

Livrable	2
Sigle du cours	SEG2505
Professeur	Aziz Oukaira
Assistant (e)	Alexia Capo-Chichi
Groupe	G03

Membres du groupe

- Céline Wan Min Kee (celinewmk), 300193369
- Evan Marth (emarth), 300166093
- Samy Touabi (SamyT-code), 300184721
- Tisham Islam (TishamIslam), 300189261
- Othniel Tiendrebeogo (othnielt), 300084968

Livrable 2: Implémentation des fonctionnalités liées à l'administration

1. Diagramme UML pour le livrable 2

Nous avons continué le diagramme UML de notre application en ajoutant les classes dont nous avons besoin afin de réaliser le livrable 2 (il y a aussi quelques méthodes incluses pour les livrables prochaines). Ensuite nous avons partagé les tâches entre les différents membres du groupe. Nous avons ajouté 6 activités qui seront nécessaire pour ce livrable:

- AdminPage: page d'accueil pour les comptes administrateurs
- AccountDeletionPage: page qui nous affiche les comptes employé et client créé et qui nous permet de les supprimer
- ServicesPage: page qui nous affiche la liste des services offerts et nous donne la possibilité d'aller sur l'activité AddService, ServiceDetails ou ModifyDeleteServicePage
- AddService: page qui nous permet d'ajouter un service
- ModifyDeleteServicePage: page qui nous permet de modifier ou supprimer un service
- ServiceDetails: page qui nous affiche les détails d'un service

Nous avons également ajouté AccountList et ServiceList qui nous permettent d'avoir une liste scrollable pour afficher les comptes et les services.

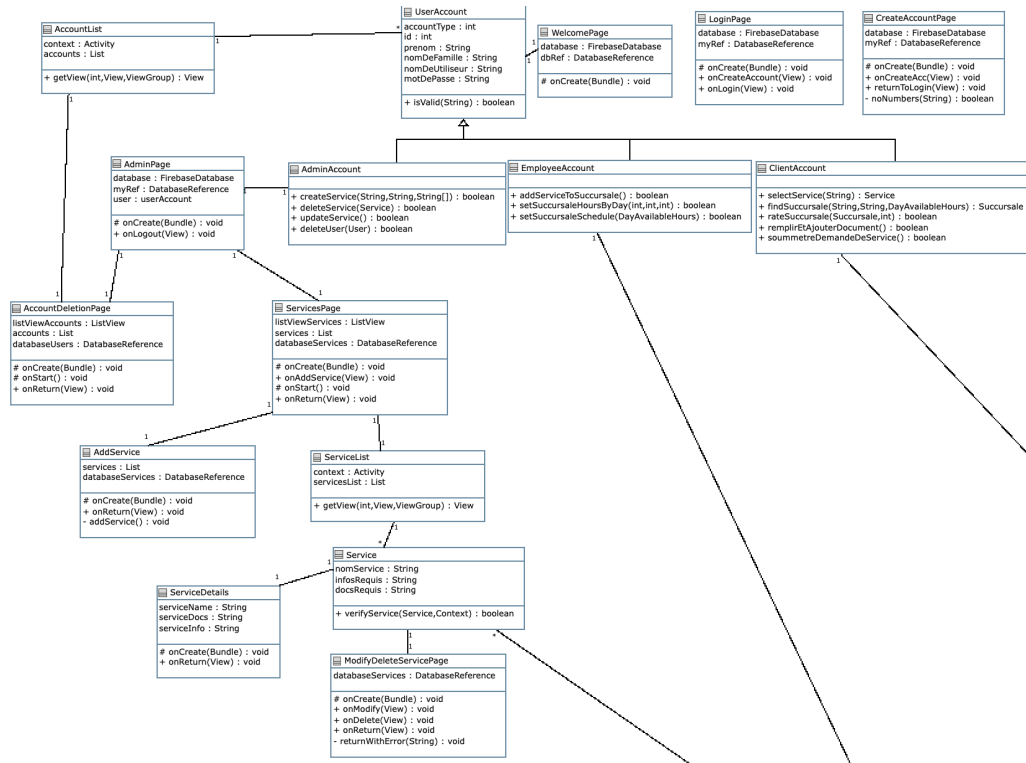


Figure 1: Diagramme UML

Céline a créé la classe Service afin de pouvoir créer des services pour l'application.

```

public class Service {

    // instance variables
    private String nomService;
    private String infosRequises;
    private String docsRequis;

    // Constructor 1
    public Service() {

    }

    // Constructor 2
    public Service(String nom, String infos, String docs){
        this.nomService = nom;
        this.infosRequises = infos;
        this.docsRequis = docs;
    }

    // Getters
    public String getNomService() {
        return nomService;
    }

    public String getInfosRequises() {
        return infosRequises;
    }

    public String getDocsRequis() {
        return docsRequis;
    }

}

```

Figure 2: Classe Service

2. Design des nouvelles activités de l'application

Céline, Samy, Evan et Tisham ont travaillé sur le design et le front-end des nouvelles activités de l'application en utilisant ce qui a été appris dans les laboratoires sur Android Studio.

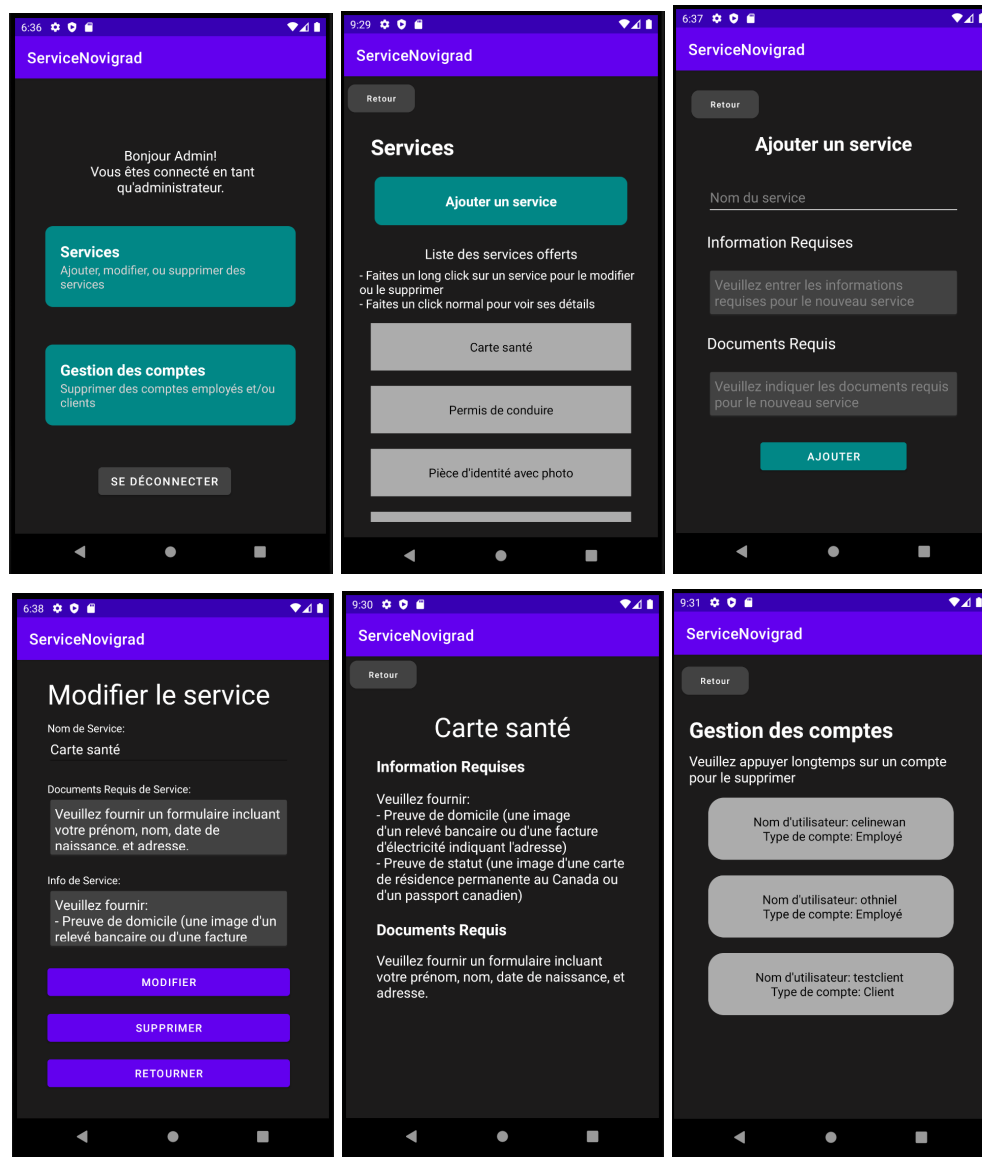


Figure 3: Les pages principales des services administratifs: le “landing page” pour administrateur, les pages pour voir, ajouter et modifier/supprimer les services, et la page pour voir et supprimer les comptes employés et clients.

Samy et Tisham ont ajouté les messages Toast lorsqu'il y a des champs de textes invalides ou lorsqu'un service a été ajouté, modifier ou supprimer.

- Ajout d'un service

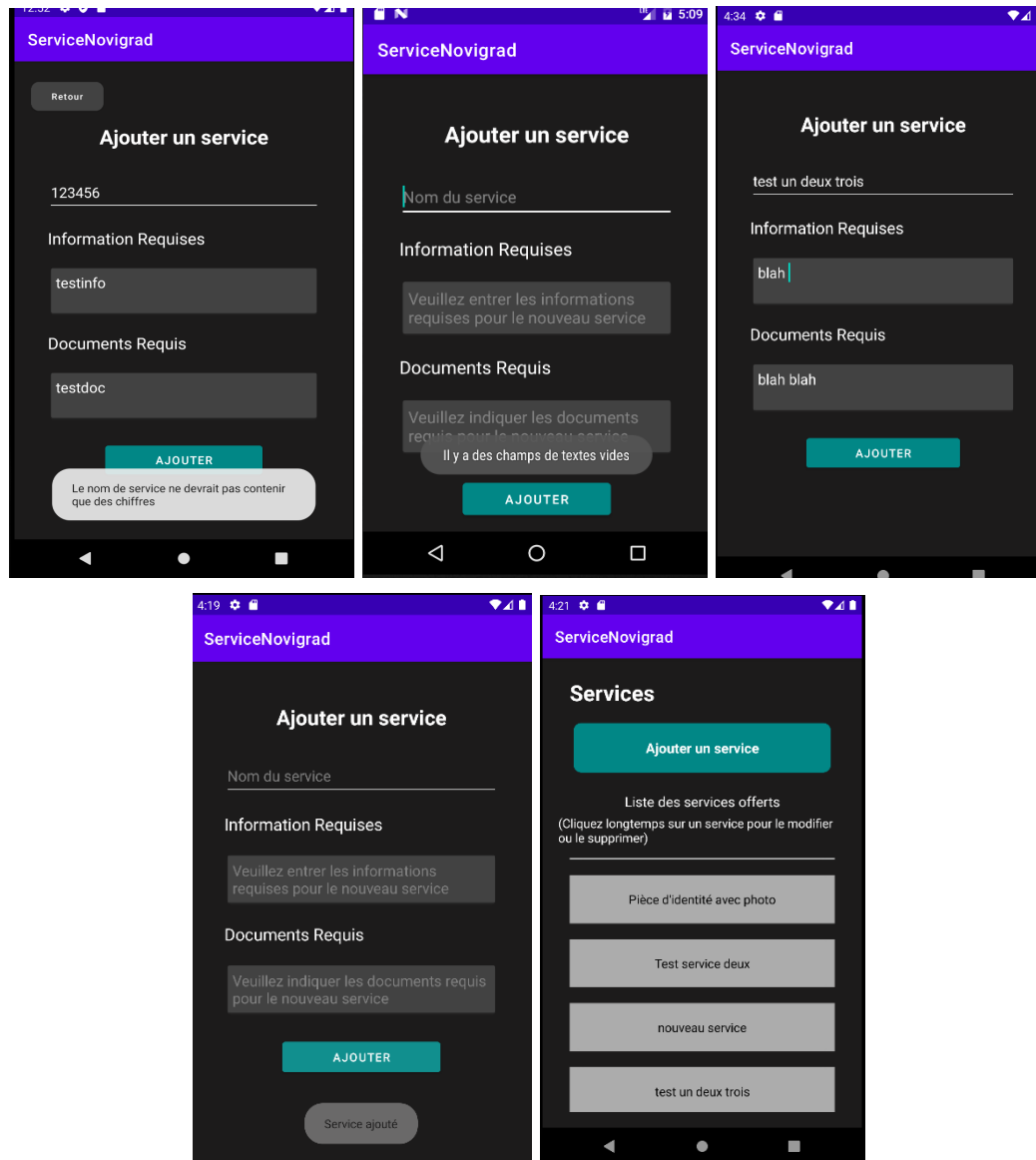


Figure 4: Ajout d'un service, avec messages d'erreur, message de succès, et finalement démonstration du service ajouté.

- Modification/Suppression d'un service

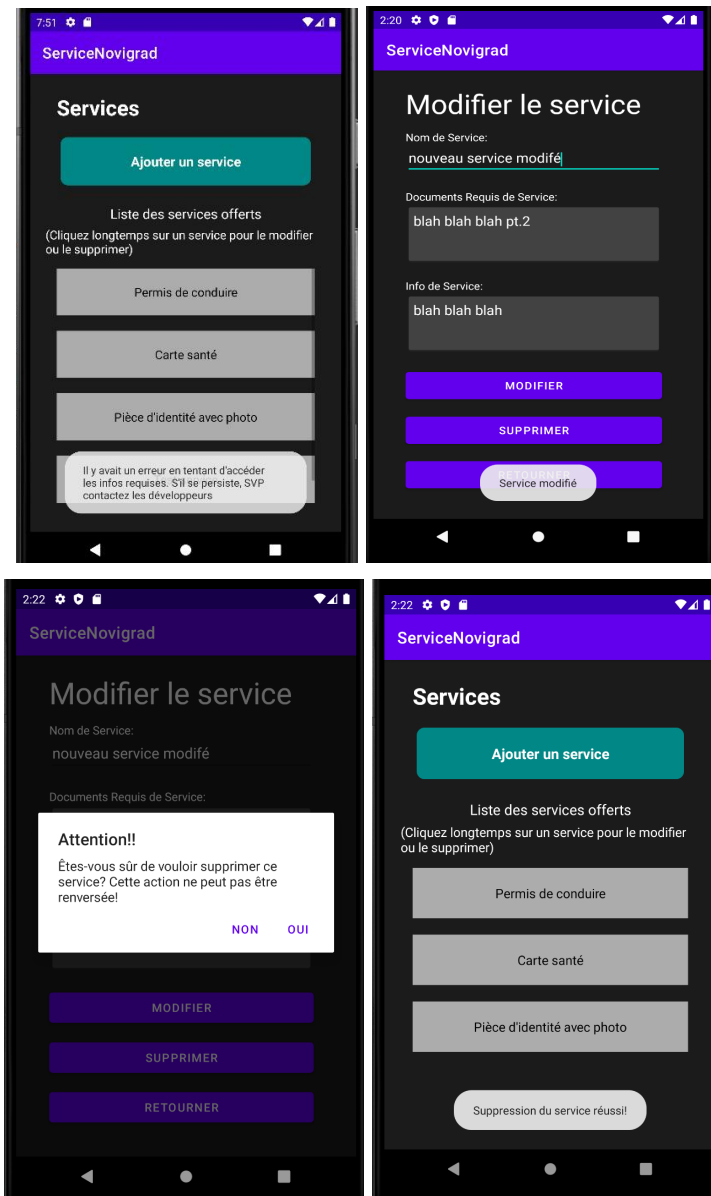


Figure 5: Modification et suppression d'un service. Le premier message d'erreur se produit quand il y a un faute de base de données.

- Suppression des comptes

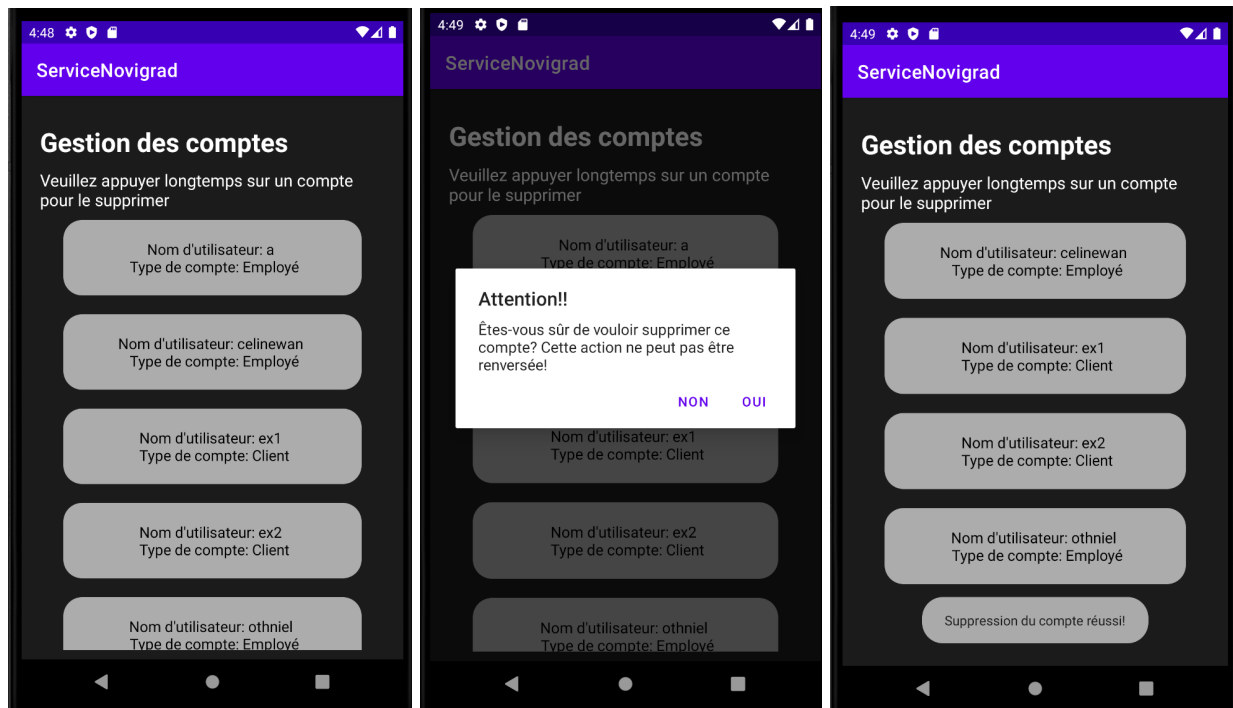


Figure 6: Suppression du premier compte de la liste

3. Backend de l'application

Céline et Tisham ont travaillé sur le backend qui s'assure que les services sont bien ajoutés, modifiés et supprimés dans la base de données, ainsi que la suppression des comptes. Nous avons utilisé la même base de données Firebase qui nous permet de garder en mémoire les comptes créés dans l'application afin de garder en mémoire les services aussi.

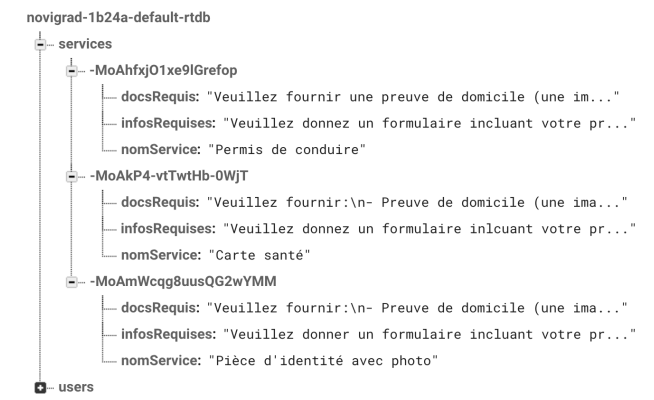


Figure 7: Aperçu de notre base de données après ajout des 3 services: Permis de conduire, Carte santé et Pièce d'identité avec photo

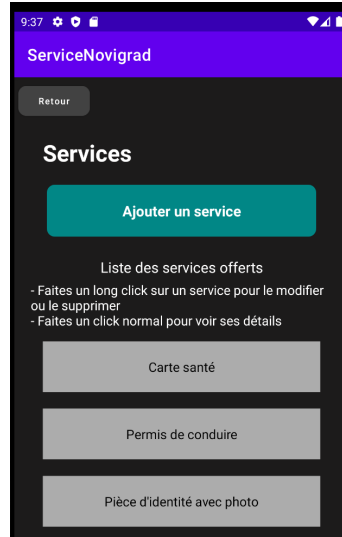


Figure 8: Aperçu de l’affichage de donnée dans l'application

4. Navigation sur l’application

Céline a ajouté des boutons retour afin de pouvoir mieux naviguer sur l’interface usager de l’application, ainsi que la fonctionnalité de pouvoir se déconnecter du compte. En effet, simplement utiliser la flèche retour dans le menu en bas du téléphone n’est pas tout le temps efficace car par moment, il nous ramène à la mauvaise activité. Ainsi, nous avons implémenté des fonctions *onReturn()* qui contient la méthode *finish()* afin de faciliter la navigation. Pour se déconnecter du compte, nous avons implémenté un fonction *onLogout()* qui permet de réinitialiser les champs de texte *username* et *password* lorsque nous revenons au LoginPage. (Avant cette implémentation, lorsque nous cliquons sur déconnecter, nous étions en effet dirigé vers le Login Page, mais les informations du compte (username et password) était toujours dans les champs de texte)

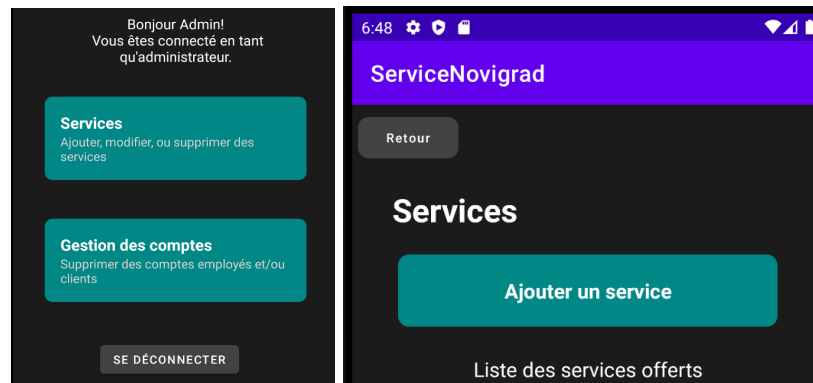


Figure 9: Boutons de déconnexion et “Retour”

Veillez noter que les boutons “Retour” ne figurent pas sur certaines captures d’écrans dans ce rapport car certaines fonctionnalités ont été implémentées et prises en photo avant l’ajout des boutons.

5. Test unitaires

Nous avons écrit 7 tests unitaires au total. Othniel a écrit les tests unitaires qui vérifient la création des comptes administrateur, employé et client. Evan a écrit les tests unitaires pour la création de service.

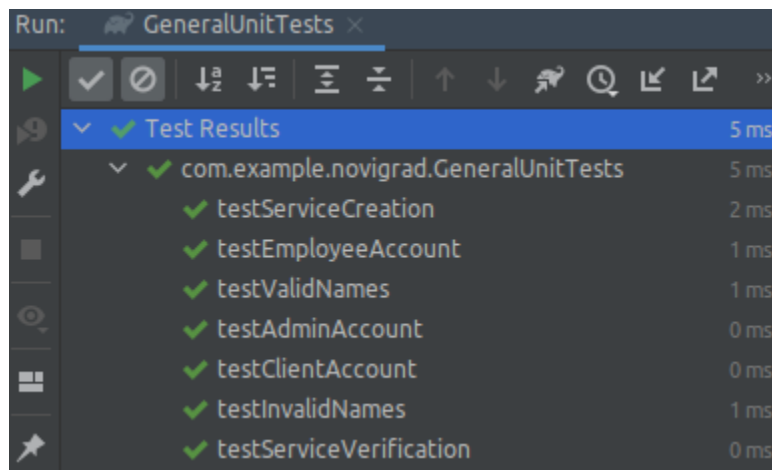


Figure 10: Résultat des test unitaires GeneralUnitTests (Couvre la validation des comptes et les services)

Nous avons essayé d’intégrer CircleCI dans notre répertoire. Cependant, cela était sans succès. Au début, nous recevons des erreurs au niveau “Restoring cache” et “Download Dependencies”.

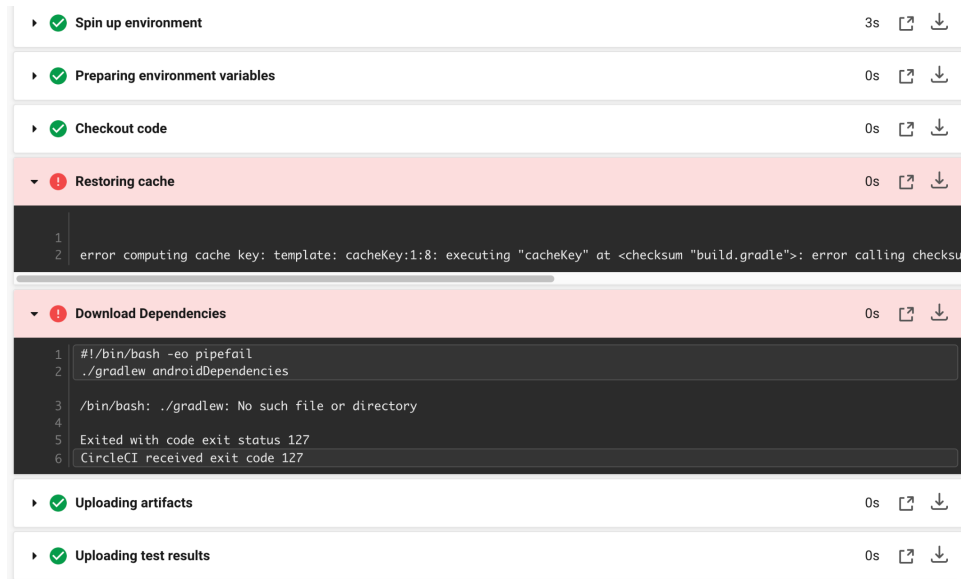


Figure 12: Erreurs reçues

Nous avons réalisé que cela était causé par le fait que notre application se trouvait dans un fichier “App” du répertoire. En effet, pour que cela fonctionne, notre application doit se trouver dans le fichier principal du répertoire et non dans un fichier “App”. Nous avons créé une branche afin de ne pas casser la branche main en cas d’échec.

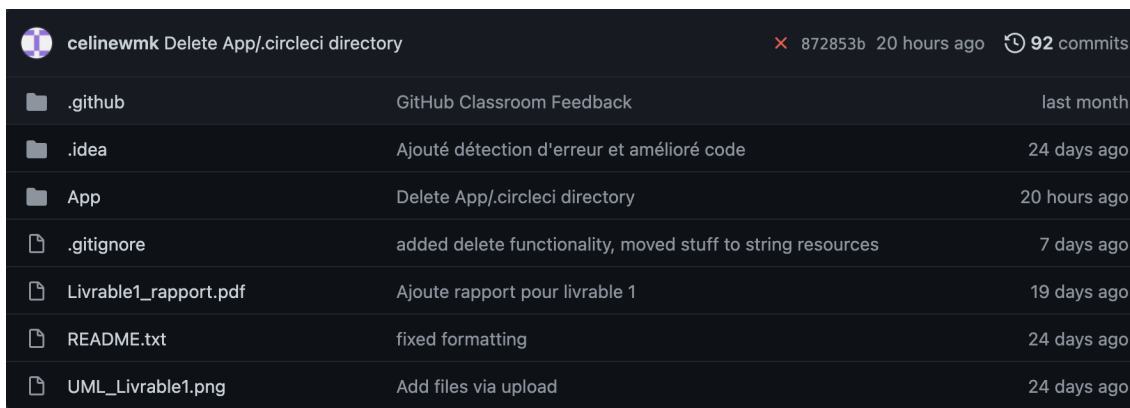


Figure 13: Aperçu du répertoire avant modification

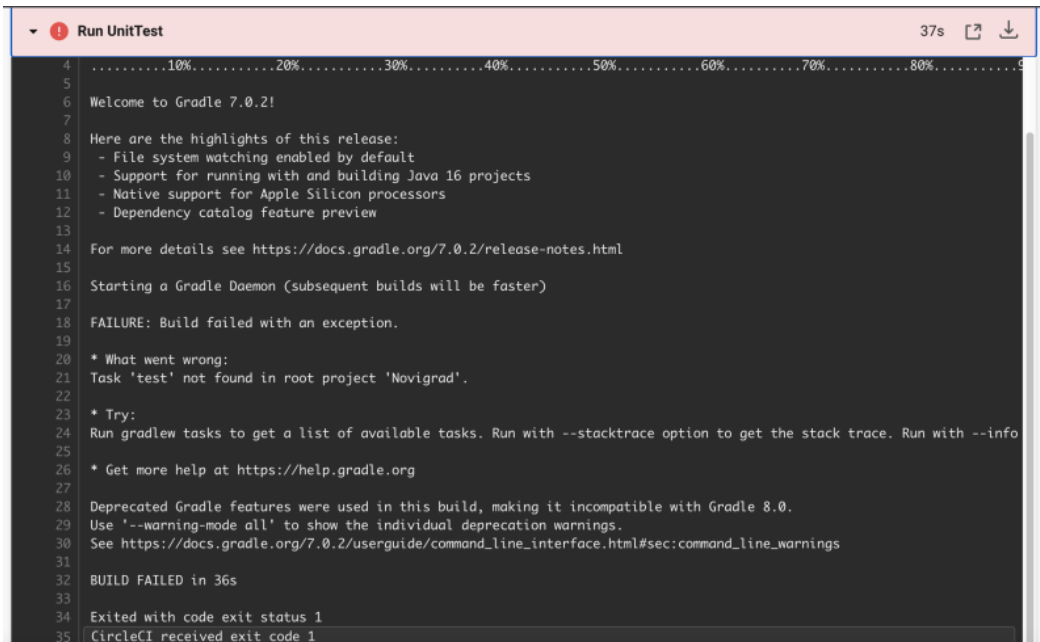
celinewmk Updated config.yml	c8c9771 22 minutes ago	103 commits
.circleci	Updated config.yml	22 minutes ago
.github	GitHub Classroom Feedback	last month
.gradle	test set up	35 minutes ago
.idea	testing set up	26 minutes ago
App	circleci set up test	1 hour ago
app	circleci set up test	1 hour ago
gradle/wrapper	circleci set up test	1 hour ago
.gitignore	added delete functionality, moved stuff to string resources	7 days ago
Livable1_rapport.pdf	Ajoute rapport pour livrable 1	19 days ago
README.txt	fixed formatting	24 days ago
UML_Livrable1.png	Add files via upload	24 days ago
build.gradle	circleci set up test	1 hour ago
gradle.properties	circleci set up test	1 hour ago
gradlew	circleci set up test	1 hour ago
gradlew.bat	circleci set up test	1 hour ago
local.properties	circleci set up test	1 hour ago
settings.gradle	circleci set up test	1 hour ago

Figure 14: Aperçu du répertoire après modification

Cependant, nous rencontrons maintenant des erreurs au niveau du “Run UnitTest” qu’on ne comprenait pas la raison. Ainsi, par manque de temps, nous avons décidé de laisser tomber l’intégration CircleCI pour le moment.

Spin up environment	28s	📄	⬇
Preparing environment variables	0s	📄	⬇
Checkout code	0s	📄	⬇
Restoring cache	7s	📄	⬇
Chmod permissions	0s	📄	⬇
gradle dependencies	11s	📄	⬇
Saving cache	0s	📄	⬇
Run UnitTest	37s	📄	⬇
Uploading artifacts	0s	📄	⬇
Uploading test results	0s	📄	⬇

Figure 15: Erreur reçu



```
4 .....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%
5
6 Welcome to Gradle 7.0.2!
7
8 Here are the highlights of this release:
9   - File system watching enabled by default
10  - Support for running with and building Java 16 projects
11  - Native support for Apple Silicon processors
12  - Dependency catalog feature preview
13
14 For more details see https://docs.gradle.org/7.0.2/release-notes.html
15
16 Starting a Gradle Daemon (subsequent builds will be faster)
17
18 FAILURE: Build failed with an exception.
19
20 * What went wrong:
21 Task 'test' not found in root project 'Novigrad'.
22
23 * Try:
24 Run gradlew tasks to get a list of available tasks. Run with --stacktrace option to get the stack trace. Run with --info
25
26 * Get more help at https://help.gradle.org
27
28 Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.
29 Use '--warning-mode all' to show the individual deprecation warnings.
30 See https://docs.gradle.org/7.0.2/userguide/command_line_interface.html#sec:command_line_warnings
31
32 BUILD FAILED in 36s
33
34 Exited with code exit status 1
35 CircleCI received exit code 1
```

Figure 16: Message erreur

Nous avons remarqué que le projet continue à build à chaque fois qu’on push des commits. Cependant, nous n’arrivons pas à arrêter cela car nous avons besoin de la permission de l’administrateur. (Si c’est possible de l’arrêter pour nous, ça serait apprécié)

Stop Building

If you choose to stop building **novigrad-project---part-1-g03** on CircleCI, we will halt all ongoing builds, and any teammates who are currently following the project will automatically `unfollow` it.

Stop Building


 You must have admin access to stop building this project.

Figure 17: Message pour le “Stop building”

6. Ligne de temps de développement

- 06/11/2021: Évaluation des besoins du livrable 2 + Continuation du développement du diagramme UML + Développement front-end
 - Tisham, Evan, Samy et Céline ont contribué au diagramme UML
 - Céline et Samy ont commencé à implémenter les nouvelles activités nécessaires pour le livrable 2
- 10/11/21: Suite du codage du front-end et début du codage du backend
 - Samy a apporté quelques modifications au design de la page `activity_add_service.xml`, particulièrement l'ajout de messages Toasts dans les cas où l'utilisateur n'entre pas des bonnes données.
 - Céline a commencé à coder le backend de l'application qui permet d'ajouter des services à notre base de données Firebase et a ajouté la classe Service
- 11/11/21: Continuation du front-end + backend
 - Céline a implémenté le code qui affiche la liste des services pris depuis la base de données Firebase (Ce qu'il reste à implémenter pour les services: capacité de modifier ou supprimer un service)
- 13/11/21: Développement des autres fonctions requises
 - Tisham a implémenté la modification et la suppression des services sur une nouvelle page, et a implémenté la fonctionnalité pour voir et supprimer les comptes employés et clients.
- 16/11/21: Navigation
 - Céline a implémenté des boutons afin de faciliter la navigation sur l'application
- 17/11/21: Test unitaires
 - Othniel a écrit des tests unitaires pour vérifier la création de compte
- 18/11/21: Finition de l'application + Test unitaires
 - Nous avons terminé le code pour l'interface usager pour notre application.
- 19/11/21: UML + tests unitaires
 - Céline a finalisé le diagramme UML, Evan a finalisé les tests unitaires, Samy a mis à jour des messages Toast
- 20/11/21: Finalisation du rapport et soumission
 - Finalisation des tests unitaires et débogage par Tisham, Evan et Céline.

7. Conclusion

Dans ce livrable nous avons poursuivi le développement de notre application pour le Service Novigrad qui offre des services à ses clients. Nous avons cette fois-ci implémenté les fonctionnalités du compte administrateur de l'application. L'administrateur peut maintenant ajouter, modifier et supprimer des services, ainsi que supprimer des comptes employés et clients. Grâce à Firebase, notre application garde maintenant en mémoire tous les comptes qui sont créés dans l'application et la liste de services offerts par les succursales. Pour ce livrable, nous avons également écrit des cas de tests unitaires afin de vérifier si notre code ne contient pas d'erreur en utilisant les connaissances acquises au cours des laboratoires 7 et 8. Nous avons éprouvé des difficultés avec l'intégration de CircleCI et nous ne sommes pas parvenus à écrire un test unitaire concernant le login, nous le ferons à l'avenir. Ainsi nous avons inclus juste des test unitaires d'autres classes afin de remplir les critères du livrable qui demande au moins 5 test unitaires locaux.