, ce qui est relativement ancien. Cela signifie qu'il est possible que l'application ne fonctionne pas toujours avec les nouvelles versions d'Android. Pour améliorer l'application, il serait donc nécessaire de mettre à jour la version d'API utilisée afin de s'assurer que l'application fonctionne correctement sur toutes les versions d'Android.

En outre, il y a eu des soucis avec Gradle et Android Manifest, car les versions utilisées ont affecté les imports. Pour améliorer l'application, il serait donc important de s'assurer que les versions utilisées sont compatibles avec les imports et de mettre à jour si nécessaire.

Une perspective intéressante pour améliorer l'application serait d'ajouter une fonctionnalité de Markdown pour les textes des todos. Cela permettrait aux utilisateurs de formater le texte de leurs tâches, en ajoutant des titres, des liens ou des images, pour une meilleure organisation et une meilleure lisibilité.

En outre, actuellement, l'application ne permet qu'une seule image par todo. Il serait donc intéressant d'ajouter la possibilité d'ajouter plusieurs images pour chaque tâche, pour une meilleure visualisation et organisation des tâches.

Une autre limitation importante est liée aux notifications. Actuellement, les notifications ne fonctionnent pas avec toutes les versions des API, ce qui peut causer des problèmes pour les utilisateurs qui ne reçoivent pas de notifications pour leurs tâches. Pour améliorer l'application, il serait donc important de trouver une solution pour permettre aux notifications de fonctionner sur toutes les versions des API.

Enfin, une perspective intéressante serait d'ajouter une fonctionnalité d'alarme pour les tâches importantes. Cela permettrait aux utilisateurs de ne pas oublier les tâches importantes et d'être rappelés à temps pour les réaliser.

En somme, l'application mobile de gestion de tâches créée à l'aide d'Android Studio a été très utile pour les utilisateurs, mais il reste des limitations et des perspectives à considérer pour améliorer l'application et répondre aux besoins des utilisateurs de manière plus efficace.

1. Conclusion

En conclusion, l'application mobile de gestion de tâches développée à l'aide d'Android Studio est fonctionnelle et a rempli tous les objectifs prévus initialement. Les utilisateurs peuvent facilement gérer leur liste de tâches, en ajoutant du texte et des images pour une meilleure organisation. De plus, la bonne UI/UX permet une utilisation intuitive de l'application, ce qui est un élément clé pour une expérience utilisateur satisfaisante.

En outre, l'application est évolutive grâce à son architecture bien conçue. Les fonctionnalités futures, telles que la visualisation de la liste de tâches sous forme de calendrier, la synchronisation avec un autre calendrier ou la possibilité d'ajouter plusieurs images pour chaque tâche, peuvent être implémentées facilement en utilisant les mêmes principes d'architecture.

En résumé, l'application mobile de gestion de tâches créée à l'aide d'Android Studio est une solution pratique et efficace pour la gestion des tâches. Elle a rempli tous les objectifs initialement prévus, avec une bonne UI/UX et une architecture évolutive. Cette application peut être une réponse satisfaisante aux besoins des utilisateurs pour la gestion de leurs tâches quotidiennes.

1. Annexes
   1. Table des illustrations

[Figure 1 - Maquettes de l'application 4](#_Toc133567108)

* 1. Bibliographies et références
     1. Sites Web

Prendre des photos. (n.d.). *Android Developers*. <https://developer.android.com/training/camera-deprecated/photobasics?hl=fr>

Enregistrer des données dans une base de données locale à l’aide de Room. (n.d.). *Android Developers*. <https://developer.android.com/training/data-storage/room?hl=fr>

Mettre à jour les composants de l’interface utilisateur avec NavigationUI. (n.d.). *Android Developers*. <https://developer.android.com/guide/navigation/navigation-ui?hl=fr>

*Notifications are working on an emulator but not on a real device*. (n.d.). Stack Overflow. <https://stackoverflow.com/questions/51667324/notifications-are-working-on-an-emulator-but-not-on-a-real-device>

Créer des icônes d’application. (n.d.). *Android Developers*. <https://developer.android.com/studio/write/create-app-icons?hl=fr>

Spinners. (n.d.). *Android Developers*. <https://developer.android.com/develop/ui/views/components/spinner>

GeeksforGeeks. (2022). How to Detect Shake Event in Android. *GeeksforGeeks*. <https://www.geeksforgeeks.org/how-to-detect-shake-event-in-android/>