



Université de la Nouvelle Calédonie

OPT

Rapport de projet tuteuré

Application Domaine-nc en JavaFX

---

*Etudiants :*

Soer Irwan, L4 Informatique TREC 7  
Petit Kevin, L4 Informatique TREC 7  
Domergue David, L4 Informatique TREC 7

*Tuteur :*

Sales Adrien, Chef de bureau/Division Manager, OPT

# Table des matières

## **1. Introduction**

## **2. Présentation de l'OPT**

## **3. Technologie utilisée**

- A. Langage de programmation
- B. Outils utilisés

## **4. Différentes phases du projet**

- A. Création de l'interface
- B. Créations des classes et méthodes utilisées
- C. Optimisation et Difficultés

## **5. Travail restant**

## **6. Conclusion**

## 1- Introduction

Dans le cadre de notre UE Projet Tutoré du semestre 7 de la formation Licence Informatique Trec 7, nous avons effectué un projet tuteuré sous le tutorat de Mr Adrien Sales, chef de bureau chez OPT.

L'objectif de ce projet est la réalisation d'une application desktop ([dépôt Github du projet](#)) permettant de consulter les noms de DOMAINE.nc en utilisant son API publique mise à disposition sur [le marketplace d'APIs de l'OPT](#). Ce projet est une extension d'un autre projet appelé : « [domaine-nc-mobile](#) » qui est une application mobile open source développée en mode Open Innovation et qui, via la consommation d'API permet de chercher des noms de domaine en « .nc », visualiser les détails d'un nom de domaine particulier et d'enregistrer aisément la date d'expiration d'un nom de domaine comme reminder dans le calendrier du choix de l'utilisateur.

## 2- Présentation de l'OPT

L'OPT-NC est un établissement public à caractère industriel et commercial (EPIC) depuis 1958, transféré à la Nouvelle-Calédonie en 2003. Il assure dans un cadre réglementaire dépendant de l'État (code monétaire et financier) les services financiers et pour le compte de la Nouvelle-Calédonie, l'offre et la fourniture des services publics et complémentaires du domaine des télécommunications et du postal encadrés par le code des postes et télécommunications de Nouvelle-Calédonie. L'OPT opère dans 4 principaux secteurs d'activité :

- La télécommunication : conception, exploitation et maintenance des réseaux et des accès aux services de télécommunication.
- Les services de courriers/colis : acheminement et distribution des lettres, paquet et colis en Nouvelle Calédonie et a l'internationale
- Les services financiers : gestion des comptes courant, des comptes épargne et de leurs services associés
- La distribution : garantir l'accessibilité des services et produits essentiels dans les domaines postal, financier et télécoms dans les agences.

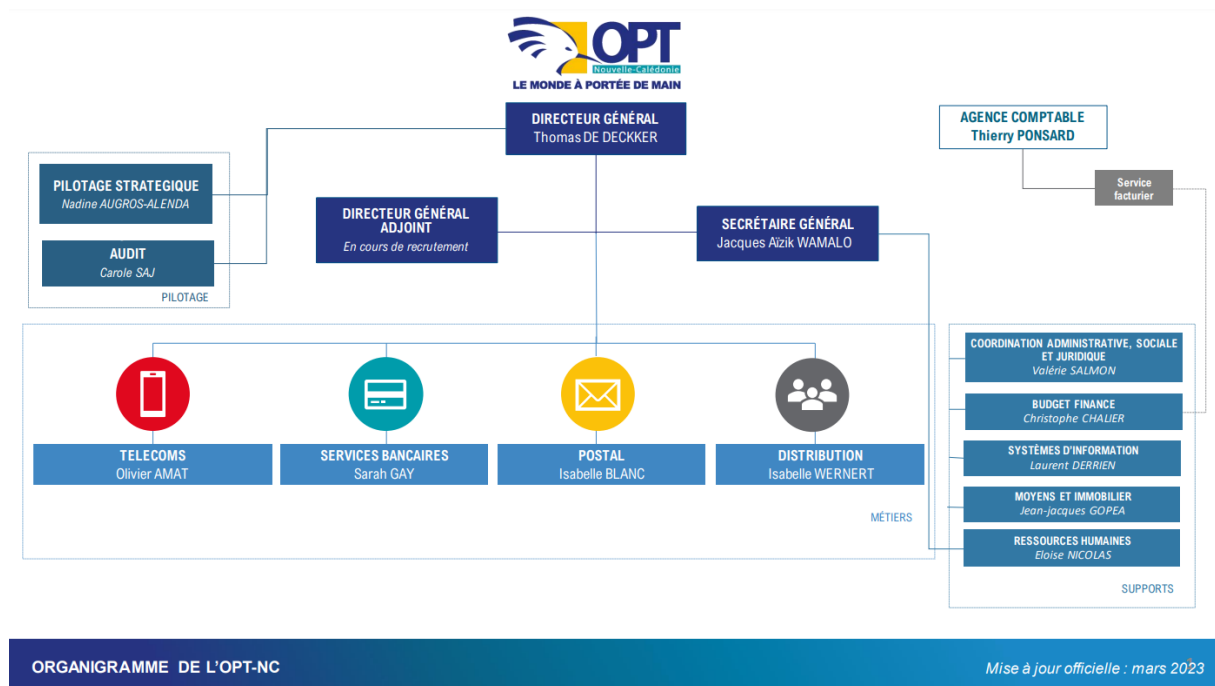


Figure 1: Organigramme de l'OPT

### 3- Technologie utilisée

#### a. Langage de programmation

Ce projet a été réalisé en Java dans sa version 17. Java est un langage de programmation orienté objet développé par Sun Microsystems en 1991, qui a ensuite été racheté par Oracle en 2009.

La syntaxe du Java se rapproche du C et C# mais contrairement à ces langages, Java offre plusieurs avantages, notamment la portabilité. Etant un langage interprété, l'utilisation de Java nécessite un JRE (Java Runtime Environment) pour lancer les programmes. Le JDK se chargera de compiler le programme et par conséquent il n'est pas nécessaire de devoir recompiler son programme d'un système à un autre, il suffit juste que chacun de ces systèmes possède sa propre machine virtuelle Java. Le compilateur Java produira du bytecode (langage binaire intermédiaire) qui est indépendant de toute architecture matérielle et de tout système d'exploitation. La machine virtuelle Java se chargera ensuite de traduire le bytecode en code natif.

#### b. Outils utilisés

##### Appache Maven :

Maven est un outil de gestion et d'automatisation de production des projet logiciel. L'outil se charge notamment de gérer les librairies ainsi que de la compilation et le déploiement d'application java.

### **JavaFX :**

JavaFX est un framework et une bibliothèque d'interface utilisateur. Ce framework nous a permis de créer l'interface graphique pour notre application bureau mais il est également possible de créer des applications mobiles ou tablette.

### **Scene Builder :**

Scene Builder est un outil interactif de conception de fenêtre graphique pour JavaFX. L'application nous a permis d'avoir une visualisation de la fenêtre créée.

### **Git :**

Git est un logiciel de gestion de version décentralisé. Il est le système à la base du célèbre site Github. C'est grâce à git qu'on a pu effectuer des push, pull voir synchroniser notre projet.

### **Github :**

Github est un service web d'hébergement et de gestion de développement. Etant donné que sur ce projet nous travaillons parfois sur le même fichier mais sur des postes différents, nous mettons alors en commun nos modifications grâce au service de partage de Github.

### **Discord :**

Discord est une application permettant de communiquer en distancielle. Cette application était le principal moyen de pouvoir communiquer au sein de l'équipe en dehors de l'université. Un serveur discord dédié au projet a également été ouvert par Sales avec un channel notifiant les personnes lorsqu'une activité a été faite sur Github (pull, commentaire ...)

## **4- Différentes phases du projet**

### **a. Création de l'interface**

Pour la création de l'interface nous avons pensés à quelque chose de simple mais efficace. L'application sera composée d'une fenêtre principale. Sur cette fenêtre on pourra y trouver une zone de recherche ou l'utilisateur pourra entrer ce qu'il souhaite rechercher, ainsi qu'une zone vide dédié à l'affichage des résultats trouvés.

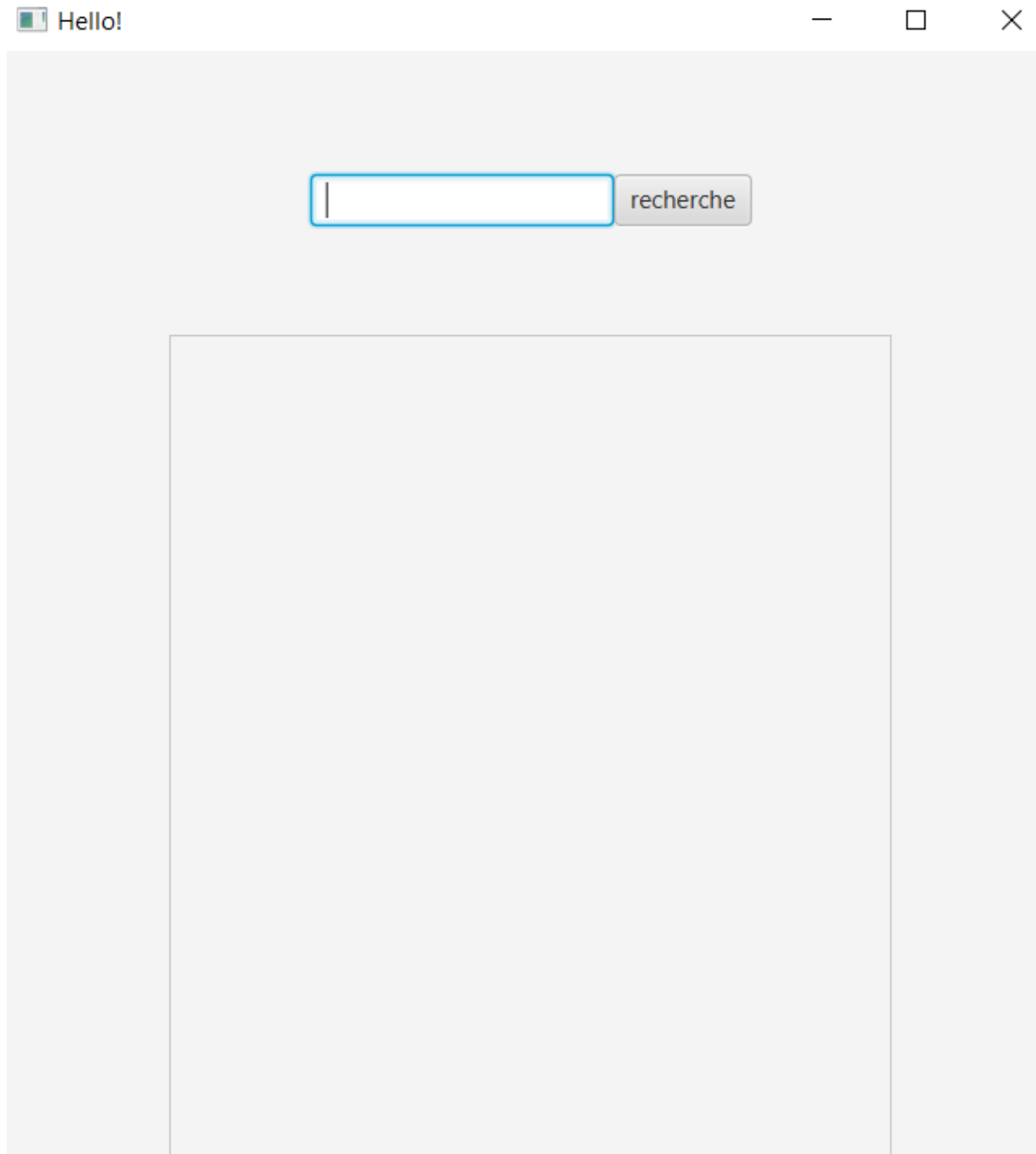
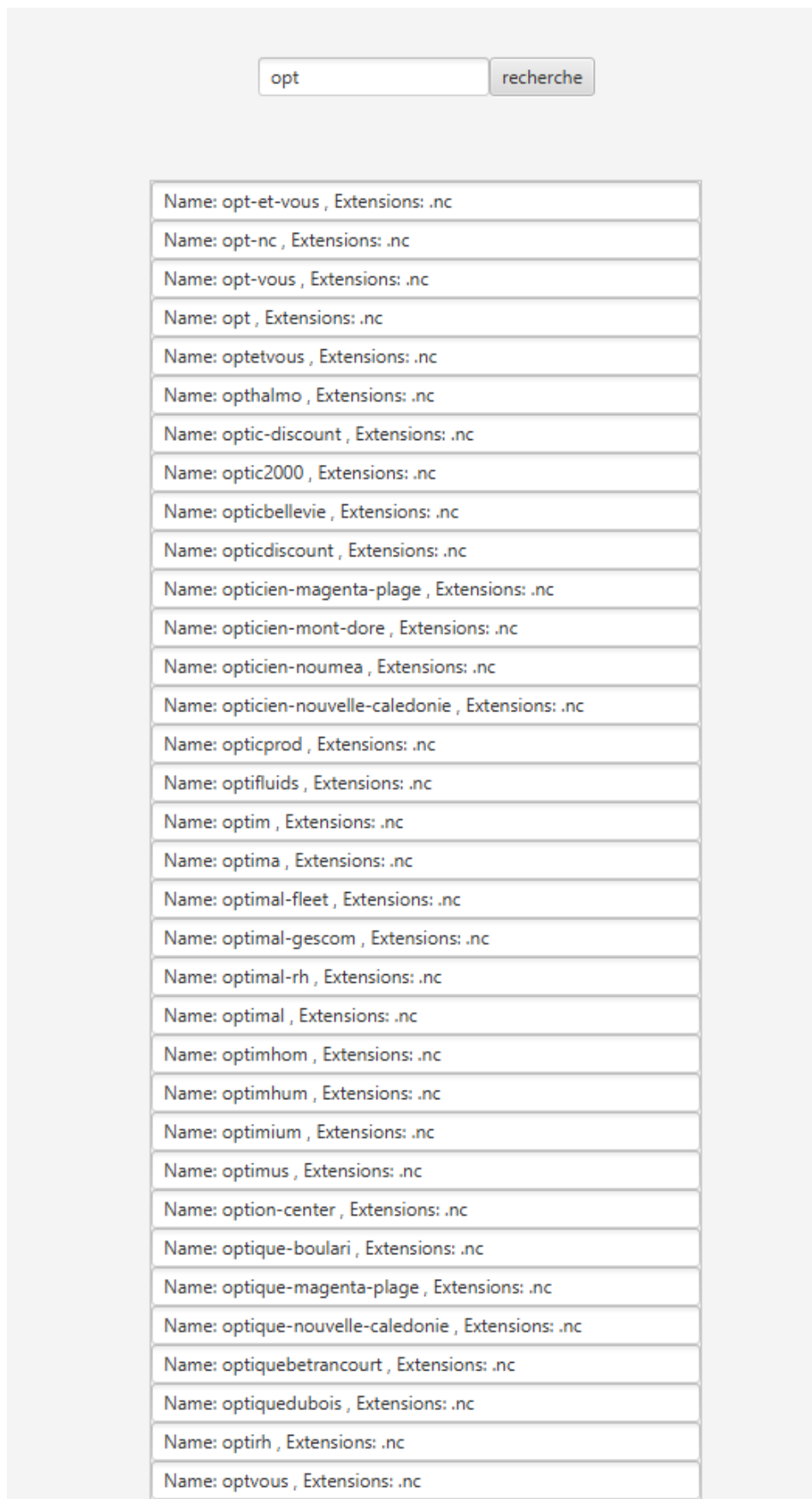


Figure 2: Fenêtre principale de l'application V1

Pour générer cette fenêtre nous avons commencé par placer un « AnchorPane » qui est la base de notre fenêtre. Dans cette fenêtre nous plaçons d'abord la barre de recherche. Pour cela nous avons placé un « HBOX » afin d'afficher les informations à l'horizontale et dans cette HBOX nous avons placé un « TextField » ainsi qu'un bouton. Le TextField permettra à l'utilisateur de rentrer le nom de domaine qu'il souhaite rechercher et le bouton servira à initialiser la recherche.

Le rectangle vide sous la barre de recherche a été créé en plaçant un « ScrollPane » qui fait apparaître un scroll bar afin de faire monter ou descendre le contenu de la fenêtre si nécessaire. Dans ce ScrollPane on a inséré une « VBox » qui permettra d'afficher les informations en colonne.



opt recherche

Name: opt-et-vous , Extensions: .nc
Name: opt-nc , Extensions: .nc
Name: opt-vous , Extensions: .nc
Name: opt , Extensions: .nc
Name: optetvous , Extensions: .nc
Name: ophthalmo , Extensions: .nc
Name: optic-discount , Extensions: .nc
Name: optic2000 , Extensions: .nc
Name: opticbellevie , Extensions: .nc
Name: opticdiscount , Extensions: .nc
Name: opticien-magenta-plage , Extensions: .nc
Name: opticien-mont-dore , Extensions: .nc
Name: opticien-noumea , Extensions: .nc
Name: opticien-nouvelle-caledonie , Extensions: .nc
Name: opticprod , Extensions: .nc
Name: optifluids , Extensions: .nc
Name: optim , Extensions: .nc
Name: optima , Extensions: .nc
Name: optimal-fleet , Extensions: .nc
Name: optimal-gescom , Extensions: .nc
Name: optimal-rh , Extensions: .nc
Name: optimal , Extensions: .nc
Name: optimhom , Extensions: .nc
Name: optimhum , Extensions: .nc
Name: optimum , Extensions: .nc
Name: optimus , Extensions: .nc
Name: option-center , Extensions: .nc
Name: optique-boulari , Extensions: .nc
Name: optique-magenta-plage , Extensions: .nc
Name: optique-nouvelle-caledonie , Extensions: .nc
Name: optiquebetrancourt , Extensions: .nc
Name: optiquedubois , Extensions: .nc
Name: optirh , Extensions: .nc
Name: optvous , Extensions: .nc

Figure 3: Recherche OPT



Une fois que les résultats sont affichés, on veut que l'utilisateur puisse cliquer sur le résultat de son choix afin d'obtenir plus de renseignement.

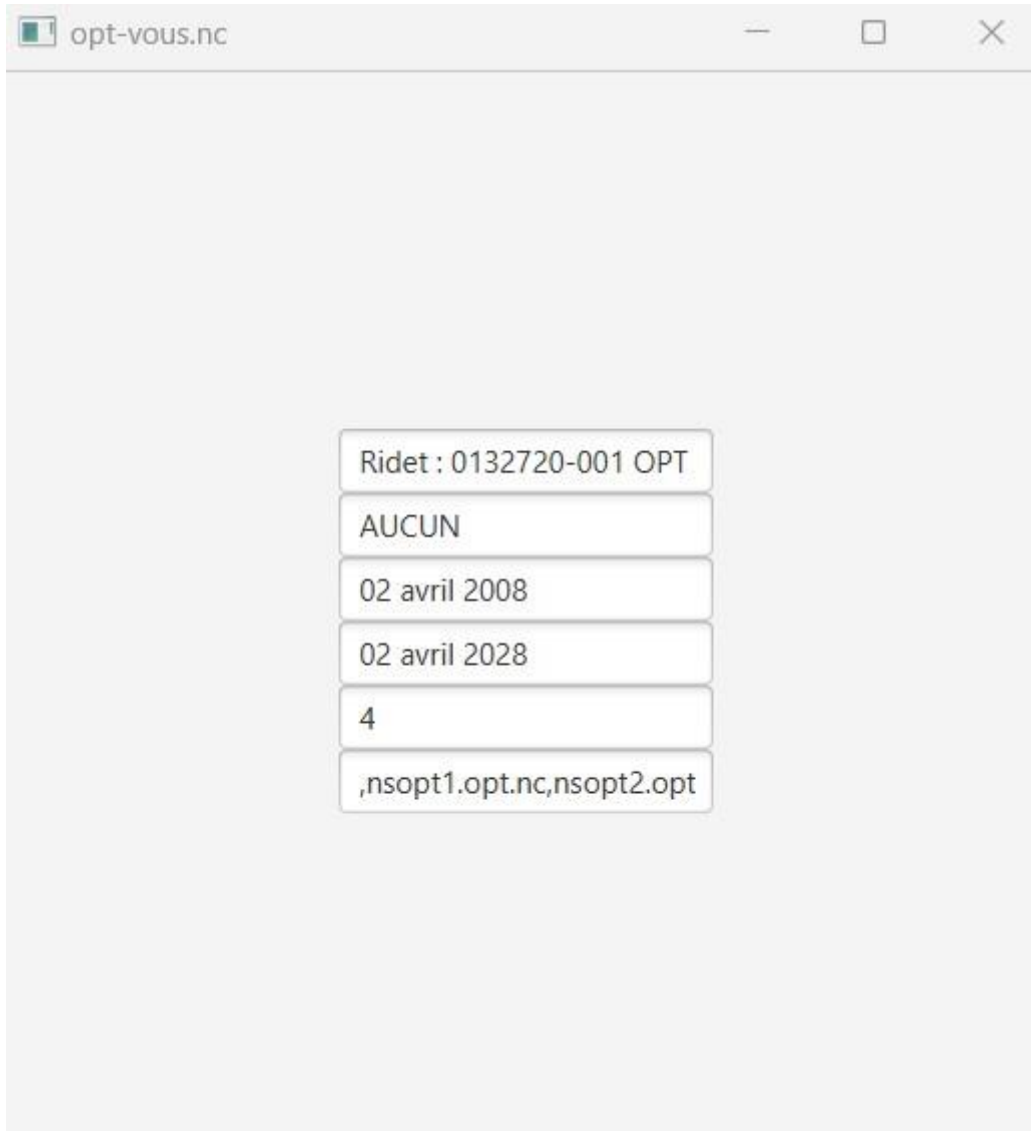


Figure 4: Exemple de fenêtre d'information V1

Afin de générer cette fenêtre nous avons dû créer tout d'abord un « AnchorPane » qui est la base de notre fenêtre encore une fois. Dans cette fenêtre nous avons inséré un « ScrollPane » qui fait apparaître un scroll bar afin de faire monter ou descendre le contenu de la fenêtre si nécessaire. Enfin nous avons ajouté une « VBox » avec un alignement centré afin que les informations apparaissent en colonne et au centre de notre fenêtre.

## **b. Création des classes et méthodes utilisées**

Notre application se décompose en 1 classe main +6 autres classes Java : 3 classes concernant l'API ainsi que 3 classes se chargeant de faire tourner l'application. Tout d'abord nous avons la classe `DomaineNcApp` qui est en quelque sorte notre launcher. Cette classe se chargera de créer notre fenêtre initiale et de la maintenir actif.

Ensuite nous avons la classe `DomaineNcController` qui sert à la fois à relier les éléments de notre interface graphique de notre fenêtre principale avec nos méthodes (par exemple l'exécution d'une action lorsque l'utilisateur clique sur le bouton « Recherche », la méthode de recherche...) mais qui sert également à récupérer et placer les informations dans la « boîte » situé en dessous de notre zone de recherche.

La classe `DomaineNcInfoController` se charge de récupérer et d'afficher les informations dans la fenêtre secondaire lorsque l'utilisateur clique sur le nom recherché (nom, bénéficiaire, date de création...). Depuis cette fenêtre l'utilisateur pourra également cliquer sur un lien qui le mènera vers le site « `Domaine.nc` » contenant sa recherche actuelle. Si l'objet de sa recherche a un numéro de ridet, il peut choisir de cliquer sur le ridet, ce qui le mènera vers le site « `data.gouv.nc` » avec le numéro de ridet recherché.

Les informations à placer viennent de la classe `Request`. Cette classe a pour but de consommer l'API afin de trouver les données recherchées par l'utilisateur. La classe prend en compte que l'API utilise spring et par conséquent, on utilise des « `RestTemplate` » avec une méthode « `exchange` » pour personnaliser nos requêtes (on ajoute donc des headers de requête). Ensuite on utilise un « `ObjectMapper` de jackson » afin de sérialiser la réponse « `Json` » en objet simple d'utilisation. Pour récupérer la liste des domaines on utilise un objet de type liste. Cette classe crée une liste de `DomaineEntity`.

Puis nous avons la classe `DomaineEntity` qui sert à représenter un domaine. La classe contient le nom du domaine ainsi que son extension.

La classe `DomaineInfoEntity` contient les informations de la fenêtre secondaire (donc les informations relatives au domaine). Le fonctionnement de cette classe est similaire à celui de `DomaineEntity`.

La classe `main` n'a été créé uniquement afin de générer le `.jar` exécutable.

### **c. Optimisations et difficultés**

Bien que primitif, l'application était quasiment fonctionnelle mais beaucoup d'amélioration pouvait être apporté. L'une des premières améliorations apportées fut au niveau des fenêtres de l'application. Au début les fenêtre étaient configurés avec une certaine taille prédéfinie mais celle-ci ne s'adaptait pas selon la préférence de l'utilisateur. Nous avons donc fait en sorte que la fenêtre ainsi que son contenu s'adaptent à la taille choisit par l'utilisateur.

Ensuite viens la méthode de recherche ainsi que l'exécution de cette méthode. Au début, lors du clic sur le bouton recherche, notre méthode de recherche se contente simplement de vérifier si la chaine de caractère entrée dans notre barre de recherche existe dans le nom du nom de domaine peu importe son emplacement. Par exemple si je rentre la lettre « o » dans la barre de recherche et que je clique sur recherche, j'aurai comme résultat « opt » et « lagoon ». Dans le cas où on tente de lancer la recherche en laissant notre barre de recherche vide, l'application va afficher l'intégralité des noms de domaine. Trois améliorations ont été faite sur cette partie : la première était de faire en sorte que l'utilisateur ne soit plus obligé d'appuyer sur le bouton recherche pour exécuter la recherche. Désormais l'utilisateur peut tout simplement appuyer sur « Entrée » pour lancer la méthode de recherche. La deuxième optimisation un peu plus complexe était de faire en sorte que la recherche se fasse en fonction de l'ordre de la chaine de

caractère entré dans la barre de recherche. En reprenant l'exemple plus haut, désormais si l'utilisateur entre la lettre « o » et effectue la recherche, l'application trouvera « opt » mais elle ne trouvera plus « lagoon » étant donné que lagoon ne commence pas par un o. La troisième consiste à tout simplement bloquer le lancement de la recherche s'il y a déjà une recherche en cours (lien de l'issue : <https://github.com/adriens/domaine-nc-javafx/issues/8>).

Autre optimisation réalisée concerne l'utilisation de thread. Lors de la recherche notre application se gèle et l'utilisateur était obligé d'attendre que la recherche se termine avant de pouvoir interagir de nouveau avec l'application. Cela peut être assez gênant lorsque l'utilisateur entre une chaîne de caractère qui peut être associée à beaucoup de résultats possibles (le figuré 3 montré précédemment est un bon exemple). L'utilisation de thread permet de fluidifier l'application et de ne plus le geler lors de l'exécution de la recherche.

Pour pouvoir mieux repérer les potentiels problèmes lors du fonctionnement de notre application, nous avons décidé d'implémenter des « loggings » (ou log). Les logs sont une sorte de journal de bord qui décrit les événements qui se sont produits. Par exemple lorsqu'on lance notre application, on peut voir ces messages apparaître ([lien de l'issue](#)).

```
[20:51:33] (main) *INFO* com.unc.domainenc.DomaineNcApp - Lancement de l'application.
[20:51:33] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcApp - Recherche de la clef d'API dans l'environnement du systeme.
[20:51:33] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcApp - Clef d'API trouvee dans l'environnement du systeme.
[20:51:33] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcController - Recuperation des noms de domaines.
[20:51:37] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcController - Recherche op.
[20:51:42] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcController - Recherche opt.
[20:51:44] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcInfoController - Recuperation des information sur opt-et-vous.nc.
[20:51:46] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcApp - Ouverture de https://www.domaine.nc/whos?domain=opt-et-vous&ext=.nc
[20:51:50] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcApp - Ouverture de https://data.gouv.nc/explore/dataset/entreprises-actives-au-ridet/table?disjunctive.libelle_formjur&
```

Figure 5: log du lancement de l'application

Afin de pouvoir utiliser notre application, nous avons actuellement besoin d'une clé « RAPIDAPI ». Cette clé d'API est gratuite mais nécessite d'être authentifiée. Nous nous sommes donc authentifiés en utilisant notre compte Github puis l'avons récupéré et rangé dans un fichier « .env » en début de projet afin de pouvoir travailler. Mais si on souhaite partager notre fichier .jar à d'autres personnes afin qu'ils puissent utiliser l'application, il est très probable que ces personnes ne disposent pas de la clé API. Afin d'optimiser cela, il nous a été demandé de faire en sorte qu'au lancement de l'application, on fasse une vérification de l'existence (ou non) en mémoire de la variable « X\_RAPIDAPI\_KEY ». Si la variable existe alors on la prend et on l'utilise. Si la variable n'existe pas, dans ce cas on va la chercher dans le fichier « .env » et si ce fichier « .env » n'existe pas, alors on sort un autre message d'erreur.

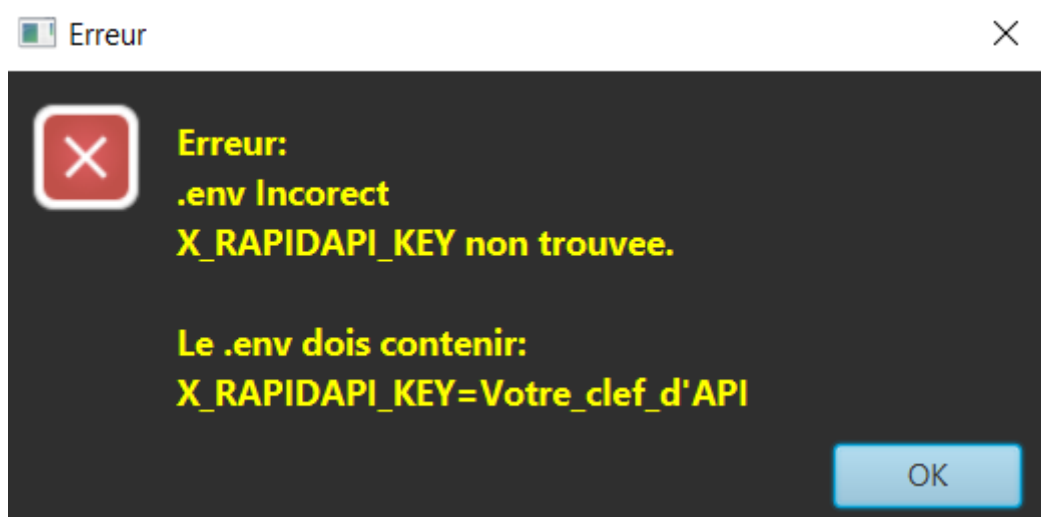


Figure 6: variable non existant

Comme nous pouvons le voir, dans cet exemple notre utilisateur semble posséder le fichier « .env » cependant, la variable « X\_RAPIDAPI\_KEY » est introuvable. Nous avons également fourni une petite explication de ce qu'est censé contenir le fichier. La console affiche également les logs d'erreurs.

```
[10:02:07] (main) *INFO* com.unc.domainenc.DomaineNcApp - Lancement de l'application.  
mai 28, 2023 10:02:07 AM com.sun.javafx.application.PlatformImpl startup  
WARNING: Unsupported JavaFX configuration: classes were loaded from 'unnamed module @5cee5251'  
[10:02:07] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcApp - Recherche de la clef d'API dans l'environnement du systeme.  
[10:02:07] (JavaFX Application Thread) *WARN* com.unc.domainenc.DomaineNcApp - Clef d'API non trouvee dans l'environnement du systeme.  
[10:02:07] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcApp - Recherche de la clef d'API dans le fichier .env.  
[10:02:07] (JavaFX Application Thread) *WARN* com.unc.domainenc.DomaineNcApp - Clef d'API non trouvee dans le fichier .env.
```

Figure 7: Logs console du figure 6

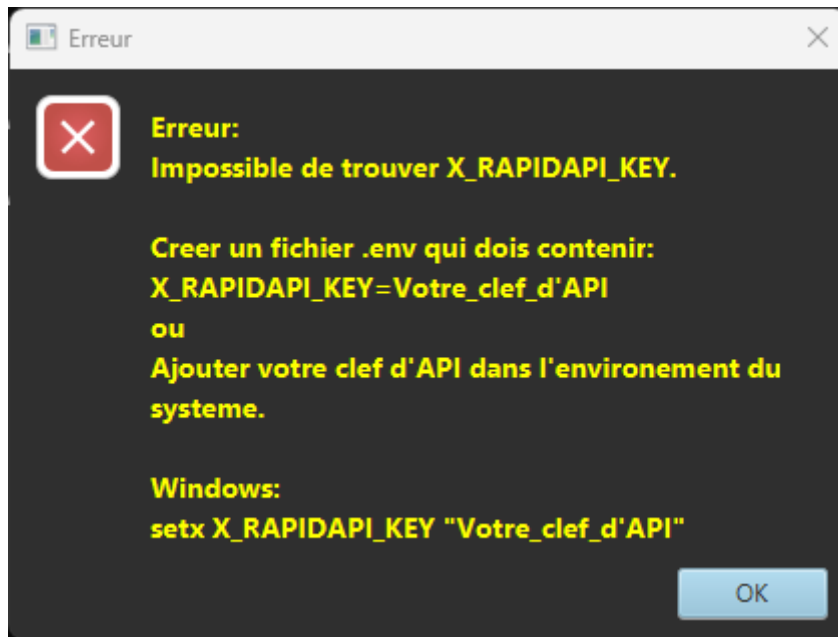


Figure 8: variable et fichier .env non existant

Ici, notre utilisateur ne possède ni la clé d'API en mémoire, ni le fichier « .env ». Nous laissons donc le choix à l'utilisateur sur la solution à apporter. Comme pour le message d'erreur précédent, nous avons mis les logs dans la console.

```
[10:21:29] (main) *INFO* com.unc.domainenc.DomaineNcApp - Lancement de l'application.  
mai 28, 2023 10:21:29 AM com.sun.javafx.application.PlatformImpl startup  
WARNING: Unsupported JavaFX configuration: classes were loaded from 'unnamed module @5cee5251'  
[10:21:29] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcApp - Recherche de la clef d'API dans l'environnement du systeme.  
[10:21:29] (JavaFX Application Thread) *WARN* com.unc.domainenc.DomaineNcApp - Clef d'API non trouvee dans l'environnement du systeme.  
[10:21:29] (JavaFX Application Thread) *INFO* com.unc.domainenc.DomaineNcApp - Recherche de la clef d'API dans le fichier .env.  
[10:21:29] (JavaFX Application Thread) *WARN* com.unc.domainenc.DomaineNcApp - Fichier .env introuvable.
```

Figure 9: logs du figuré 8

Notre message d'erreur prend en compte le système d'exploitation de la machine qui fait tourner le programme. Dans les screenshots montrés plus haut, nous sommes sur Windows donc nos messages

d'erreur sont des messages d'erreur spécifique à Windows mais admettons que nous travaillons sur Linux et qu'il nous manque notre clé API, nous aurions donc reçu le message suivant :

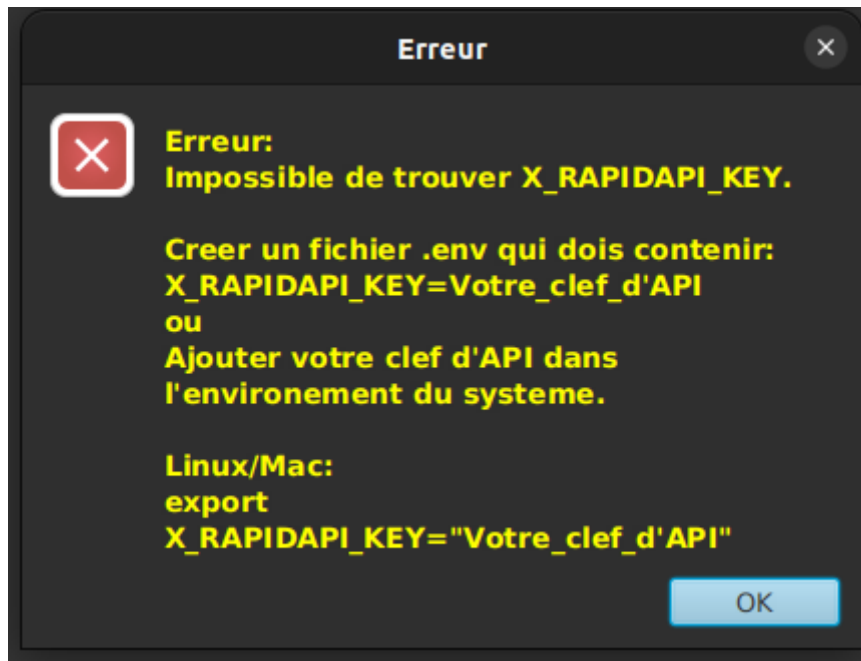


Figure 10: Fenêtre d'erreur Linux

([Lien de l'issue](#))

Enfin, nous avons travaillé sur l'apparence de l'application afin qu'il soit visuellement identique à celle de « domaine-nc-mobile ».

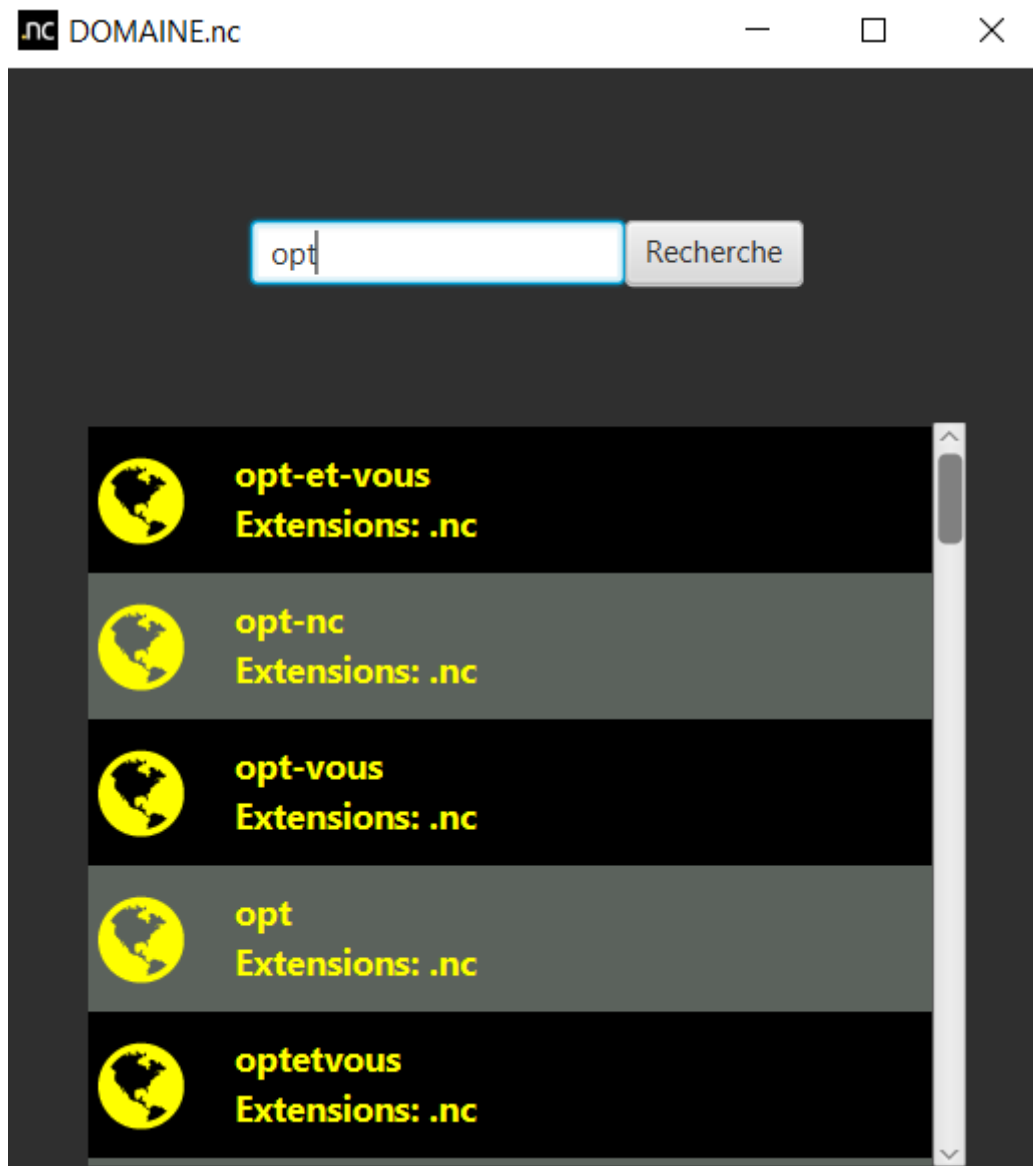


Figure 11:fenêtre de recherche version finale

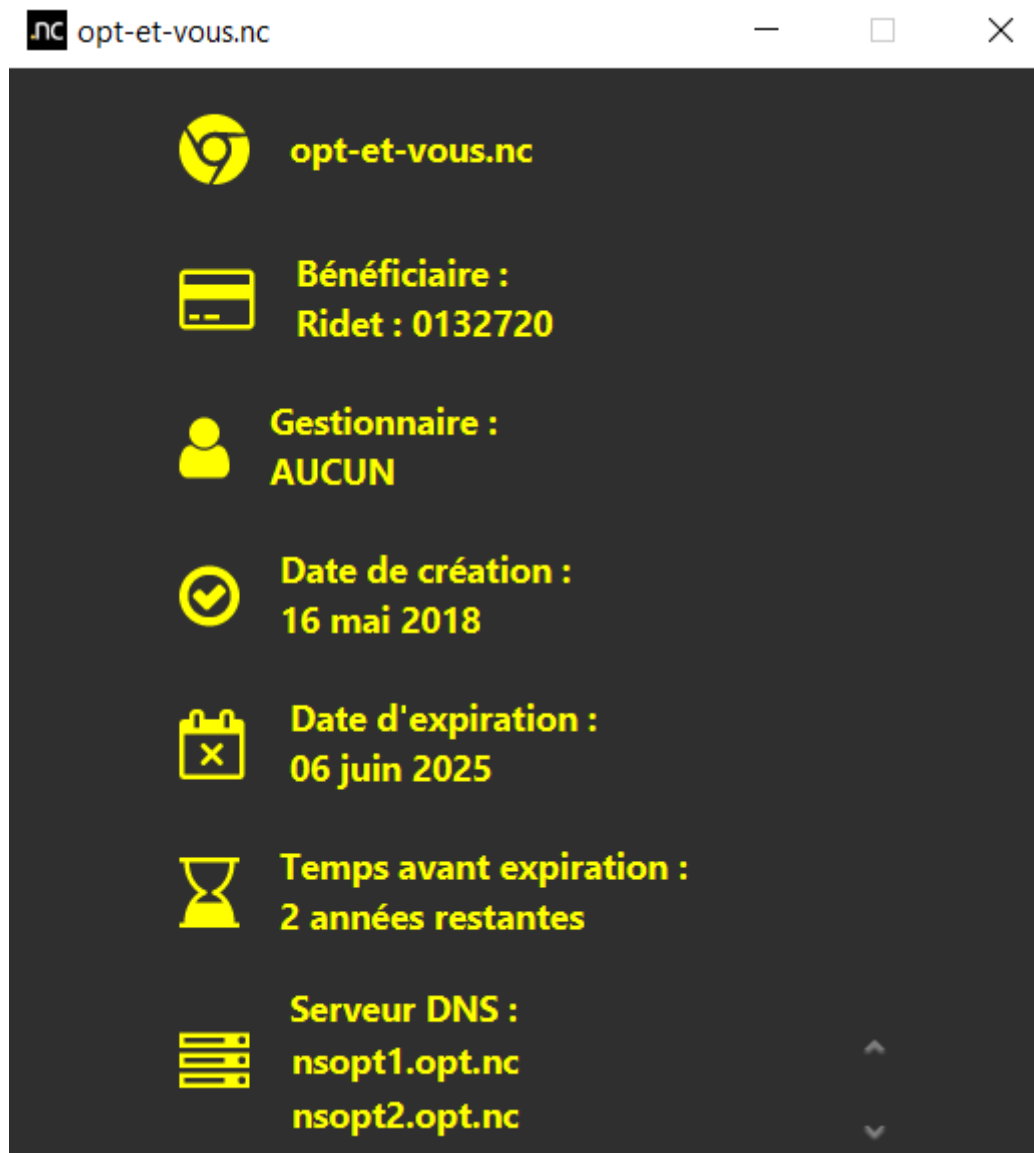


Figure 12:fenêtre d'information version finale

([lien de l'issue](#))

## 5- Travail restant

L'application étant quasiment terminée, il nous reste à préparer la release pour les utilisateurs non développeur. Cette tâche comprend de respecter les conventions de nommage de tag (par exemple v1.0.0) ainsi que de générer un changelog en automatique afin de documenter les changements/features des versions.

Une autre petite modification peut être faite au niveau du comportement de la souris lorsque l'on survole le lien ou le numéro de ridet de la fenêtre secondaire. L'icône pourrait se transformer en une main afin de montrer à l'utilisateur qu'il est possible d'ouvrir les liens ([lien du pull](#)).



Enfin il nous reste à livrer un .jar pour les utilisateurs non développeur. Ce .jar devra être prêt à l'emploi mais également taggé avec un numéro de version ([lien de l'issue](#)).

## 6- Conclusion

En conclusion, malgré un départ difficile en début de projet principalement lié à notre manque d'expérience concernant JavaFX et la consommation d'une API, nous avons réussi à créer une application assez basique qui réponds à notre sujet de stage. Cette application a ensuite été sujet à de nombreuses optimisations afin de le rendre plus performant, plus agréable à utiliser et le tout en suivant le thème de l'application mobile.

Ce projet nous a permis de travailler et d'améliorer nos compétences de coopération ainsi que de travail à distance étant donné que tout le travail s'est réalisé en distanciel. Au cours du projet nous avons élu un lead dev et créé une sorte de hiérarchie au sein de notre équipe. Toutes nos tâches étaient réparties sur Github à travers plusieurs issues ou le lead dev se chargeait d'assigner la tâche à un membre de l'équipe et vérifier que le résultat correspond à la tâche demandée. Une fois fini, le lead dev ou parfois le membre assigné à la tâche se chargeait également de fermer les issues. Nous avons également pu améliorer notre maîtrise de Github, qui est de nos jours indispensable dans la vie d'un développeur.

Nous tenons enfin à remercier notre tuteur pour son accompagnement et le temps investi tout au long de ce projet. Ce fut une expérience enrichissante.