|  |  |  |
| --- | --- | --- |
| EPCF3 | 26 Septembre  2024 | |
| Dossier projet – Fresh ! La préparation du déploiement d’une application sécurisée. | | EXPERNET |



Table des matières

[I. Liste des compétences du référentiel qui sont couvertes par le projet 2](#_Toc172730872)

[II. Expression des besoins du projet pour définir les objectifs et les limites du projet 3](#_Toc172730873)

[III. Environnement technique 5](#_Toc172730874)

[IV. Réalisations permettant la mise en œuvre des compétences. 6](#_Toc172730875)

[Étapes à réaliser pour créer une course : 7](#_Toc172730876)

[Jeu de donnée : 7](#_Toc172730877)

[Tests unitaires : 8](#_Toc172730878)

[Étapes à réaliser pour ajouter des produits à une course : 9](#_Toc172730879)

[Jeu de donnée : 9](#_Toc172730880)

[Tests unitaires : 11](#_Toc172730881)

[Signature de l’application : 12](#_Toc172730882)

[Tests de déploiement : 13](#_Toc172730883)

[V. Annexes 14](#_Toc172730884)

[Annexe 1 : méthode « buildAlertDialog() » 14](#_Toc172730885)

[Annexe 2 : méthode « add() » et méthode « persist() » 16](#_Toc172730886)

# Liste des compétences du référentiel qui sont couvertes par le projet

|  |  |
| --- | --- |
| Préparer le déploiement d’une application sécurisée | Préparer et exécuter les plans de tests d’une  application |
| Préparer et documenter le déploiement d’une  application |
| Contribuer à la mise en production dans une démarche  DevOps |

Le projet couvre obligatoirement les compétences suivantes :

* Préparer et exécuter les plans de tests d’une application
* Préparer et documenter le déploiement d’une application
* Contribuer à la mise en production dans une démarche DevOps

# Expression des besoins du projet pour définir les objectifs et les limites du projet

Le présent cahier des charges définit les exigences et les spécifications pour le développement d'une application mobile de gestions de courses.

• Contexte

Fresh souhaite mettre en place une solution permettant à n’importe qui, la possibilité de créer des listes de courses depuis son téléphone.

• Fonctionnalités

Les fonctionnalités permettent de savoir ce qui est fonctionnel pour l’utilisateur

1. L’application a une page d’accueil.

* Un bouton « Créer une course » est cliquable et permet de créer une course dans un pop-up.
* Listage des courses à faire.
* Chaque élément à une date à faire (sauf cas contraire)
* Chaque élément est cliquable qui permet d’ajouter la date à faire (si elle n’est pas encore attribuée) avec un pop-up ou afficher un pop-up récapitulatif de la course, où l’on peut modifier la course, manager les produits de la course et marquer la course comme « terminer »
* Un bouton « Voir les courses déjà effectuées » est cliquable et permet de voir la liste de courses ayant été terminées.

1. L’application a une page de management des produits de course

* Un bouton « Retour » est cliquable et permet de revenir à la page d’accueil.
* Un récapitulatif de la course est disponible avec la date de création de la course, la date à faire et la date quand la course est terminée (sauf cas contraire)
* Un bouton « Modifier » est cliquable et permet de modifier le nom de la course, la date à faire et de marquer la course comme « terminer »
* Listage des produits.
* Chaque élément affiche la date de création de la course
* Chaque élément qui n’a pas encore été pris est cliquable et permet d’afficher un pop-up qui contient un bouton « Modifier », qui permet de modifier la ligne du produit et un bouton « Marquer comme pris » qui permet de catégoriser le produit comme pris avec la date à laquelle il a été pris.
* Un bouton « Ajouter un produit » est cliquable et permet d’afficher un pop-up qui contient des champs libres pour indiquer le nom du produit, sa quantité et son unité (champ facultatif)
* Un bouton « Marquer comme terminer » qui permet de classer la course comme terminer avec la date à laquelle elle a été terminée.

• Technologies utilisées

* Développement de l’application
* Java
* Gradle Kotlin
* Android SDK API 34

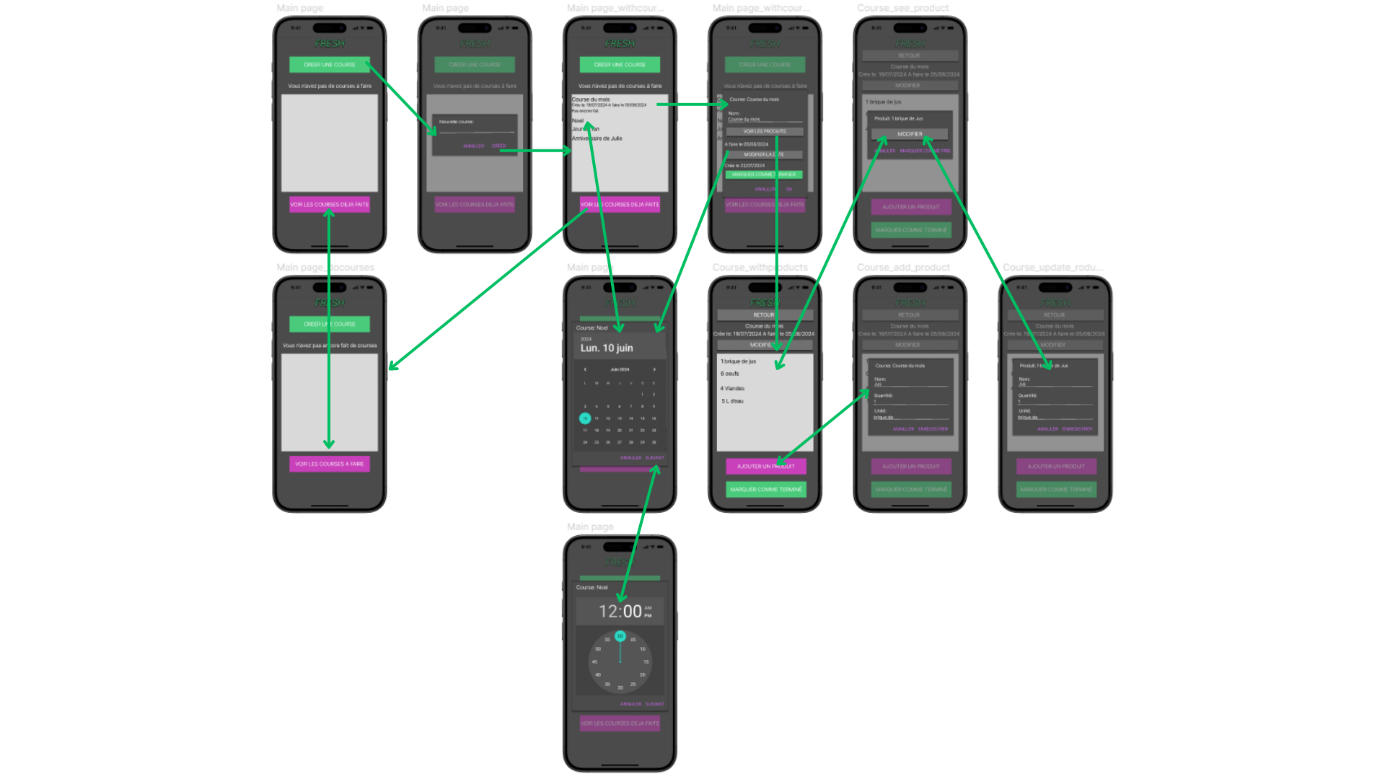
• Livrable et délais

* L’application complète et fonctionnelle doit être livrée conformément aux spécifications.
* La date de livraison prévue pour la plateforme est le 26/07/2024.

# Environnement technique

* Développement de l’application
  + Android Studio Koala
* Déploiement de l’application
  + Git 2.45.1 sur Github : [SteveHoareau18/MSP3 (github.com)](https://github.com/SteveHoareau18/MSP3)
* Tests de l’application
  + JUnit 4 et tests manuels

# Réalisations permettant la mise en œuvre des compétences.



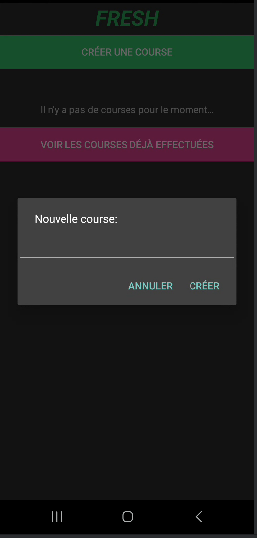
⏏Enchaînement de maquette⏏

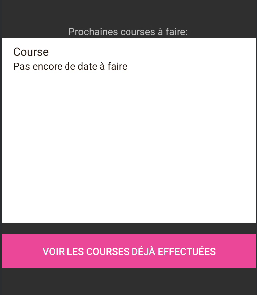


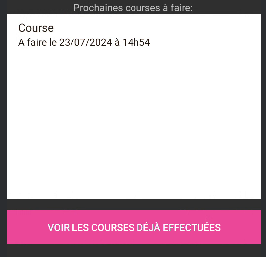
⏏Page d’accueil⏏

## Étapes à réaliser pour créer une course :

* Cliquer sur le bouton « Créer un course »
* Entrer le nom de de la course (facultatif) puis cliquer sur le bouton « Créer »
* Définir la date et l’heure à laquelle la course doit être effectuée.





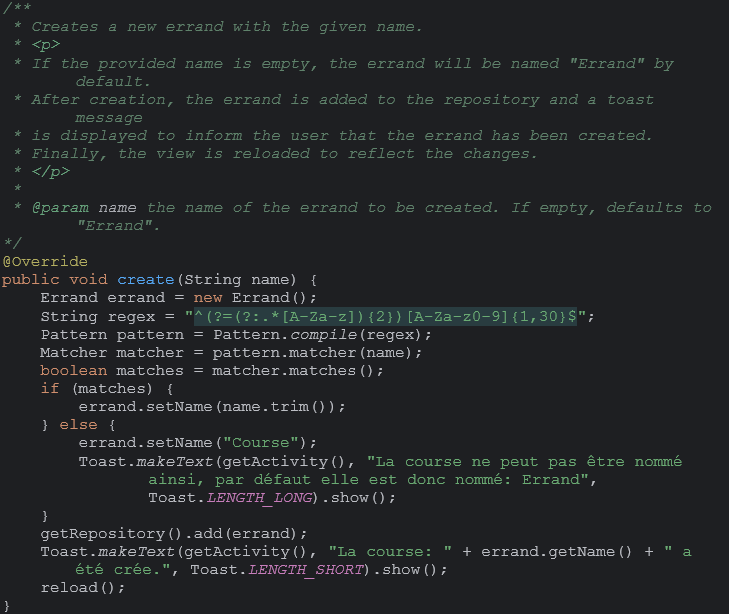


### Jeu de donnée :

* Données attendues : Toutes types de données entre 2 et 30 caractères (dont 2 lettres obligatoires) ou Aucune

|  |  |  |
| --- | --- | --- |
|  | Données en entrée | Données obtenues |
| Aucune donnée | Aucune | « Course » |
| Données ne correspondantes pas aux critères attendus | * « 12 » * « C » * « Coursequifaitplusdetrenteslettres » | « Course » |
| Données correspondantes aux critères attendus | * « Course du mois » * « Course du 12 » * « Course du 123 » * « Espace » | * « Course du mois » * « Course du 12 » * « Course du 123 » * « Espace » |

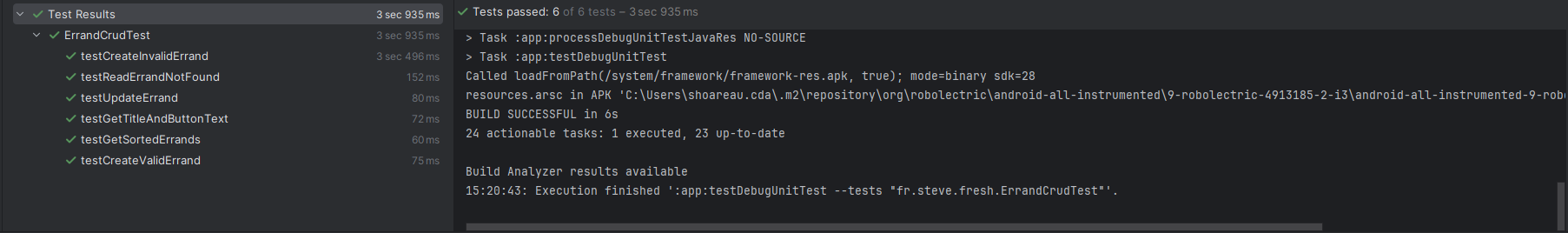
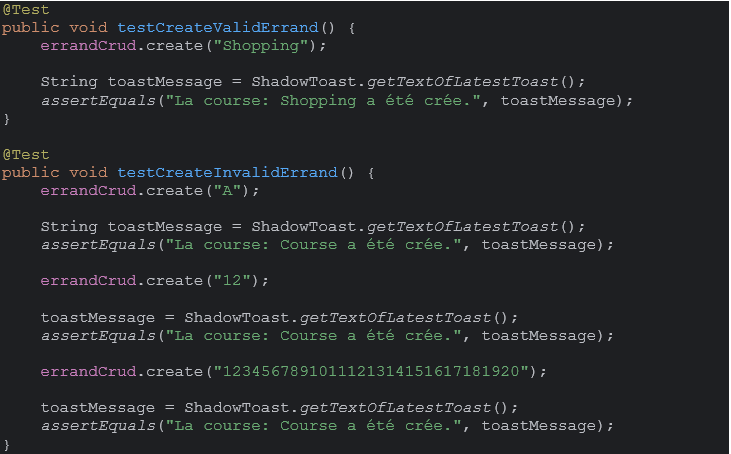
Code :



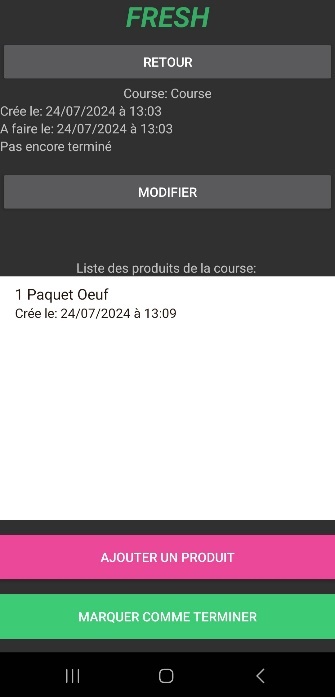
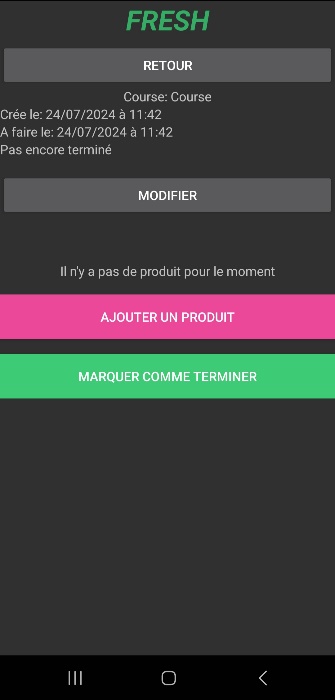
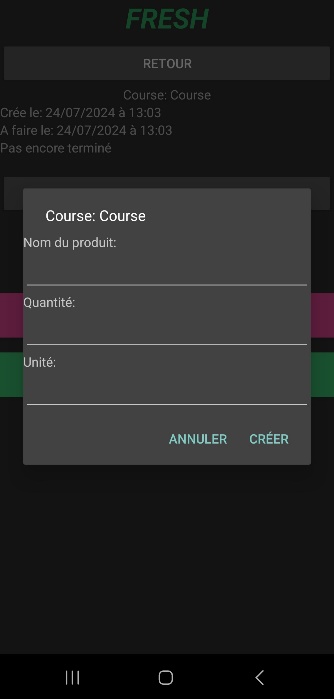
⏏Méthode « create() », permettant à créer une course.⏏

[Voir annexe 2 pour la méthode « getRepository().add() »](#_Annexe_2_:)

### Tests unitaires :

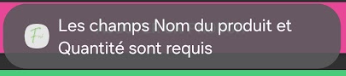
## Étapes à réaliser pour ajouter des produits à une course :

* Cliquer sur la course
* Cliquer sur le bouton « Voir les produits »
* Cliquer sur le bouton « Ajouter un produit »
* Entrer obligatoirement le nom et la quantité du produit, l’unité est facultative.
* Cliquer sur le bouton « Créer »

### Jeu de donnée :

* Données attendues :
  + Pour le champ « Nom du produit » : toutes types de données entre 2 et 30 caractères (dont 2 lettres obligatoires)
  + Pour le champ « Quantité » : un entier strictement supérieur à 0 et inférieur à 1000
  + Pour le champ « Unité » : toutes types de données entre 1 et 15 caractères (dont 1 lettre obligatoire) ou Aucune

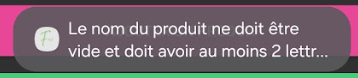
|  |  |  |
| --- | --- | --- |
|  | Données en entrée | Données obtenues |
| Aucune donnée | Nom du produit : Aucune  Quantité : Aucune  Unité : Aucune | Aucune   * Erreur avec affichage en Toast (1) |
| Aucun nom du produit | Nom du produit : Aucune  Quantité : 1  Unité : L | Aucune   * Erreur avec affichage en Toast (1) |
| Aucune quantité | Nom du produit : « Jus »  Quantité : Aucune  Unité : L | Aucune   * Erreur avec affichage en Toast (2) |
| Nom du produit ne correspondant pas aux critères | * Nom du produit : « J » * Nom du produit : « J1 » * « Nomduproduitquidépassetrentecaractères »   Quantité : 1  Unité : L | Aucune   * Erreur avec affichage en Toast (3) |
| Quantité du produit ne correspondant pas aux critères | Nom du produit : « Jus »   * Quantité : 0 * Quantité : -1 * Quantité : 1000   Unité : L | Aucune   * Erreur avec affichage en Toast (4) |
| Unité du produit ne correspondant pas aux critères | Nom du produit : « Jus »  Quantité : 1   * Unité : « 1 » * Unité : « Unitéquidépassetrentecaractères » | Aucune   * Erreur avec affichage en Toast (5) |
| Données correspondantes aux critères attendus | * Nom du produit : « Jus » * Quantité : 1 * Unité : « L » * Nom du produit : « Jus  » * Quantité : 1 * Unité : « L  » * Nom du produit : « 7up » * Quantité : 999 * Unité : « cl » * Nom du produit : « Oeuf cat 1 » * Quantité : 1 * Unité : « paquet » * Nom du produit : « Carreaux » * Quantité : 15 * Unité : « m2 » | * Nom du produit : « Jus » * Quantité : 1 * Unité : « L » * Nom du produit : « Jus » * Quantité : 1 * Unité : « L » * Nom du produit :   « 7up »   * Quantité : 999 * Unité : « cl » * Nom du produit :   « Oeuf cat 1 »   * Quantité : 1 * Unité : « paquet » * Nom du produit : « Carreaux » * Quantité : 15 * Unité : « m2 » |



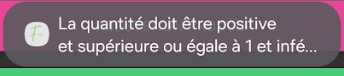
Toast (1) :



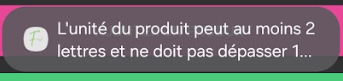
Toast (2) :



Toast (3) :



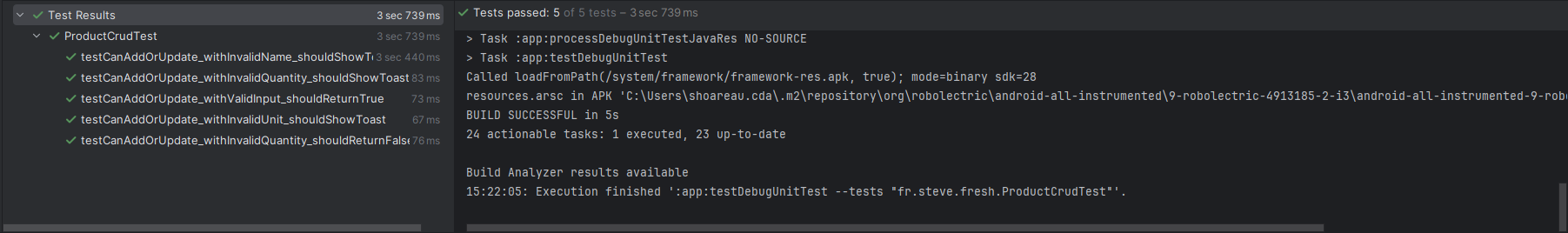
Toast (4) :



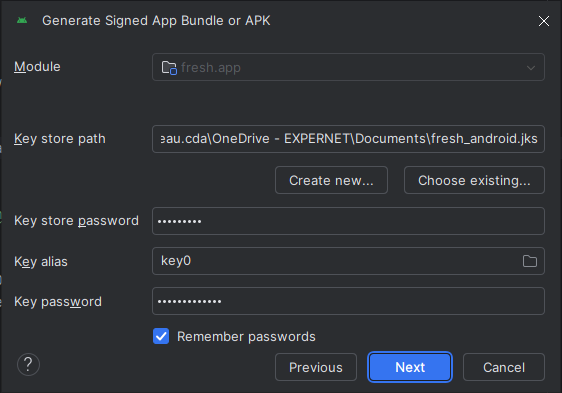
Toast (5) :

### Tests unitaires :

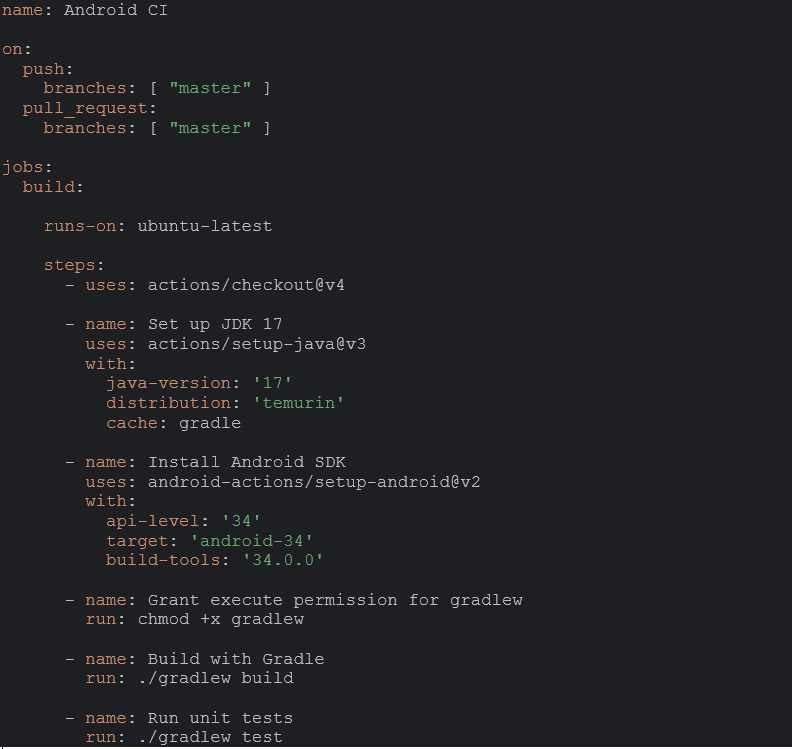




## Signature de l’application :

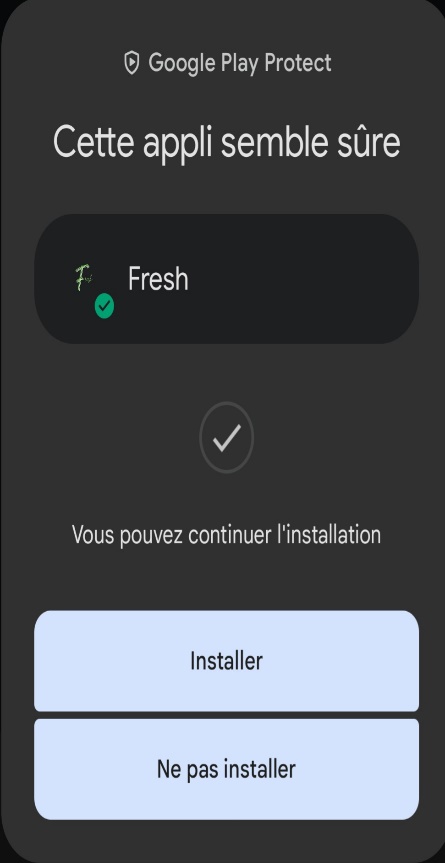
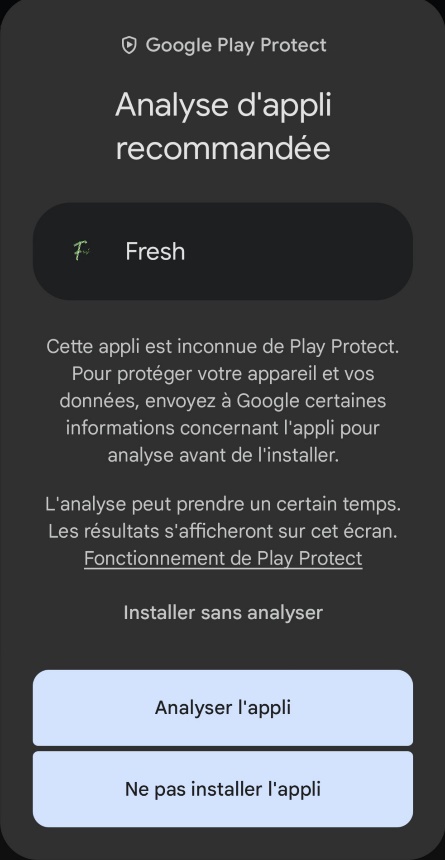


## Tests de déploiement :



⏏Workflow de déploiement sur Github⏏



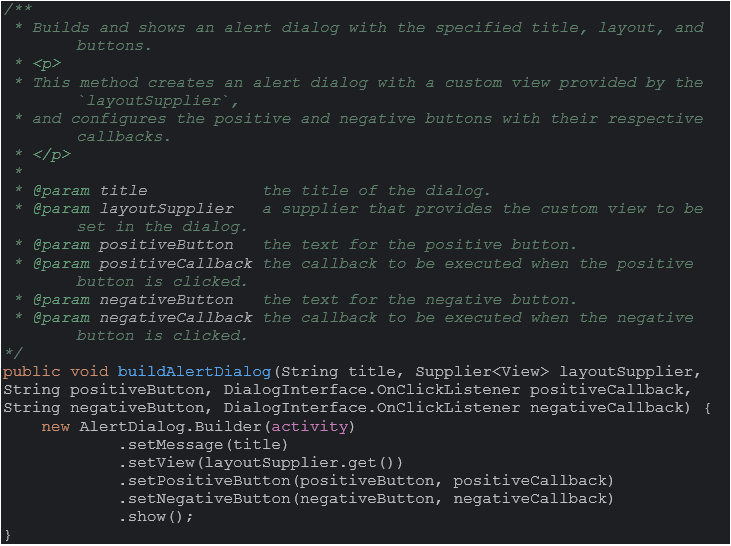


⏏Lors de l’installation de l’application⏏

# Annexes

## Annexe 1 : méthode « buildAlertDialog() »

La méthode « buildAlertDialog() » se situe dans la classe  « Dialog<P extends IPage> » et permet de générer un pop-up.

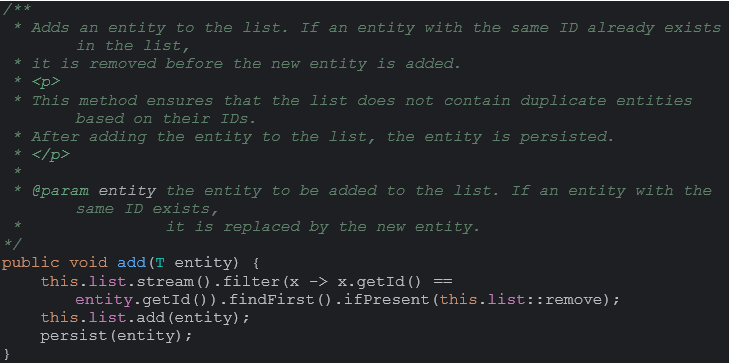


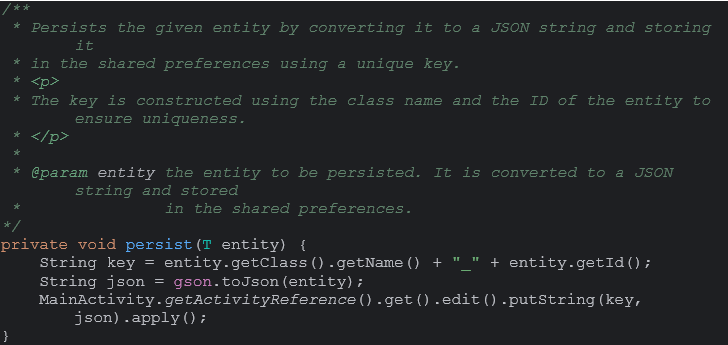
## Annexe 2 : méthode « add() » et méthode « persist() »

Les méthodes « add() » et « persist() » se trouvent dans la classe « Repository<T extends Entity> ».

La méthode « add() » permet d'ajouter une entité à la liste, tandis que la méthode « persist() » permet d'enregistrer l'entité.

L'entité peut être une course ou un produit, et chaque type d'entité possède sa propre liste associée.





La méthode « persist() » utilise le SharedPreferences.