Table des matières

1. Introduction

Chaque année, plus de 100’000 morts sont causées et près de 400'000 victimes se retrouvent assujetties à une forme d’invalidité par suite d’une morsure de serpent venimeux. En effet, avec environ 3’700 espèces de serpents dont 650 venimeuses il est difficile de déterminer la morsure d’un serpent à première vue. De plus, des présomptions erronées quant à l’identification d’un serpent mènent à l’application du mauvais anti-venin sur la victime ce qui est souvent coûteux financièrement parlant et peut avoir des effets de bord sur la personne mordue [1].

Il existe différents types de venins. Chacun cause des effets distincts et agit sur une durée distincte. Les neurotoxines par exemple provoquent un arrêt respiratoire dans les environs des 30 premières minutes aux 8 heures après la morsure. D’autres venins tels que les cardiotoxiques mènent à un arrêt cardiaque et agissent dans les 2 aux 12h heures qui suivent la morsure. Le venin cytotoxique quant à lui cause des nécroses ou la gangrène et les effets agissent sur le corps entre 3 jours et 1 mois après morsure sur la victime [2].

En Afrique, en raison du manque d’anti-venin dans les institutions de santé, seulement 10% des personnes victimes de morsure de serpent peuvent être soignées. Le faible approvisionnement d’anti-venin dans les institutions de santé publique est notamment causé par le manque de récolte de données concernant la faune des serpents mais aussi en raison des coûts élevées de leur fabrication et de leur courte durée de vie de 3 à 5 ans [3].

Aussi, en plus du manque de données, la majorité des cas de morsures de serpents ne sont pas signalés. En effet, après une étude réalisée sur les morsures de serpents en Inde, il s’est avéré que les recherches ont donné des résultats jusqu’à 30 fois plus élevés que les chiffres rapportés par le gouvernement [4]. La situation actuelle étant critiques face à cette problématique, l’OMS a inscrit les morsures de serpents sur la liste des maladies tropicales négligées en 2017 [5].

Étant données les problématiques citées ci-dessus, j’ai reçu un mandat de la part des HUG ayant pour objectif le développement d’une solution informatique qui permettra d’identifier un serpent. En plus de répondre au besoin direct d’un utilisateur, celle-ci contribuera à offrir des solutions pour les problèmes cités ci-dessus. Elle permettra indirectement la récolte des données concernant la faune des serpents. En effet, pour chaque serpent qui sera publié par un utilisateur, la localisation depuis laquelle a été posté ce serpent sera aussi enregistrée. De cette façon, des analyses statistiques pourront être effectuées sur ces données afin d’avoir par exemple une vue d’ensemble des serpents venimeux par région. Ces informations seront essentielles notamment pour les instituts de santé publique qui sauront quels sont les anti-venins dont ils ont le plus besoin. Enfin, l’application disposera aussi d’une documentation telle qu’un guide de bonnes pratiques en cas de morsure de serpent et des informations sur les serpents identifiés.

2. Méthodologie

2.1 Description et Caractéristiques

2.1.1 Problématique

Résumons les problèmes liés aux morsures de serpents cités précédemment :

- Tout d'abord, il y a un manque d'information quant à la faune des serpents. C'est-à-dire qu'on n'a pas suffisamment d'informations concernant les serpents qu'il peut y avoir dans des régions spécifiques.
- Par conséquent, les hôpitaux n'ont pas les quantités adéquates d'anti-venin et parfois certains anti-venins restent non-utilisés et doivent être jetés car ceux-ci ont une courte durée de vie.
- En raison de l'insuffisance de stock d'anti-venin dans les instituts de santé publique, seule une minorité des personnes mordues peut être soignée.
- Enfin, la majorité des cas de morsure de serpent ne sont pas signalés.

2.1.2 Description de l'application

Cette application permettra d'identifier un serpent qu'un utilisateur aura pris en photo. Une fois la photo publiée, l'ensemble des spécialistes ayant pour tâche d'identifier le serpent seront notifiés.

Au fur à mesure que les spécialistes iront voter, l'utilisateur pourra déjà consulter les résultats avant même que l'identification ne soit validée, ce qui lui donnera une idée de l'espèce de serpent qu'il cherche à identifier. Une fois les votes terminés, l'utilisateur pourra voir le résultat final avec un résultat contenant les trois serpents les plus votés.

Lors du vote, les spécialistes auront comme information la photo du serpent ainsi que le lieu depuis lequel elle a été publiée. De plus, la solution informatique possèdera aussi une documentation sur les serpents identifiés ainsi qu'un guide concernant les bonnes pratiques à mettre en place en cas de morsure.

2.1.3 Solutions apportées par l'application

Tout d'abord, l'application sera en mesure de répondre à un besoin immédiat. C'est-à-dire celui d'identifier un serpent. Toutefois, sur le long terme elle permettra aussi la récolte de données qui aujourd'hui encore est cruciale pour lutter contre la problématique des morsures de serpent. En effet, il sera possible d'effectuer des analyses statistiques afin d'avoir des chiffres concernant les serpents et les morsures qu'il y a eu par région.

Cela permettra de tenir informés les hôpitaux afin qu'ils s'adaptent et prévoient les anti-venins adéquats car rappelons qu'à cause du manque d'anti-venin seulement 10% des personnes mordues en Afrique peuvent être soignées. Ce chiffre pourrait se voir augmenter avec le temps et des vies humaines pourraient être sauvées. De plus, en réduisant les erreurs dans les approvisionnements d'anti-venin cela réduit aussi les coûts financiers des instituts de santé publique.

Enfin, la documentation présente sur l'application pourra servir pour sensibiliser les gens et les former un minimum si l'un d'eux venait à subir une morsure. Par exemple, les simples gestes comme calmer la personne mordue ainsi que de la placer au repos pour ralentir la diffusion du venin peuvent être décisifs [6].

2.2 Méthodes de développement

Étant donné que j'effectuerai ce travail seul, l'utilisation d'une des méthodes de gestion de projet classique n'aurait pas de sens car la plupart contiennent plusieurs rôles. Je vais donc définir une méthode de développement que je vais utiliser tout le long de ce projet. Comme illustré ci-dessous sur la figure 1, les trois premières étapes concernent la documentation ainsi que la planification du projet et n'auront lieu qu'une seule fois (au début du projet) tandis qu'il y aura plusieurs itérations pour l'étape développement. Vous trouverez ci-dessous une explication pour chacune de ces étapes :

Documentation : Lors de cette étape il faudra fournir une documentation donnant une vue d'ensemble et une bonne compréhension du projet, à savoir la définition des besoins du mandant, les technologies qui seront utilisées, une analyse de risques et l'architecture de l'application.

Spécification : Il s'agira de déterminer les fonctionnalités de l'application qui seront à développer durant ce travail de Bachelor (On peut considérer cela comme la définition d'un « work-item list » ou « product backlog »).

Planification : Cette étape consistera à planifier l'implémentation des fonctionnalités déterminées à l'étape précédente à l'aide d'un diagramme de Gantt.

Développement : Chaque itération de cette étape représentera le développement d'une fonctionnalité ou d'une partie de fonctionnalité.

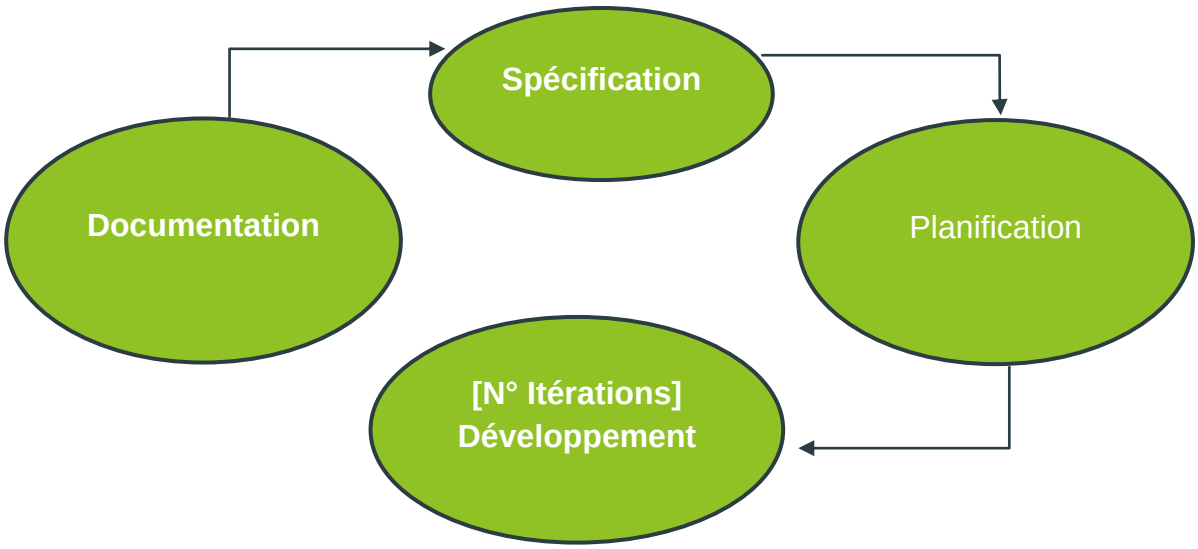


Figure 1 - Méthodes de développement

2.2.1 Structure d'une itération

Sont illustrées sur la figure 2 les étapes par lesquelles je passerai pour chaque itération à l'exception de la première itération durant laquelle il faudra déterminer l'architecture de l'application, un modèle de données (qui pourrait évoluer par la suite) et effectuer la mise en place de l'environnement. Toutes les autres itérations seront composées de ces 4 étapes :

Analyse & Conception : Lors de cette étape il s'agira de trouver une solution pour développer la fonctionnalité demandée puis de déterminer les différents composants qui seront utilisés.

Implémentation : Lors de cette étape, par la suite des solutions trouvées précédemment, la fonctionnalité en question sera développée.

Test : Pour cette étape, seulement si le temps restant à disposition est suffisant, il faudra développer des tests pour la fonctionnalité concernée.

Évaluation : Enfin, il s'agira de « faire » une démonstration de la fonctionnalité afin de savoir si celle-ci correspond aux attentes.

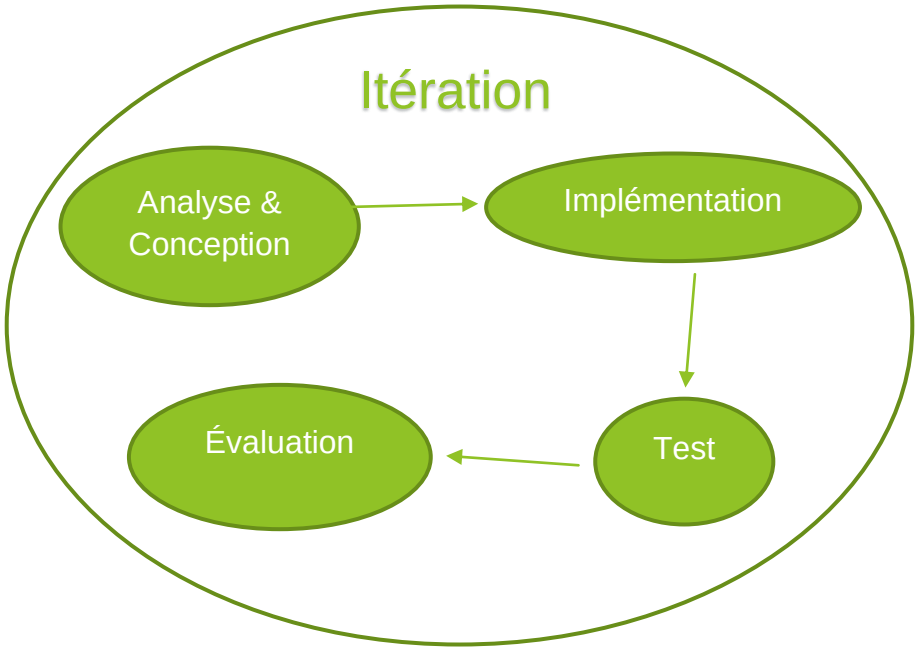


Figure 2 - Structure d'une itération

2.3 Caractéristiques de l'application

Les fonctionnalités de l'application implémentées durant ce travail de Bachelor seront les suivantes :

- Photographier un serpent, récupérer la localisation et publier l'ensemble des informations
- Notifier les spécialistes lors d'une publication de serpent à identifier

- Afficher les serpents identifiés et non-identifiés pour les utilisateurs et les spécialistes
- Afficher le détail d'une publication de serpent avec les résultats en pourcentage des trois serpents les plus votés.
- Enregistrer le vote d'un spécialiste concernant un serpent non-identifié

2.3.1 Outils et framework

Ionic

Le framework open-source hybride Ionic sera utilisé pour le front-end. Il est basé sur les technologies web HTML, CSS et Javascript [7]. Le mandat étant de développer une application mobile, celui-ci répond parfaitement à ce besoin.

Spring boot

Ayant pour seule contrainte l'utilisation du langage Java pour le back-end, le framework open-source Spring Boot sera utilisé pour le back-end. Il est basé sur le langage de programmation Java et est aussi utilisé pour créer des micro-services [8].

Oracle

Les données seront stockées sur le système de gestion de base de données Oracle. J'ai fait ce choix car celui-ci supporte le PL/SQL et que j'ai déjà utilisé ce langage durant mon parcours,

GitLab

J'utiliserai GitLab pour la gestion de version de mes projets front-end et back-end.

2.3.2 Architecture de l'application

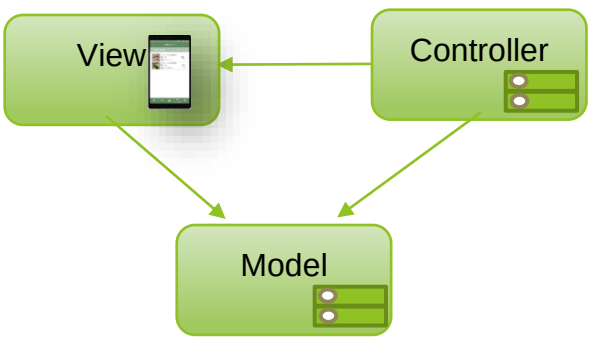
La structure choisie de l'application est : l'architecture MVC (Model View Controller). *Voici ci-dessous une explication de la figure 3 :*

Model : Le Model serait représenté par Oracle. Il contient toutes les données et s'occupe de stocker ainsi que de retourner les données demandées.

Controller : Le Controller s'occupera de faire le lien entre la vue et le modèle selon des actions spécifiques. C'est le projet Spring Boot qui aura cette responsabilité.

View : La Vue Aura pour rôle d'afficher les informations nécessaires à l'écran. Celle-ci sera représentée par le front-end Ionic.

Figure 3 - Architecture de l'application



3. Besoins des utilisateurs

3.1 Définitions des user Personnas

Un « user persona » représente un utilisateur potentiel de l'application et permet de mieux définir les cas d'utilisations, les besoins réels auxquels peut répondre l'application et les utilisateurs cibles d'une application [9]. Ci-dessous seront définis et décrits plusieurs user personas ainsi que leurs objectifs dans le tableau 1.

Tableau 1 - Users Personas

Rôle	Détails	Objectifs
La personne mordue	• Il s'agit d'une personne n'ayant pas ou peu de connaissances à propos des serpents et qui vient de se faire mordre par un serpent. Celle-ci ne sait pas si le serpent qui l'a mordue est dangereux ou non et souhaiterait le faire identifier.	• Prendre un serpent en photo afin de le faire identifier • Lire le guide des bonnes pratiques en cas de morsure • Accéder aux numéros d'urgence de la région • Consulter la documentation des serpents connus. • Trouver les centres d'urgence les plus proches
L'expert(e) en serpent	• C'est un spécialiste des reptiles et celui-ci est spécialisé dans le domaine des serpents. Il souhaite simplement faire identifier des espèces de serpent qu'il a pris en photo.	• Consulter la documentation des serpents connus • Prendre un serpent en photo afin de le faire identifier • Lire le guide des bonnes pratiques en cas de morsure
Le/la spécialiste	• C'est une personne qui a de bonnes connaissances en serpent. Celle-ci a réussi le test de connaissance de l'application et a obtenu le titre de "Spécialiste". Elle souhaite aider la communauté en contribuant à l'identification de serpents.	• Voter lors d'un post afin d'identifier un serpent • Consulter la documentation des serpents connus • Voir son classement • Accéder au forum afin d'avoir des échanges avec d'autres experts

3.2 Définition des stories

Ci-bas sera défini le product backlog contenant les user stories déterminées à partir des user personnas définis au point précédent. Les user stories seront classées par ordre d'importance MOSCOW. Chaque story sera estimée en temps et en difficulté. Rappel des trois user personnas :

- La personne mordue
- L'expert(e) en serpent
- Le/la spécialiste

P.S : Lorsque le terme « utilisateur » sera utilisé, cela englobe les trois user personnas.

3.2.1 Story de base

Pour estimer les stories qui vont suivre je me suis basé sur une story de base à laquelle j'ai fixé un niveau de difficulté et une durée pour la terminer. Pour estimer le niveau de difficulté j'ai utilisé les chiffres de la suite de Fibonacci (1,3,5,8,13, etc.). Plus le chiffre est élevé, plus la story est considérée comme difficile. Chaque story se verra détaillée dans le point suivant. Ci-dessous un exemple d'estimation de story sur le tableau 2.

Tableau 2 – Exemple de story

MOSCOW	Story de base	Difficulté	Jour (100%)
Must Should Could	En tant que … je souhaite … afin de …	1,3,5,8,13, …	1j = 8h

3.3 Product Backlog

Tableau 3 - Product Backlog

MOSCOW	EPIC	Story	Difficulté	Jours
Must	Poster une photo de serpent.	**<u>Sous-stories</u>** 1.1 En tant qu'expert en serpent / personne mordue je souhaite pouvoir prendre une photo d'un serpent afin de le faire identifier.	8	7
		1.2 En tant qu'expert en serpent / personne mordue je souhaite voir le résultat de l'identification du serpent que j'ai publié afin de le transmettre aux secours.		
		1.3 En tant qu'expert en serpent / personne mordue je souhaite voir les serpents identifiés et non identifiés que j'ai publié.		

Must	Identifier un serpent.	**<u>Sous-stories</u>** 1.1 En tant spécialiste je souhaite accéder à la liste des posts d'identification de serpents non-identifiés pour les identifier.	8	7
		1.2 En tant que spécialiste je souhaite pouvoir voter sur un post de serpent afin de l'identifier		
		1.3 En tant que spécialiste je souhaite accéder à la liste des posts d'identification de serpents auxquels j'ai voté afin de savoir mes résultats.		
Should		En tant qu'expert en serpent / personne mordue je souhaite obtenir le rôle de spécialiste afin de pouvoir identifier un serpent	3	3
Could		En tant qu'expert en serpent / personne mordue je souhaite pouvoir supprimer un post afin de pouvoir faire le tri de ma liste de posts.	5	5

Could		En tant que spécialiste je souhaite accéder à la liste des posts d'identification de serpents auxquels j'ai voté afin de savoir mes résultats.	3	3
Could		En tant que spécialiste je souhaite avoir accès au ranking afin de voir mon classement par rapport aux autres spécialistes.	2	2
Could		En tant qu'utilisateur je souhaite avoir accès à la documentation afin de me tenir informé à propos des serpents et des bonnes pratiques à leur encontre.	2	2
Could		En tant qu'utilisateur je souhaite voir la liste des serpents connus afin de me tenir informé de tout type de serpent.	3	3
Could		En tant que client je souhaite savoir quoi faire en cas de morsure afin de savoir quoi faire si cela arrive.	2	2
Could		En tant que client je souhaite pouvoir trouver un centre d'urgence à proximité afin de pouvoir y aller en cas de nécessité.	8	7
Could		En tant que client je souhaite avoir une liste des numéros d'urgence que je peux appeler selon mon pays afin de pouvoir appeler des secours en cas de nécessité.	5	5
Could		En tant que client je souhaite pouvoir reconnaître un serpent venimeux d'un inoffensif	3	3

		afin de pouvoir agir en conséquence en conditions réelles.		
		Total		49 jours

3.4 Temps à disposition

Tableau 4 - Durée à disposition

Durée	Nb jours par mois (sur 28)	Nb jours total	Nb heures totales	Temps réel à disposition (80% du temps total)
4 mois à mi-temps (50%)	20	80	320	64 jours soit 256 heures

3.5 Diagramme des cas d’utilisation global

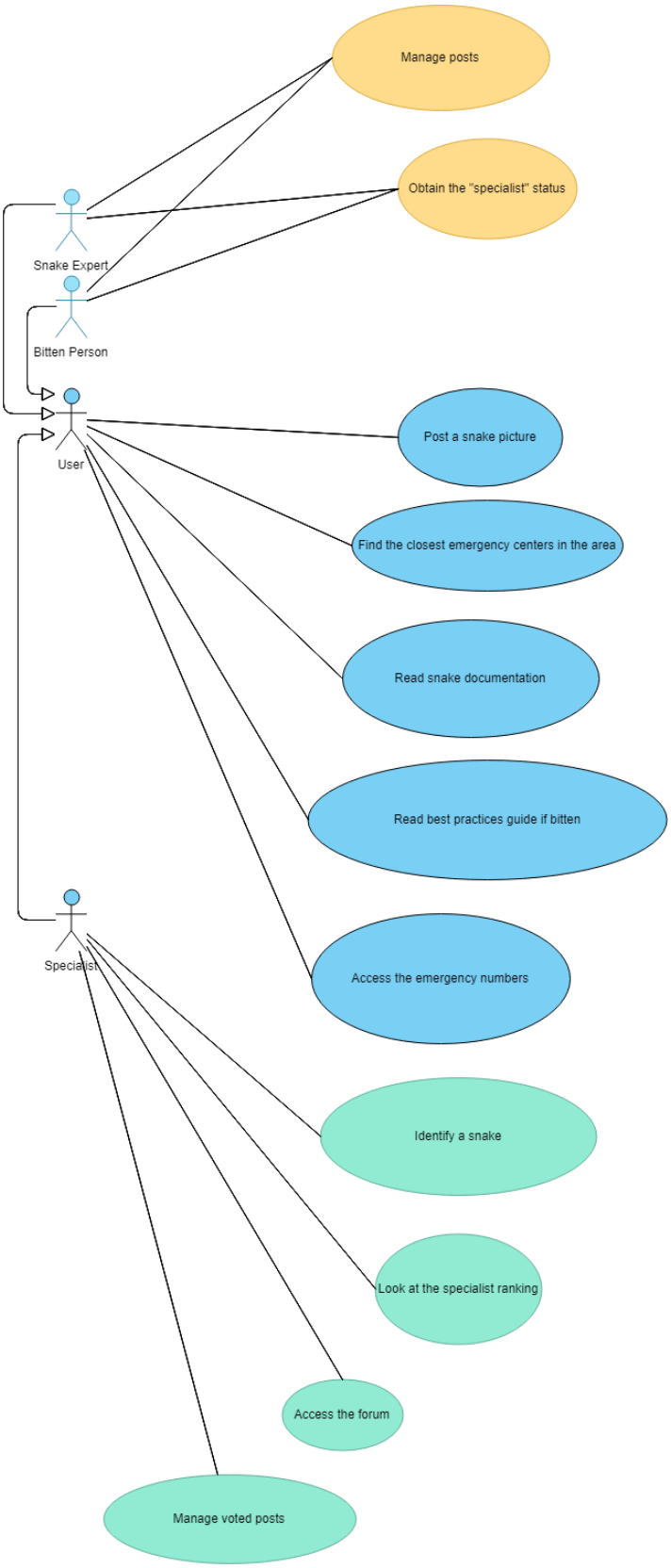


Figure 4 - Diagramme des cas d'utilisation global

3.6 Diagramme des cas d'utilisation détaillés

3.6.1 Publier une photo de serpent

Nom : Publier une photo

Acteur : Utilisateur

Acteur secondaire : Aucun

Déclencheur : L'utilisateur clique sur le bouton de l'appareil photo.

Flot principal :

1. Le système affiche l'appareil photo
2. L'utilisateur prend une photo
3. Le système demande au client de valider la photo
4. Le système crée un nouveau post avec la photo
5. Fin UC

Flot alternatif :

1a) L'utilisateur importe une photo depuis sa galerie

1a1) Passage à l'étape 4

3a) L'utilisateur souhaite reprendre une photo

3a1) Retour à l'étape 1

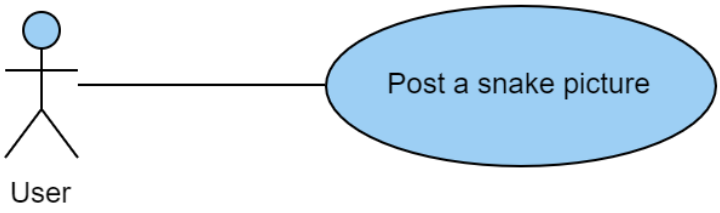


Figure 5 - Publier une photo de serpent

3.6.2 Consulter une publication de serpent

Nom : Consulter une publication de serpent

Acteur : Utilisateur

Acteur secondaire : Aucun

Déclencheur : L'utilisateur sélectionne l'onglet « identification » ou « observation ».

Flot principal :

1. Le système affiche la liste des publications de serpent
2. L'utilisateur sélectionne une de ses publications
3. Le système affiche les détails de la publication
4. Fin UC

Flot alternatif :

3a) La personne souhaite faire un traitement de type CRUD sur la publication

3a1) Le système effectue le traitement CRUD demandé

3a2) Retour à l'étape n°1

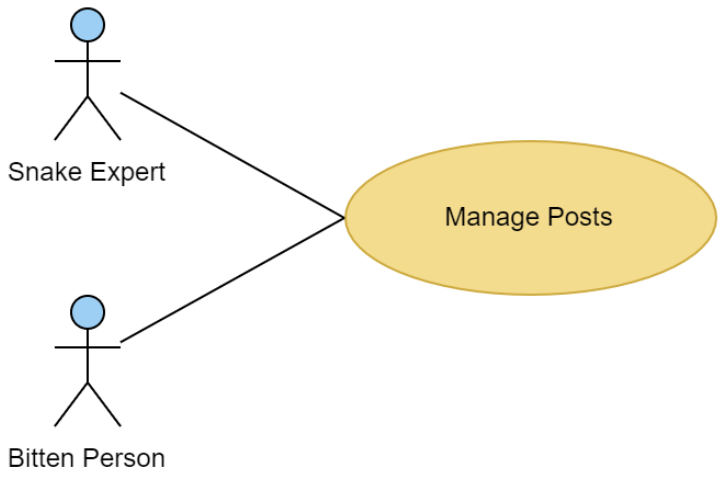


Figure 6 - Gérer une publication

3.6.3 Trouver un centre d'urgence à proximité

Nom : Trouver un centre d'urgence à proximité

Acteur : Utilisateur

Acteur secondaire : Aucun

Déclencheur : L'utilisateur accède à l'onglet « Trouver un centre d'urgence à proximité ».

Flot principal :

1. Le système récupère la localisation du client
2. Le système affiche les centres les plus proches de la localisation du client
3. L'utilisateur sélectionne un centre d'urgence
4. Le système affiche les détails du centre d'urgence sélectionné
5. Fin UC

Flot alternatif :

1a) La localisation du client est désactivée

1a1) Le système demande l'activation de la localisation

1a2) L'utilisateur active la localisation

1a3) Passage à l'étape n°2

1b1) L'utilisateur n'active pas la localisation

1b2) Fin UC

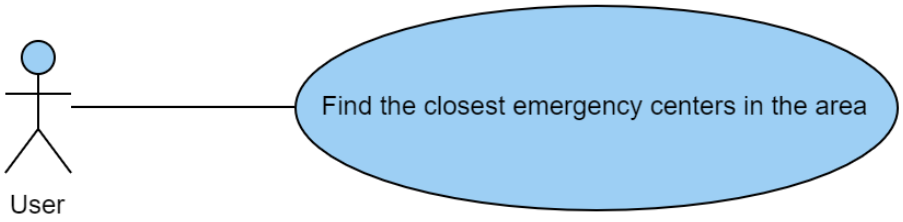


Figure 7 - Trouver le centre d'urgence le plus proche de la région

3.6.4 Obtenir le statut de spécialiste

Nom : Obtenir le statut de spécialiste

Acteur : Expert en serpent, personne mordue

Acteur secondaire : Aucun

Déclencheur : L'expert clique sur « Aider la communauté ».

Flot principal :

1. Le système affiche QCM avec des questions aléatoires
2. L'expert répond aux questions
3. Le système calcule et affiche le résultat
4. L'expert obtient le statut de spécialiste
5. L'expert est redirigé vers la partie « Communauté »
6. Fin UC

Flot alternatif :

4a) Le résultat obtenu par l'expert est insuffisant pour lui attribuer le statut de spécialiste

4a1) Le système affiche le résultat et un message d'information

4a2) Fin UC

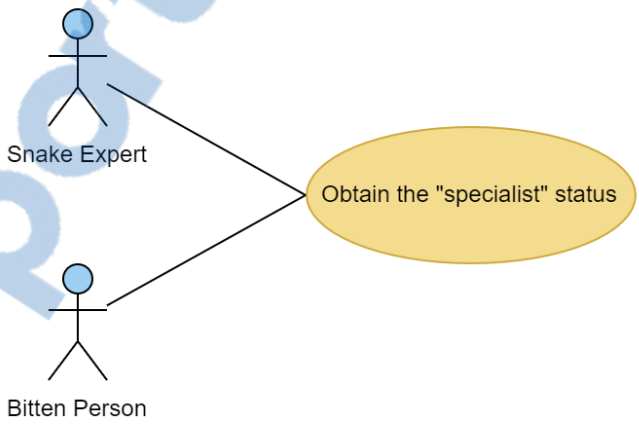


Figure 8 - Obtenir le statut de spécialiste

3.6.5 Consulter tout type de documentation

Nom : Consulter la documentation (des serpents, guide des bonnes pratiques, liste des numéros d'urgence, centres d'urgences à proximité)

Acteur : Utilisateur

Acteur secondaire : Aucun

Déclencheur : L'utilisateur accède à la section « documentation ».

Flot principal :

1. Le système affiche la section documentation
2. L'utilisateur sélectionne une documentation spécifique
3. Le système affiche les détails de cette documentation
4. Fin UC

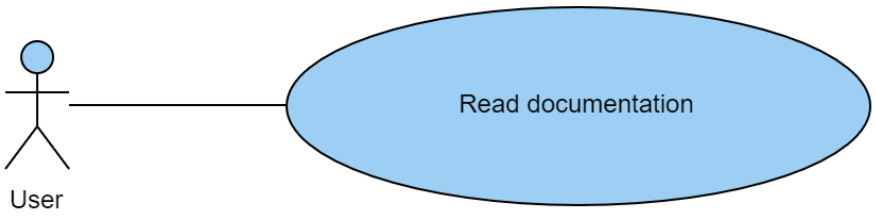


Figure 9 - Accéder à la documentation

3.6.6 Identifier un serpent

Nom : Identifier un serpent

Acteur : Spécialiste

Acteur secondaire : Aucun

Déclencheur : L'utilisateur accède à l'onglet « Identification ».

Flot principal :

1. Le système affiche la partie communautaire
2. Le spécialiste sélectionne la section « Identifier un serpent »
3. Le système affiche la liste des serpents identifiés auxquels il a voté
4. Le spécialiste sélectionne l'onglet des serpents non-identifiés
5. Le système affiche la liste des serpents non-identifiés
6. Le spécialiste sélectionne une publication de serpent non-identifié
7. Le système affiche le détail de la publication avec les choix multiples
8. Le spécialiste vote/identifie le serpent parmi les choix multiples
9. Le système enregistre la réponse du spécialiste
10. Fin UC

Flot alternatif :

5a) Il n'y a aucun serpent non-identifié

5a1) Fin UC

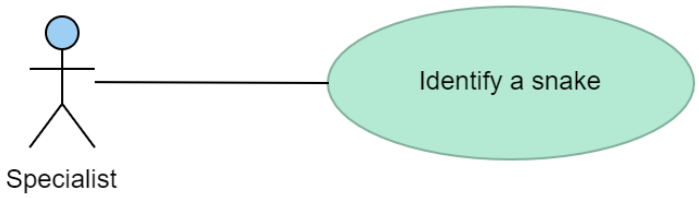


Figure 10 - Identifier un serpent

3.6.7 Consulter son classement de spécialiste

Nom : Consulter son classement de spécialiste

Acteur : Spécialiste

Acteur secondaire : Aucun

Déclencheur : L'utilisateur accède à l'onglet « Communauté ».

Flot principal :

1. Le système affiche la partie communautaire
2. Le spécialiste sélectionne la section « Classement »
3. Le système récupère les informations du spécialiste
4. Le système affiche le classement du spécialiste avec son nombre de points
5. Fin UC

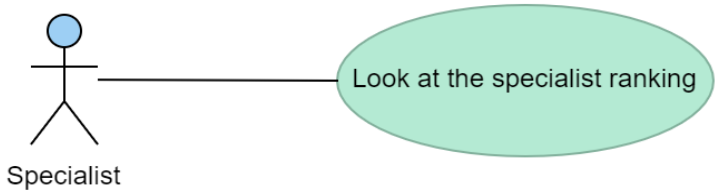


Figure 11 - Regarder son classement

3.6.8 Consulter le forum

Nom : Consulter le forum

Acteur : Spécialiste

Acteur secondaire : Aucun

Déclencheur : L'utilisateur accède au forum.

Flot principal :

1. Le système affiche la partie communautaire
2. L'utilisateur sélectionne la section « Forum »
3. L'utilisateur est redirigé sur le forum
4. Fin UC

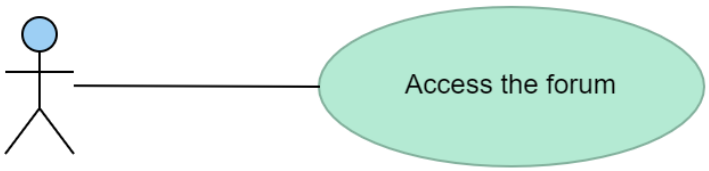


Figure 12 - Accéder au forum

3.6.9 Consulter ses posts votés

Nom : Consulter ses posts votés

Acteur : Spécialiste

Acteur secondaire : Aucun

Déclencheur : Le spécialiste accède à l'onglet « Identifier un serpent ».

Flot principal :

1. Le système affiche la liste des serpents identifiés que le spécialiste a identifié
2. Le spécialiste sélectionne une publication de serpent identifié
3. Le système affiche le détail de la publication sélectionnée
4. Fin UC

Flot alternatif :

3a) Il n'y a aucun post de serpent identifié auquel le spécialiste a voté

3a1) Fin UC

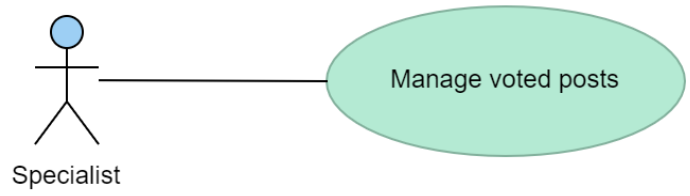


Figure 13 - Gérer ses votes

3.7 Modèle logique de données

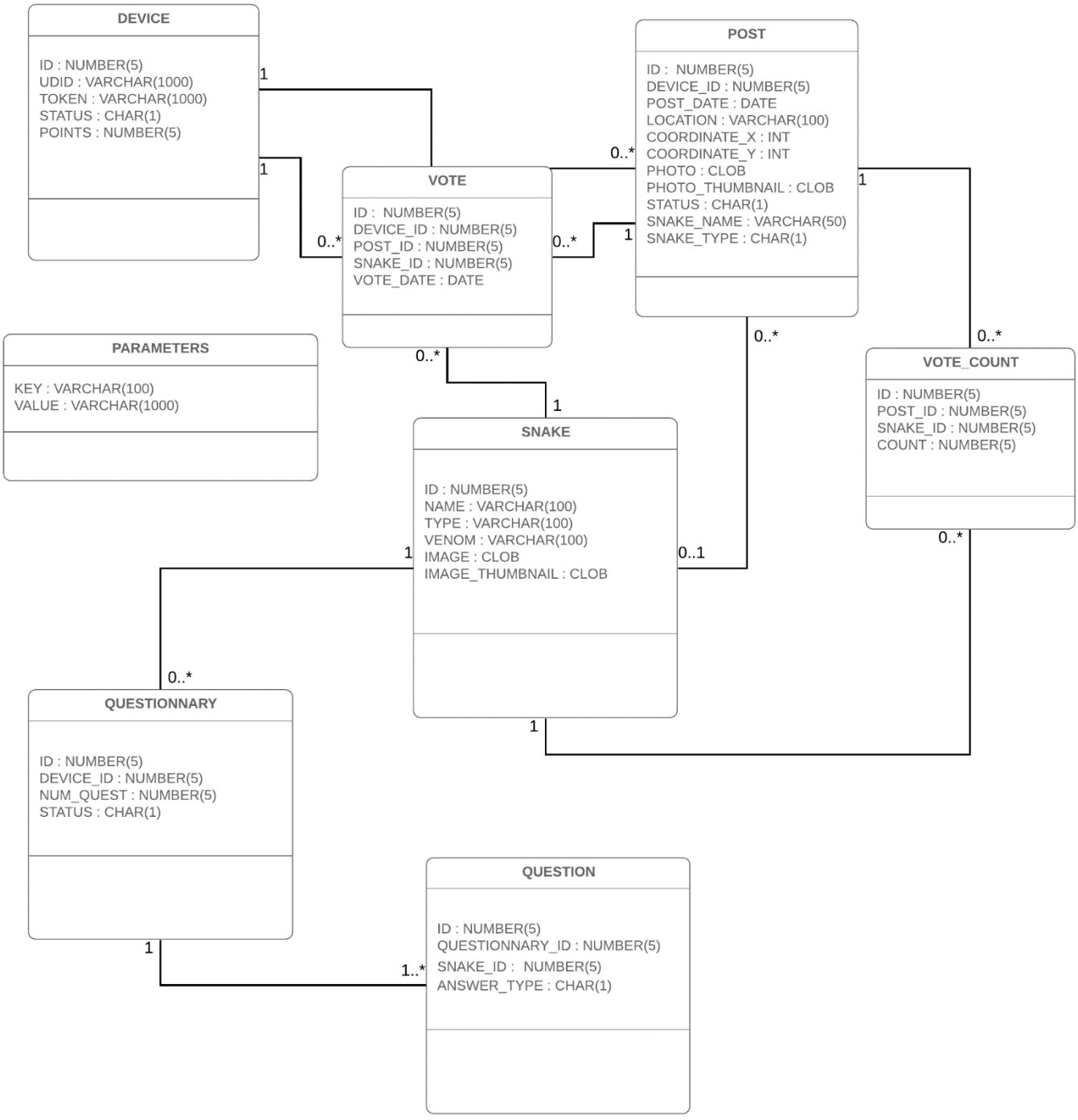


Figure 14 - Modèle logique de données

3.8 Définition des API

3.8.1 Requêtes HTTP POST

Enregistrer un post de serpent

Lorsqu'une personne aura pris un serpent en photo celle-ci devra être postée sur l'url « /posts ».

Exemple : http://adresseDuServeur:8080/posts

Paramètres

À cette adresse, devront être envoyés les paramètres suivants dans le même ordre avec l'UDID de l'appareil en header « Parameter » :

- Les coordonnées X de la localisation de la personne
- Les coordonnées Y de la localisation de la personne
- La photo en question

Enregistrer le vote d'un utilisateur concernant un post

Lorsqu'une personne aura voté sur un post de serpent, sa réponse devra être enregistrée. Il faudra enregistrer sa réponse sur l'url « /votes ».

Exemple : http://adresseDuServeur:8080/votes

Paramètres

À cette adresse, devront être envoyés les paramètres suivants avec l'UDID de l'appareil en header « Parameter » :

- L'ID du serpent sélectionné
- L'ID du post sur lequel il a voté

Enregistrer un serpent inconnu

Si on souhaite enregistrer un nouveau serpent. Il faudra l'enregistrer sur l'url « /snakes ».

Exemple : http://adresseDuServeur:8080/snakes

Paramètres

À cette adresse, devront être envoyés les paramètres suivants avec l'UDID de l'appareil en header « Parameter » :

- Le nom du serpent
- Le type du serpent (dangereux, inoffensif)
- Son type de venin (cytotoxiques, neurotoxiques, hémotoxiques, aucun, etc.)

3.8.2 Requêtes HTTP GET

Afficher tous les serpents identifiés d'un utilisateur

Lorsqu'un utilisateur voudra voir la liste de tous les serpents identifiés qu'il a publié, il faudra les récupérer sur l'adresse « /posts/identified » avec l'UDID de l'appareil en header « Parameter ».

Exemple : http://adresseDuServeur:8080/posts/identified?page=x&size=y

Réponse

Sera récupérée la réponse sous le format suivant :


```
“posts” [
{
  "id_post” : 1,
  "photo": "",
"thumbnail": "",
  "coordinate_x": 50,
  "coordinate_Y": 36,
  "status": "I",
“sna_name” : “Black Mamba”,
“sna_type” : ‘D’
},
{
  "id_post” : 2,
  "photo": "",
"thumbnail": "",
  "coordinate_x": 50,
  "coordinate_Y": 36,
  "status": "I",
“sna_name” : “Black Mamba”,
“sna_type” : ‘D’
},
]
```


Afficher tous les serpents non-identifiés d'un utilisateur

Lorsqu'un utilisateur voudra voir la liste de tous les serpents non-identifiés qu'il a publié, il faudra les récupérer sur l'adresse « /posts/identified » avec l'UDID de l'appareil en header « Parameter ».

Exemple : http://adresseDuServeur:8080/posts/unidentified?page=x&size=y

Réponse

Sera récupérée la réponse sous le format suivant :


```
“posts” [  {
  "id_post” : 1,
  "photo":
"
DAoMDAsKC",
"thumbnail":
"
DAoMDAsKC",
  "coordinate_x":  50,
  "coordinate_Y": 36,
  "status": "U",
“sna_name” : “Unknown”,
“sna_type” : ‘U’
},
{
  "id_post” : 2,
  "photo":
"
DAoMDAsKC",
"thumbnail":
"
DAoMDAsKC",
  "coordinate_x":  50,
  "coordinate_Y": 36,
  "status": "U",
“sna_name” : “Unknown”,
“sna_type” : ‘U’
}, ]
```


Afficher tous les serpents identifiés pour lesquels un spécialiste a voté

Lorsqu’un spécialiste voudra voir la liste de tous les serpents identifiés pour lesquels il a voté, il faudra les récupérer sur l’adresse « /posts/identified/specialists » avec l’UDID de l’appareil en header « Parameter ».

Exemple : http://adresseDuServeur:8080/posts/identified/specialists?page=x&size=y

Réponse

Sera récupérée la réponse sous le format suivant :


```
“posts” [
{
  "id_post” : 1,
```

"photo":
"",

"thumbnail":
"",

"coordinate_x": 50,

"coordinate_Y": 36,

"status": "I",

“sna_name” : “Black Mamba”,

“sna_type” : ‘D’

},

{

"id_post” : 1,

"photo":
"",

"thumbnail":
"",

"coordinate_x": 50,

"coordinate_Y": 36,

"status": "I",

“sna_name” : “Black Mamba”,

“sna_type” : ‘D’

},

]

Afficher tous les serpents non-identifiés pour un spécialiste

Lorsqu’un spécialiste voudra voir la liste de tous les serpents non-identifiés, il faudra les récupérer sur l’adresse « /posts/unidentified/specialists » avec l’UDID de l’appareil en header « Parameter ».

Exemple : http://adresseDuServeur:8080/posts/unidentified/specialists?page=x&size=y

Réponse

Sera récupérée la réponse sous le format suivant :

“posts” [{

"id_post” : 1,

"photo":
"",

"thumbnail":
"",

"coordinate_x": 50,

"coordinate_Y": 36,

"status": "U",

“sna_name” : “Unknown”,

“sna_type” : ‘U’

},

{

"id_post” : 2,

"photo":
"",

"thumbnail":
"",

"coordinate_x": 50,

"coordinate_Y": 36,

"status": "U",

“sna_name” : “Unknown”,

“sna_type” : ‘U’

},]

Afficher le détail d’un post

Lorsqu’un utilisateur voudra voir une de ses publications de serpent qu’il a publié, il faudra la récupérer sur l’adresse « /posts/{id} » avec l’UDID de l’appareil en header « Parameter ».

Exemple : http://adresseDuServeur:8080/posts/3

Réponse

Sera récupérée la réponse sous le format suivant :

{

"id_post” : 1,

« dev_id » : 2,

« date » : 22.11.2019

"photo":
"",

"thumbnail":
"",

"coordinate_x": 50,

"coordinate_Y": 36,

"status": "U",

“sna_name” : “Black Mamba”,

“sna_type” : ‘D’ },

Afficher la liste de tous les serpents

Lorsqu’un utilisateur voudra voir la liste de tous les serpents, il faudra les récupérer sur l’adresse « /snakes » avec l’UDID de l’appareil en header « Parameter ».

Exemple : http://adresseDuServeur:8080/snakes

Réponse

Sera récupérée la réponse sous le format suivant :

```
“snakes” [
{
  "snake_id” : 1,
  "snake_name": Mamba noir,
  "snake_type":  "dangereux",
  "venom_type": "neurotoxique"
  "snake_picture": “…”,
  "snake_thumbnail": “…”
},
{
  "snake_id” : 2,
  "snake_name": "Boiga",
  "snake_type":  "inoffensif",
  "venom_type": "aucun"
  "snake_picture": “…”,
  "snake_thumbnail": “…”
}
```

Afficher la liste des noms des serpents

Lorsqu’un utilisateur voudra voir la liste de tous les serpents mais uniquement en récupérant leurs noms, il faudra les récupérer sur l’adresse « /snakes/names » avec l’UDID de l’appareil en header « Parameter ».

Exemple : http://adresseDuServeur:8080/snakes/names

Réponse

Sera récupérée la réponse sous le format suivant :

```
“snakes” [
{
  "snake_id” : 1,
  "snake_name": Mamba noir,
},
{
  "snake_id” : 2,
  "snake_name": "Boiga",
}
```

Afficher le détail d’un serpent

Lorsqu’un utilisateur voudra voir les informations d’un serpent en particulier, il faudra les récupérer sur l’adresse « /snakes/ {idSnake} » avec l’UDID de l’appareil en header « Parameter ».

Exemple : http://adresseDuServeur:8080/snakes/1

Réponse

Sera récupérée la réponse sous le format suivant :

```
{
  "snake_id” : 1,
  "snake_name": Mamba noir,
  "snake_type":  "dangereux",
  "venom_type": "neurotoxique"
  "snake_picture": “…”,
  "snake_thumbnail": “…”
}
```

4. Résultats

4.1 Mise en place de l’environnement

4.1.1 Structuration du projet

Comme illustré sur la figure 15, le serveur sera composé des 5 packages suivants :

Tableau 5 – Structure du projet

Controller	Contiendra tous les controllers qui auront pour tâche de retourner ou d’enregistrer une ressource lors d’un appel sur une URL spécifique.
Dao	Ici se trouveront toutes les classes à partir desquelles seront récupérées ou enregistrées les données dans la base de données.
Exceptions	Toutes les exceptions que le serveur retournera selon des cas spécifiques se trouveront dans ce package.
Model	Tous les objets métiers se trouveront dans ce package.
Tools	Contiendra les outils qui seront utilisés tout le long du projet.

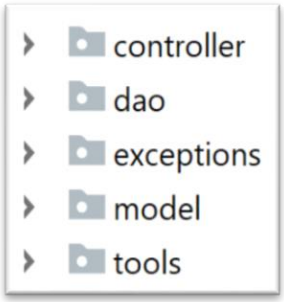


Figure 15 - Structure du projet

4.1.2 Création des tables et insertions des données

Avant toute chose, il faut créer les tables afin que les données puissent être stockées et récupérées. Les tables ont été créées selon le MLD défini dans la documentation. Dans ce cas, les créations des tables, des séquences et des données se trouvent dans le même fichier mais il est possible de les séparer en fichier distincts.

Voici sur la figure 16, le script de création de la table des serpents et d'insertion d'un serpent

```
CREATE TABLE snake (
sna_id NUMBER(5) CONSTRAINT pk_snake PRIMARY KEY,
sna_name VARCHAR2(50) CONSTRAINT nn_sna_name NOT NULL,
sna_type CHAR(1) CONSTRAINT nn_sna_type NOT NULL,
sna_venom VARCHAR2(20) CONSTRAINT nn_sna_venom_type NOT NULL,
sna_img CLOB CONSTRAINT nn_sna_img NOT NULL,
sna_img_thumbnail CLOB CONSTRAINT nn_sna_img_thumb NOT NULL,
CONSTRAINT chk_snake_type CHECK (sna_type IN ('D','H')),
CONSTRAINT chk_snake_venom CHECK (sna_venom IN ('Cytotoxic','Hemotoxic', 'Neurotoxic', 'Myotoxic', 'None'))
);

INSERT INTO snake VALUES(1, 'Black Mamba', 'D', 'Hemotoxic', '/9', '/9');
```

Figure 16 - Création et insertion de serpents

Aussi, il est possible d'intégrer la création des tables et les insertions de données directement dans Spring Boot. Celui-ci exécutera les scripts lors du lancement du projet. Toutefois, il faut les séparer en deux fichiers distincts et les placer dans les ressources du projet. De plus, il faudra renommer le script de création des tables en « schema.sql » et celui des insertions de données en « data.sql » pour qu'ils soient exécutés lors du lancement du projet.

4.1.3 Récupération et traitement des données avec JPA

Tout d'abord, qu'est-ce qu'un JPA ? JPA aussi appelé Java Persistence API permet de faire le lien entre nos classes java et les tables relationnelles de notre base de données [10]. Prenons par exemple la table snake illustrée dans la figure 16, nous pourrions donc faire le lien entre notre classe java « snake » et cette table.

Pour ce faire, il faut premièrement annoter les classes qui auront pour but de « représenter » nos tables avec des annotations spécifiques. Vous trouverez ci-dessous une explication pour chaque annotation dans le tableau 6.

Tableau 6 - Annotation des classes métier java

Annotation	Explication
@Entity	Permet d'informer JPA que les instances d'une classe devront être persistées dans la base de données [10].
@Table(name = "nom de la table")	Permet d'informer JPA à quelle table de notre base de données la classe correspond.
@Column(name = "nom de la colonne")	Permet d'informer JPA à quelle colonne de la table de notre base de données l'attribut de la classe correspond.
@Id	Permet d'informer JPA que l'attribut de la classe correspond à l'ID d'un élément de la table.
@SequenceGenerator(name = "nom de la séquence")	Permet d'informer JPA quant à la séquence qu'il devra utiliser pour générer un ID.

Une fois nos classes métier annotées, il faut récupérer les données contenues dans la base de données. Pour cela, nous allons créer des interfaces DAO. Expliquons avant tout ce terme. DAO signifie Data Access Object, un DAO permettra de récupérer, modifier et supprimer un ou plusieurs éléments contenus dans nos tables à partir des classes métiers comme expliqué précédemment [11].

Comme défini précédemment, nous aurons besoin non seulement de faire des traitements CRUD (Create, Read, Update, Delete) mais aussi de retourner les éléments de façon paginée, c'est-à-dire qu'au lieu de retourner tous les éléments, nous en retournerons seulement une partie selon les informations fournies par le côté client. Nos DAO hériteront donc de l'interface « JpaRepository ». Comme illustré sur la figure 17, celle-ci nous fournira les méthodes pour les traitements CRUD mais aussi le système de pagination car elle hérite cela des interfaces « CrudRepository » et « PagingAndSortingRepository ».

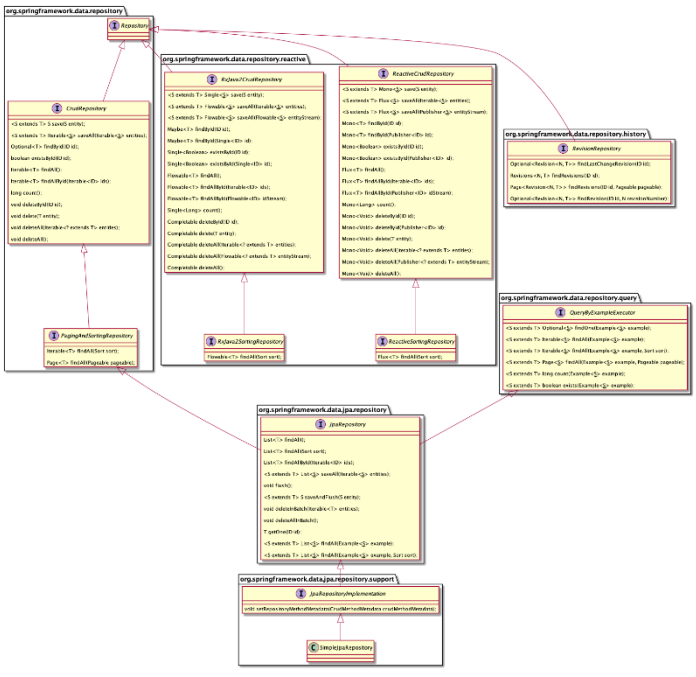


Figure 17 – JpaRepository | Source : « https://i.stack.imgur.com/SBx4b.png »

Vous trouverez ci-dessous dans tableau 7, quelques méthodes spécifiques pour chaque DAO que nous utiliserons le long de ce projet ainsi que des requêtes personnalisées concernant les données à récupérer :

Tableau 7 - Méthodes des DAO

Méthodes	Explication
Page<Device> findAll()	Retourne une page de device. Nous aurons besoin des informations concernant la taille de la page à fournir et l'élément par lequel il faudra commencer. Ces informations pourront être fournies directement dans l'url (size = 3&start = 5).
Device findById(int id)	Retourne le device ayant pour ID celui fourni en paramètre.
Void save(Device d)	Enregistre le device fourni en paramètre.
`@Query("FROM Thumbnail WHERE personId = ?1 and status = ?2 ")` `List<Thumbnail> findAllUsersUnidentifiedPosts(int perId, char status);`	Retourne tous les posts de l'utilisateur ayant pour id et statut ceux fournis en paramètre.

4.1.4 Création des controllers

Les controllers permettront de faire des traitements lors d'appels http sur une url spécifique. Ils utiliseront les DAO définis précédemment pour effectuer des traitements spécifiques sur les données. Voici sur le tableau 8, les annotations qui seront utilisées pour nos controllers :

Tableau 8 - Annotations des controllers

Annotation	Explication
@RestController	C'est une annotation qui combine les annotations @Controller et @ResponseBody. Grâce à cette annotation nous n'avons plus le besoin d'annoter chacun des méthodes de notre controller avec l'annotation @ResponseBody. Il s'occupera de retourner les éléments en JSON ou en XML [12].
@GetMapping(path = «/snakes », produces = « application/json »)	Permet de mapper les requêtes http de types GET sur « /snakes » à une méthode de notre controller. De plus, « produces » définit le format des données qui seront retournées par cette méthode.
@PostMapping(path = «/snakes », consumes = « application/json »)	Permet de mapper les requêtes http de types POST sur « /snakes » à une méthode de notre controller. De plus, « consumes » définit le format des données qui est accepté par cette méthode.

4.2 Réalisation des stories

4.2.1 Poster une photo de serpent

Tableau 9 - Enregistrement de la photo d'un serpent

MOSCOW	Story	Difficulté
Must	En tant qu'expert en serpent / personne mordue je souhaite pouvoir prendre une photo d'un serpent afin de le faire identifier.	7

4.2.1.1 Identification des tâches

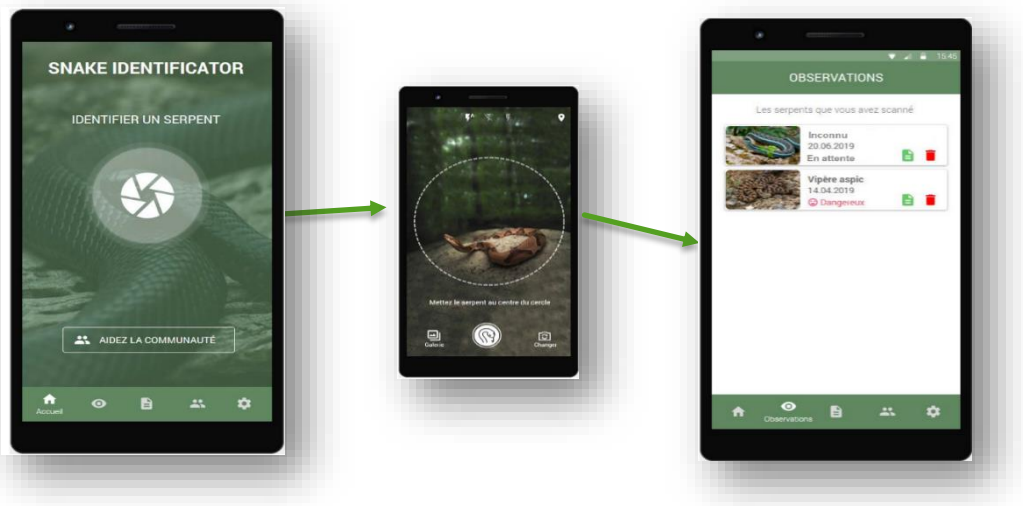
- **Prendre une photo de serpent et la publier**

Comme illustré sur la figure 18, l'utilisateur devra tout d'abord pouvoir prendre une photo depuis son appareil photo. Celui-ci aura accès à cette fonctionnalité depuis l'onglet « Accueil » dans lequel se trouvera un bouton « Identifier un serpent ».

Il sera ensuite redirigé vers l'appareil photo depuis lequel il pourra prendre une photo.

Enfin, une fois la photo validée, celle-ci sera envoyée sur le serveur et l'utilisateur sera redirigé vers l'onglet « Observation » depuis lequel il pourra voir ses publications ainsi que leurs résultats si les votes sont terminés.

Figure 18 - Enregistrement de la photo d'un serpent non-identiifé



- **Gérer l'identité de l'utilisateur afin d'afficher les posts qu'il a publié**

Étant donné qu'il n'y aura pas de login pour gérer l'identité de la personne et que nous avons besoin de pouvoir identifier une personne afin d'afficher les données qui la concernent, nous allons utiliser le concept de Unique Device ID (UDID) côté front-end. Qu'est-ce qu'un unique device id ? C'est un ID unique qui permettra d'identifier chaque appareil. Il y a plusieurs avantages à cela, le premier étant que l'id ne sera pas changé ou supprimé même après une mise à jour de l'appareil ou de l'application. De plus, l'id restera inchangé même après installation/désinstallation de l'application [13].

○ Configurer et enregistrer l'application pour recevoir les notifications

Lorsqu'il y aura un nouveau serpent à identifier, il faudra notifier l'ensemble de tous les spécialistes. Pour ce faire, je vais utiliser le système de push notification de Firebase. C'est une plateforme de développement d'application mobile qui fournit plusieurs services dont le « Cloud Messaging » qui permet d'envoyer des notifications à un appareil ou à un groupe d'appareils.

Comment ça marche ? Lors d'un événement spécifique, il est possible de personnaliser les notifications que l'on veut envoyer et les transmettre à Firebase qui s'occupera de notifier les appareils concernés. On peut personnaliser plusieurs caractéristiques d'une notification comme par exemple le titre, le corps du message ou bien encore le son de la notification et les destinataires [14].

Dans le cas où l'on souhaite notifier un appareil spécifique, il est possible de le notifier via son token. Chaque appareil possèdera un token unique avec lequel on pourra le contacter. Celui-ci est généré par le plugin Firebase Cloud Messaging (FCM). Si à l'inverse on souhaite notifier un ensemble d'appareils, il faudra que ceux-ci « s'abonnent » à un topic (sujet), par conséquent lorsque le serveur enverra une notification sur un topic, tous les appareils abonnés à ce topic recevront la notification comme illustré dans la figure 19.

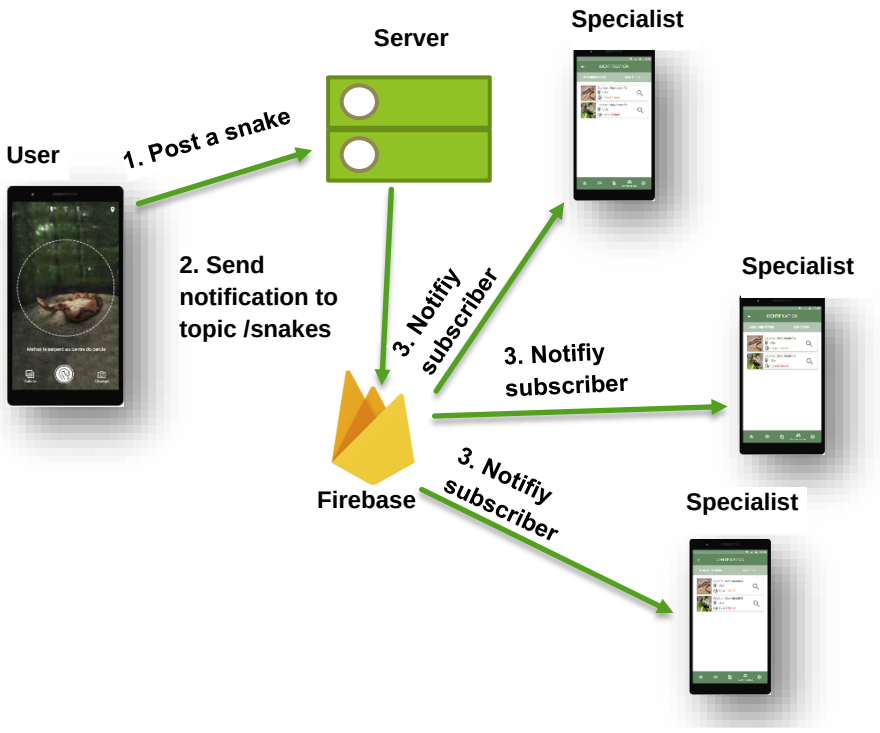


Figure 19 - Notification des spécialistes

- **Enregistrer un post avec la photo une fois celle-ci validée**

Une fois la photo validée, il faudra créer un post avec la photo concernée. Celui-ci sera visible sur l'onglet « Observations » depuis lequel un utilisateur pourra voir la liste de tous les serpents qu'il a scanné.

- Pour cela, il faudra envoyer au back-end via une requête HTTP de type POST à l'adresse « http://adresseDuServeur:8080/posts » avec l'UDID de l'appareil dans le header « Parameter », les paramètres suivants :
  - Les coordonnées X de la localisation l'appareil
  - Les coordonnées Y de la localisation l'appareil
  - La photo du serpent

- **Notifier les spécialistes lors d'un nouveau post de serpent**

Enfin, lorsque le post aura été publié, il faudra envoyer une notification via le service Firebase Cloud Messaging aux spécialistes afin que ceux-ci aillent identifier le serpent en question.

4.2.1.2 Implémentation de la story

Front-end

Les plugins suivants ont été installés afin de pouvoir mettre en place les spécifications définies au point précédent :

- Cordova-Plugin-Unique-Device-Id permettant de générer l'unique device id de l'appareil.
- Cordova-Plugin-Fcm à l'aide duquel un appareil pourra s'abonner à un topic et recevoir des notifications.
- Cordova-Plugin-Camera afin de pouvoir prendre une photo.
- Cordova-Plugin-Gelocation afin de récupérer les coordonnées X et Y de l'appareil

- **Prendre une photo depuis l'appareil mobile**

Voici sur la figure 20, la page d'accueil de l'application avec le bouton au centre de l'écran qui permettra de prendre une photo depuis l'appareil si on clique dessus. Sur la deuxième image se trouve la fonctionnalité de l'appareil photo. Si l'utilisateur clique sur OK la photo est validée, sinon il peut prendre une autre photo. Et la troisième photo contient la page des observations avec la photo que l'utilisateur vient de poster.

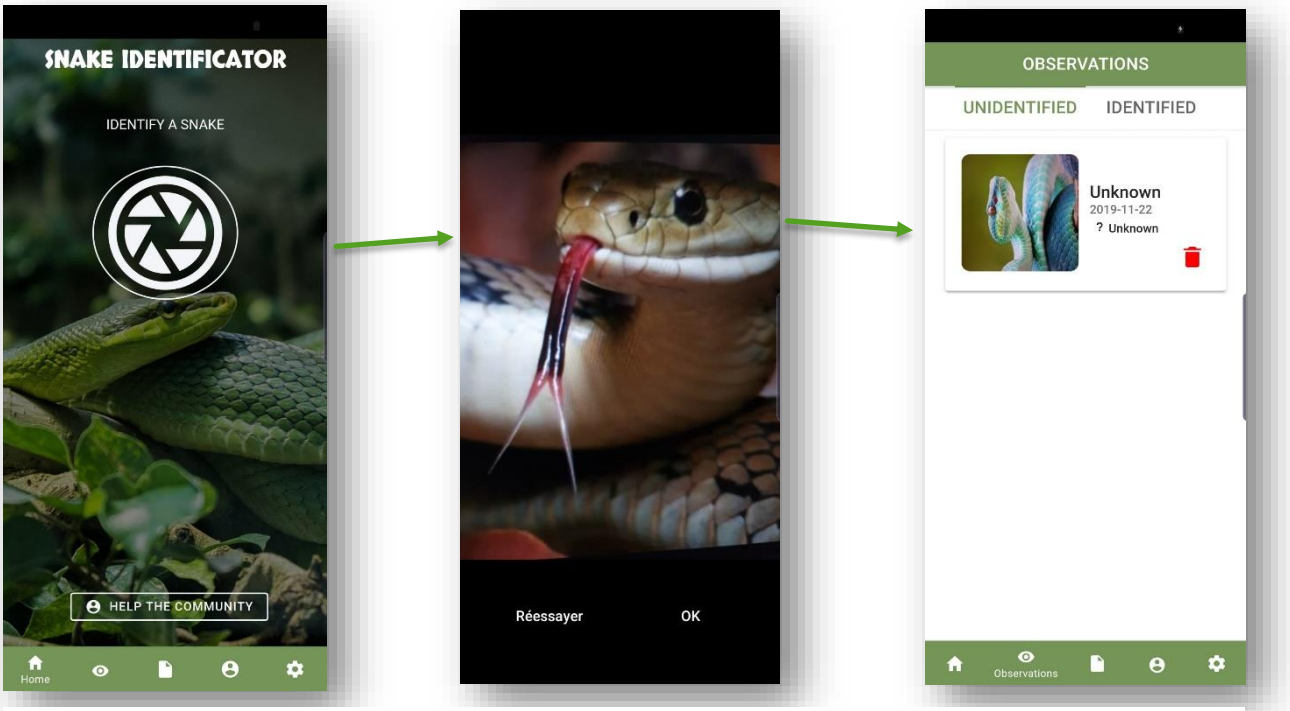


Figure 20 - Résultat de l'enregistrement de la photo d'un serpent non-identifié

- **Notifier les spécialistes**

Ci-dessous dans la figure 21, une capture d'écran de la notification qui sera envoyée à tous les spécialistes lorsqu'il y aura un nouveau serpent à identifier.

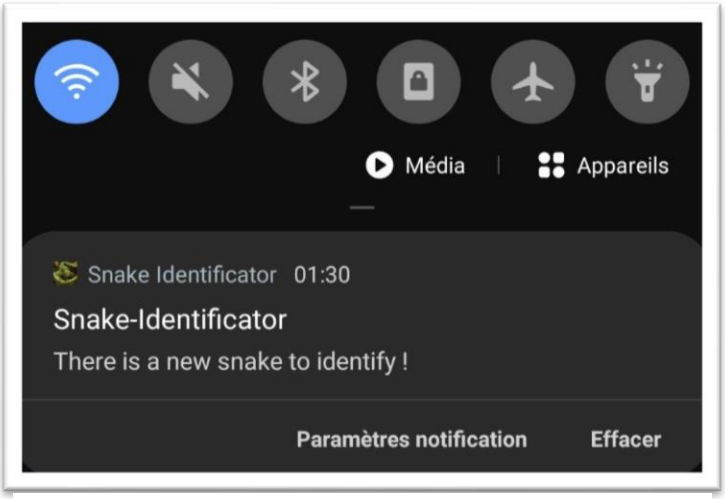


Figure 21 - Exemple de notification

Back-end

○ Enregistrer un nouvel appareil à partir du UDID

Lorsqu'un utilisateur aura installé l'application, celui-ci enverra son UDID et son token au niveau du back-end. Si l'appareil n'est pas déjà enregistré dans la base de données, il l'ajoute, sinon il ne fait rien.

Figure 22 - Cas utilisateur déjà enregistré

```
Hibernate: select parameter0_.param_id as param_id1_1_,
Hibernate: select device0_.dev_id as dev_id1_0_, device0
This device is already registered
```

Sur la figure 22, voici ce qu'il se passe au niveau du serveur lorsqu'un utilisateur déjà existant tente de s'enregistrer.

○ Enregistrer la photo d'un serpent non-identifié

Étant donné que les images sont enregistrées en base 64, lorsqu'on en reçoit une je créé aussi une icône de celle-ci (ayant une dimension de 150px sur 150px) comme illustré sur le point 2 de la figure 23.

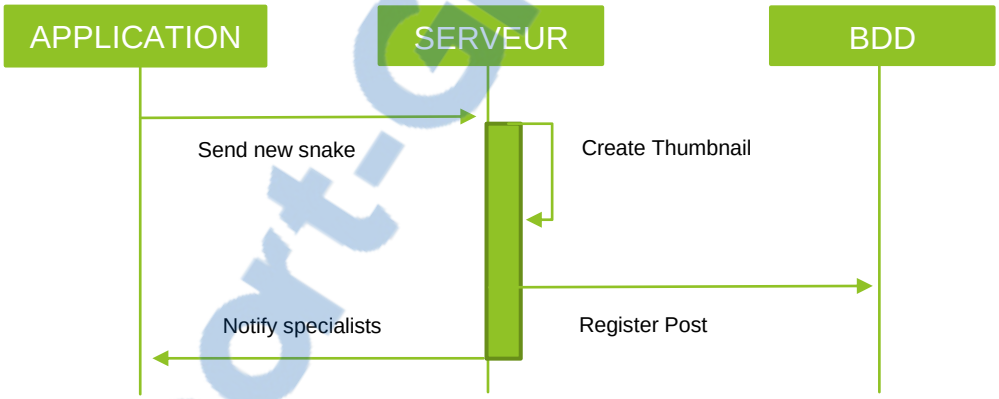


Figure 23 - Cas d'un nouveau post non-identifié

Pourquoi créer une icône ? Cela permettra de réduire la charge de la bande passante lorsqu'un client voudra récupérer un ensemble de publications de serpents car au lieu de retourner les images pour chacune des publications, seules leurs icônes seront retournées. L'entièreté de la représentation sera retournée uniquement lorsqu'un utilisateur souhaitera voir le détail de l'une des publications.

4.2.1.3 Problèmes rencontrés

- **Stockage des images**

La première version concernant la gestion des images consistait à garder les images sur l'appareil d'un utilisateur et de les stocker directement sur le serveur et non pas dans la base de données. Ainsi, lorsqu'un ensemble de serpents identifiés ou non-identifiés étaient retournés par le serveur, ils auraient eu ce format :

```
{
    "id": 1,
    "personId": 1,
    "date": "2019-11-04",
    "coordinateX": 53,
    "coordinateY": 10,
    "image": adresseDuServeur/image1.jpg
    …
}
```

Toutefois, il y avait un risque de surcharger le serveur. En effet, pour chaque post de serpent qui devait être affiché, il aurait fallu exécuter une requête http sur l'url de l'image pour la récupérer. Pour éviter cela, lorsqu'une photo de serpent est publiée, je l'enregistre en base 64 sur le serveur mais j'ajoute aussi une version plus petite que l'originale qui sera retourné à la place de l'originale lorsque je dois retourner plus d'un serpent identifié ou non-identifié.

- **Gestion des push notifications**

J'ai rencontré des difficultés lors de la mise en place des notifications. En effet, pour pouvoir utiliser le service Firebase Cloud Messaging, il faut enregistrer l'application sur la console de Firebase et effectuer des étapes de configuration consistant à rajouter des lignes de code dans des fichiers spécifiques. N'ayant pas modifié les fichiers au bon endroit, les notifications ne fonctionnaient pas. De plus, après une certaine période d'essais, j'ai aussi découvert que l'émulateur que j'utilisais pour tester mon application ne supportait pas les push notifications. Par conséquent, il a fallu tester les notifications sur mon appareil mobile.

4.2.2 Afficher les serpents identifiés et non-identifiés

Tableau 10 - Affichage ensemble des serpents identifiés et non-identifiés pour un utilisateur ou un spécialiste

MOSCOW	Story	Difficulté
MUST	En tant spécialiste je souhaite accéder à la liste des serpents identifiés et non-identifiés afin de les identifier.	5

MUST	En tant qu'utilisateur je souhaite accéder à la liste des serpents identifiés et non-identifiés que j'ai publié.	5

4.2.2.1 Identification des tâches

- **Afficher la liste des serpents identifiés et non-identifiés pour les spécialistes et pour les utilisateurs**

Lorsque le spécialiste aura cliqué sur le bouton « Communauté », la page communautaire lui sera affichée avec le nombre de serpent en attente d'identification s'il y en a. Puis quand il aura cliqué sur « identifier un serpent » celui-ci sera redirigé vers l'onglet « Identification » contenant la liste des serpents non-identifiés et identifiés comme illustrés ci-dessous sur la figure 24.

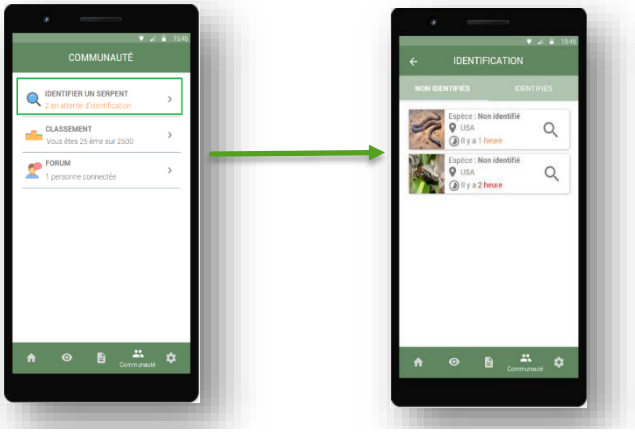


Figure 24 - Affichage des serpents non-identifiés pour un spécialiste

Comme illustré sur la figure 25, lorsque l'utilisateur aura cliqué sur l'onglet « Observations », il se verra redirigé sur cette page contenant tous les serpents qu'il a fait identifier. Toutefois, j'ai rajouté un onglet pour les serpents non-identifiés car il est important qu'un utilisateur puisse voir en tout temps les résultats et l'avancement des votes.

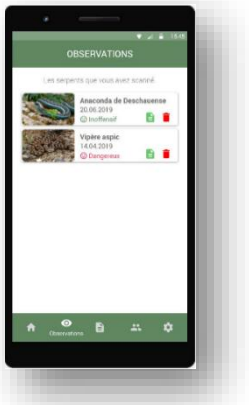


Figure 25 - Affichage des serpents non-identifiés pour un utilisateur

4.2.2.2 Implémentation de la story

Front-end

Pour cette partie, aucun plugin supplémentaire n'a été ajouté. Ci-dessous dans la figure 26, se trouve la page communautaire pour les spécialistes. S'il n'y a aucun serpent à identifier la page du haut sera affichée avec pour message « There are no unidentified snakes ».

Si au contraire il y a un ou plusieurs serpents non-identifiés, la page du dessous sera affichée avec pour message « x waiting for identification ». Enfin, Lorsque le spécialiste aura sélectionné l'onglet « Identify a snake », il sera redirigé sur la page « Identifications » et s'il y en a, les serpents identifiés ou non-identifiés seront affichés.

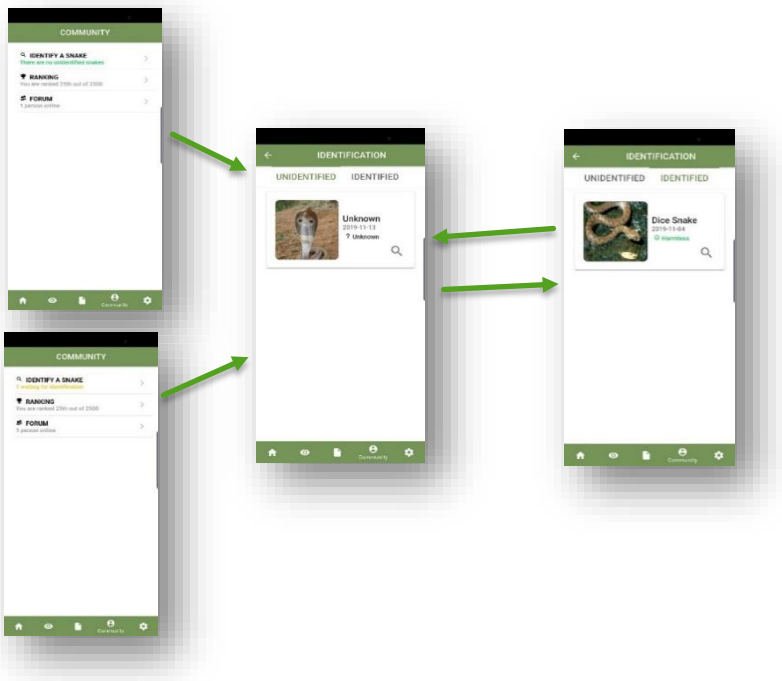
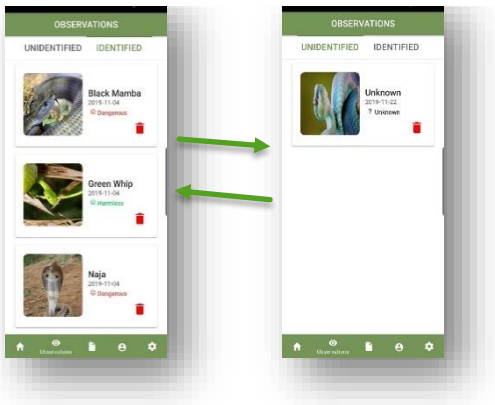


Figure 26 - Résultats des serpents identifiés et non-identifiés pour un spécialiste

Concernant la page « Observations », lorsque l'utilisateur aura accès à cette page lorsque celui-ci aura cliqué sur l'onglet « Observations ». Cette page contient les serpents identifiés et non-identifiés que celui-ci a posté. La figure 15 ci-dessous contient une illustration de ces deux pages.

Figure 27 - Résultats des serpents identifiés et non-identifiés pour un utilisateur



Back-end

- **Afficher les serpents identifiés ou non-identifiés pour un utilisateur ou un spécialiste**

Comme illustré sur la figure 28, lorsqu'un utilisateur ou un spécialiste souhaite récupérer les serpents identifiés ou non-identifiés, on regarde tout d'abord à partir de son unique device id s'il a les droits pour récupérer ces éléments. Si oui, le serveur retournera les éléments demandés sinon il déclenchera une Unauthorized Exception.

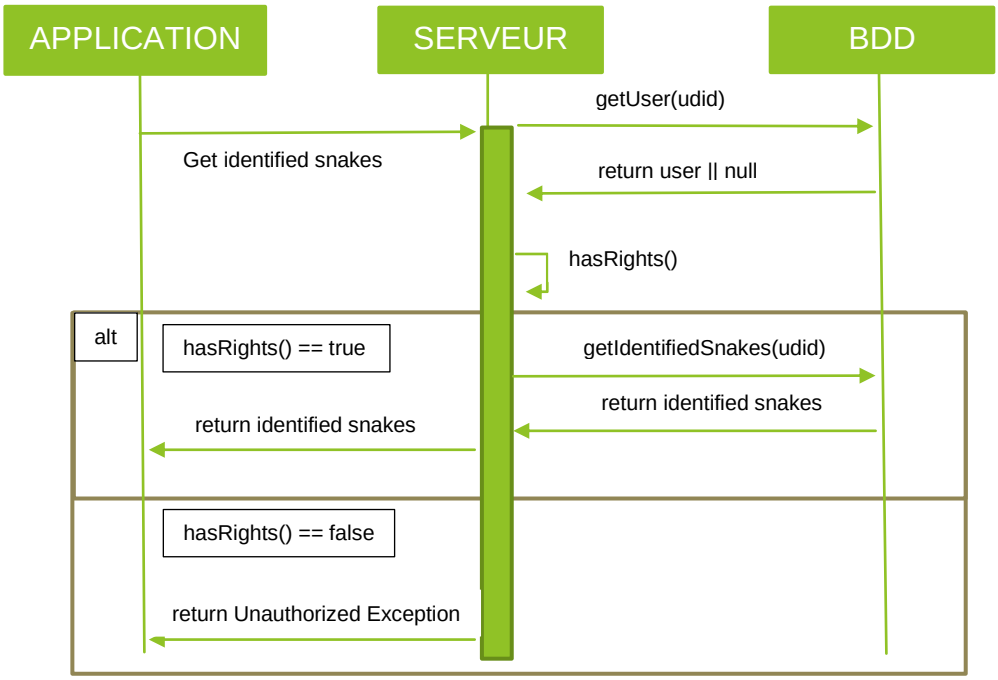


Figure 28 - Retourner les serpents identifiés ou non-identifiés

4.2.2.3 Problèmes rencontrés

- **Activation et désactivation des boutons sélectionnés**

J'ai rencontré des difficultés durant la gestion des états des deux boutons « Unidentified » et « Identified ». Le problème principal provenait lorsque je cliquais sur le deuxième bouton. En effet, les deux boutons restaient sélectionnés et l'affichage des serpents était incorrect.

4.2.3 Afficher le résultat des votes d'un serpent

Tableau 11- Affichage des résultats des votes pour un serpent

MOSCOW	Story	Difficulté
MUST	En tant qu'expert en serpent / personne mordue je souhaite voir le résultat de l'identification du serpent que j'ai publié afin de le transmettre aux secours.	5

MUST	En tant que spécialiste je souhaite accéder à la liste des posts d'identification de serpents auxquels j'ai voté afin de savoir mes résultats	5

4.2.3.1 Identification des tâches

- **Afficher les résultats des votes par rapport à un serpent**

Comme illustré sur la figure 29, cette page aura pour but d'afficher les résultats des votes pour un serpent posté par un utilisateur. Le résultat sera composé des 3 serpents les plus votés par ordre décroissant.

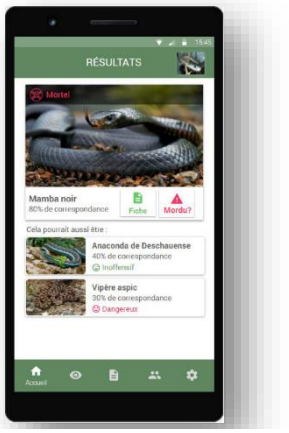


Figure 29 - Résultats des votes

4.2.3.2 Implémentation de la story

Front-end

Pour cette partie, aucun plugin supplémentaire n'a été ajouté. Ci-dessous, la figure 30 illustre la page de résultat des votes pour un serpent. Le premier cas compte plusieurs réponses possibles tandis que le 2ème n'a eu de votes que pour un seul serpent.

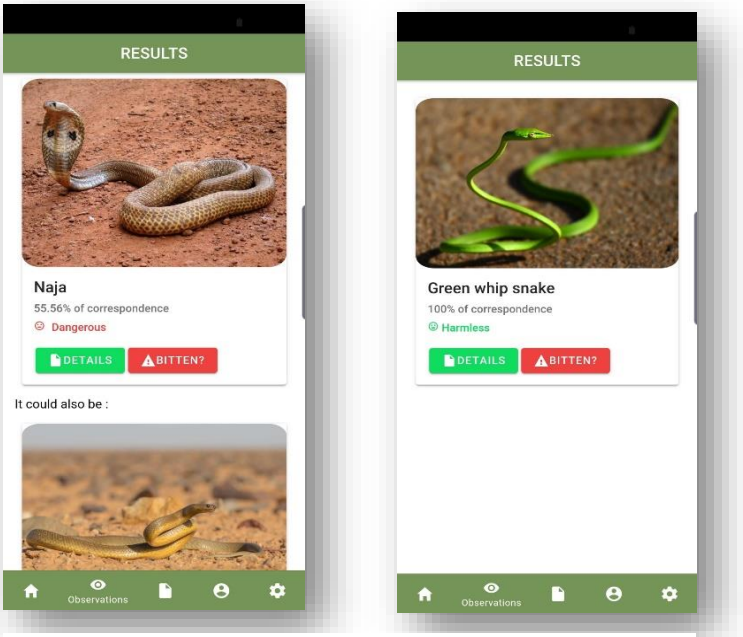


Figure 30 - Implémentation résultats des votes

Back-end

- **Afficher les votes d'un serpent identifié ou non-identifié**

Comme illustré sur la figure 31, lorsqu'un utilisateur ou un spécialiste souhaite voir les votes pour un serpent identifié ou non-identifié, on regarde tout d'abord à partir de son unique device id s'il a les droits pour récupérer cet élément. Si oui, le serveur retournera les éléments demandés sinon il déclenchera une Unauthorized Exception.

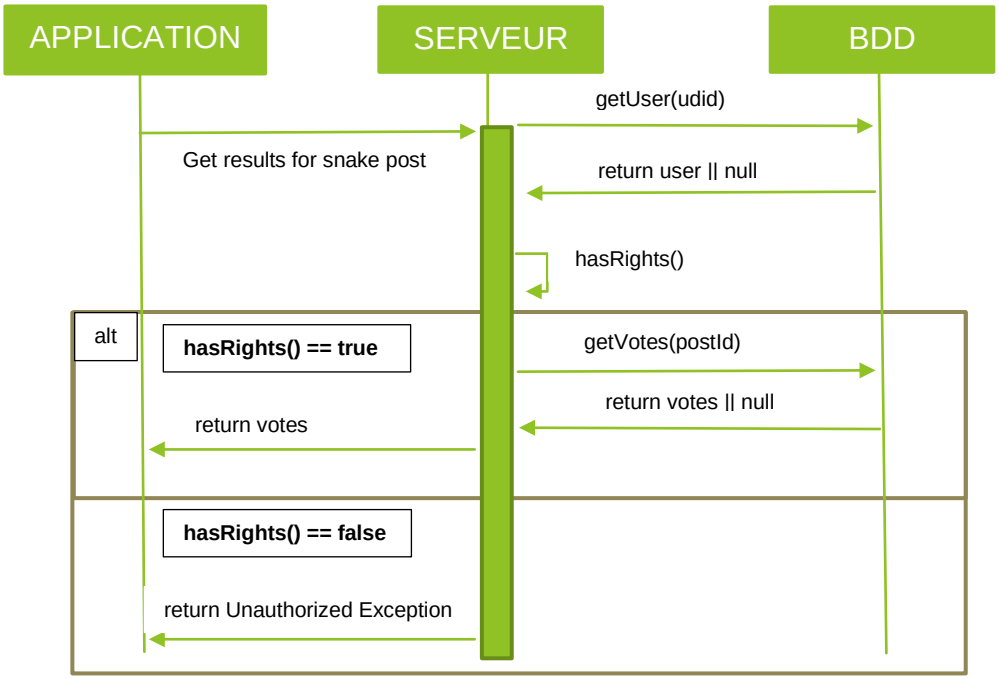


Figure 31 - Afficher les votes d'un serpent identifié ou non-identifié

4.2.3.3 Problèmes rencontrés

Je n'ai rencontré aucun problème particulier durant l'implémentation de cette story.

4.2.4 Enregistrer le vote d'un spécialiste

Tableau 12 - Voter pour identifier un serpent

MOSCOW	Story	Difficulté
MUST	En tant que spécialiste je souhaite pouvoir voter sur un post de serpent afin de l'identifier	5

4.3.4.1 Identification des tâches

- **Afficher les choix possibles de vote et enregistrer le vote du spécialiste.**

Comme illustré à la figure 32, une fois que le spécialiste aura sélectionné une espèce non-identifiée, celui-ci se verra redirigé sur le détail de cette publication dans laquelle il pourra sélectionner le serpent qui lui semblera le plus adéquat et valider son vote.

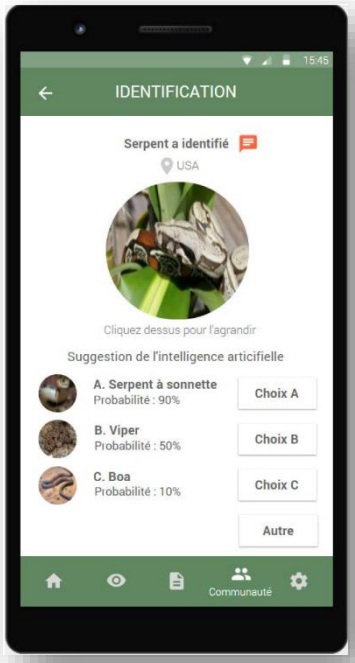


Figure 32 - Page de vote pour un serpent

Une fois que celui-ci aura voté, son vote sera envoyé au back-end via une requête HTTP de type POST à l'adresse http://adresseDuServeur:8080/votes les paramètres suivants avec l'UDID de l'appareil dans le header « Parameter » :

- L'ID du serpent sélectionné
- L'ID du post sur lequel il a voté

4.3.4.2 Implémentation de la story

Front-end

Pour cette partie, le plugin de cordova native-geocoder a été ajouté. Il permet récupérer le nom d'une région à partir des coordonnées X et Y. Ci-dessous, la figure 33 illustre la page de des choix de vote pour un serpent. Une fois le serpent sélectionné, le spécialiste doit appuyer sur le bouton « validate » pour enregistrer son vote. De plus, un spécialiste ne peut voter qu'une seule fois pour un serpent non-identifié.

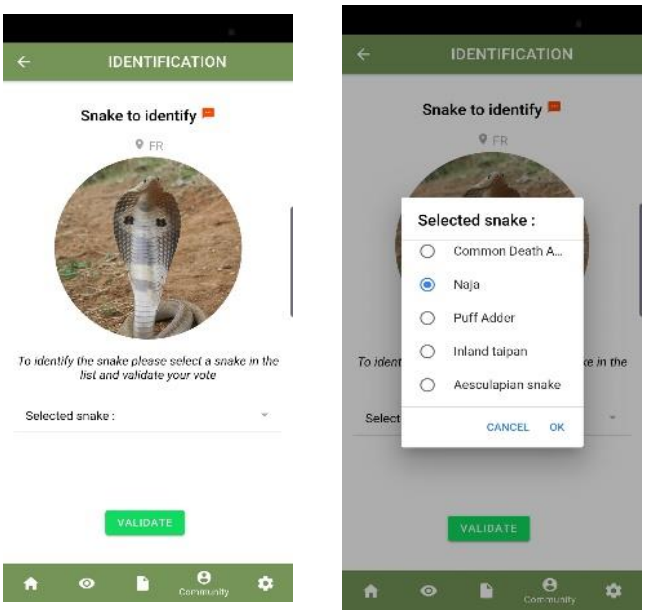


Figure 33 - Implémentation de la page de vote

Back-end

- **Enregistrer le vote d'un spécialiste**

Comme illustré sur la figure 34, lorsqu'un spécialiste aura validé son vote, son choix sera envoyé sur le serveur. Le serveur vérifiera si l'utilisateur a bien le statut de spécialiste avant d'enregistrer sa réponse. S'il a bien les droits, sa réponse sera enregistrée et le compte de vote concernant un post de serpent et un serpent sera incrémenté sinon il déclenchera une Unauthorized Exception.

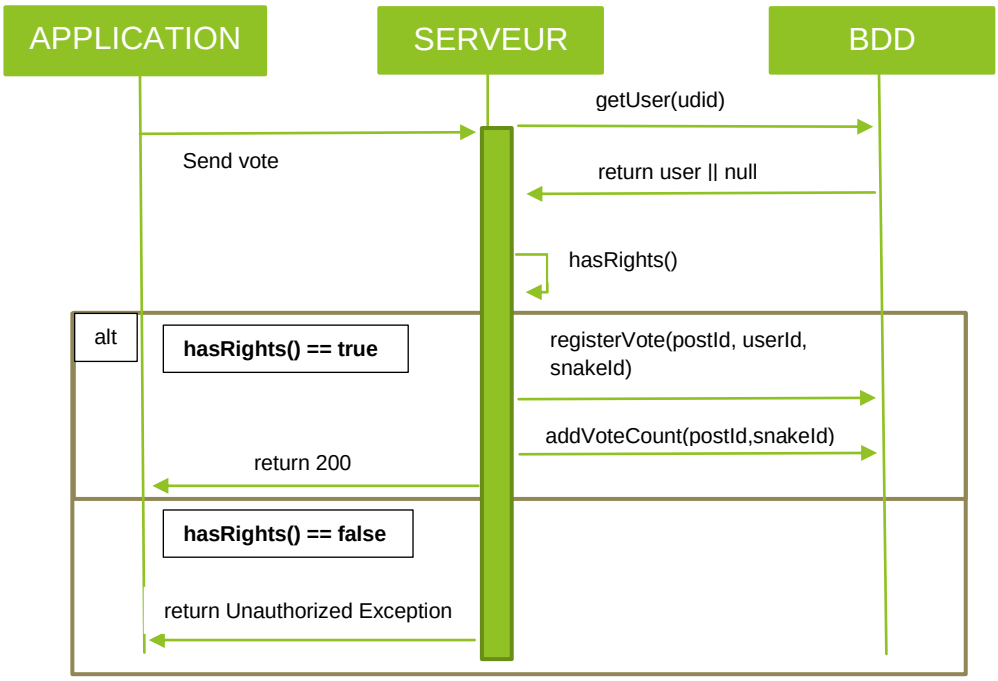


Figure 34 - Enregistrer le vote d'un spécialiste

4.3.4.3 Problèmes rencontrés

Je n'ai rencontré aucun problème particulier durant l'implémentation de cette story.

5. Conclusion

Nous avons vu que la problématique des morsures de serpent est une maladie tropicale négligée. Les gens en général ne sont pas assez informés, il y a un gros manque de données concernant la faune des serpents et la majorité des cas de morsure ne sont pas signalés. Par conséquent, les instituts de santé publique n'ont pas les quantités suffisantes d'anti-venin et ne peuvent soigner réellement qu'une minorité de blessés.

Nous avons développé une solution informatique permettant d'identifier un serpent. Celle-ci est composée de deux grandes communautés, ceux qui postent les photos de serpent (les utilisateurs) et ceux qui identifient les serpents postés (les spécialistes). L'application répond à un besoin immédiat d'identifier un serpent mais permettra aussi une récolte de données sur le long terme. À partir de ces données, des analyses statistiques pourraient être effectuées et cela donnerait notamment une meilleure vue d'ensemble de la répartition des serpents par région et des zones dans lesquelles les morsures sont les plus fréquentes.

Il faut tout de même souligner le fait que les zones les plus concernées par ce problème sont des pays en développement, ce qui veut dire qu'ils n'ont pas forcément les moyens financiers pour posséder un appareil ou tout simplement une connexion internet. Par conséquent, bien que l'application permette de récolter des données, les chiffres et résultats obtenus par de futures analyses ne reflèteront pas forcément la réalité. En effet, la quantité de cas non-signalés aura peut-être baissé mais il reste encore à voir à quel taux celle-ci aura été réduite.

De plus, il serait important d'ajouter quelques améliorations et fonctionnalités à l'application. Il faudrait notamment que l'appareil commence à filmer dès le lancement de l'application. En effet, certaines espèces de serpent sont agressives et rapides. Ainsi lorsqu'elles attaquent il est essentiel de réagir rapidement. Le fait que celle-ci filme dès son lancement pourrait faciliter la tâche à l'utilisateur qui doit avoir une image du serpent en question. Par ailleurs, l'affichage d'un tableau avec la répartition votes de serpents lors de l'affichage du détail d'un serpent identifié peut aussi être utile à l'utilisateur. Enfin, étant donné qu'il n'y a pas de système de login, la possibilité pour un utilisateur de transmettre son « compte » d'un appareil à un autre est importante car dans le cas contraire, si un spécialiste change d'appareil, il n'aurait plus accès aux données qu'il a publié et perdrait son statut de spécialiste.

6. Bibliographie

[1] Elsevier LTD, 2019. Snakebite and snake identification: empowering neglected communities and health-care providers with AI [en ligne]. Septembre 2019. [Consulté le 06 septembre 2019]. Disponible à l'adresse suivante : https://www.thelancet.com/journals/landig/article/PIIS2589-7500(19)30086-X/fulltext#articleInformation.

[2] Jean-Philippe Chippaux. Clinique et traitement des envenimations [en ligne]. 2002. [Consulté le 19 octobre 2019]. Disponible à cette adresse : https://books.openedition.org/irdeditions/10625?lang=fr.

[3] Gaëlle Courcoux. Morsures de serpents : un problème de santé publique en Afrique [en ligne]. Mai 2011. [Consulté le 19 octobre 2019]. Disponible à l'adresse suivante : https://www.ird.fr/la-mediatheque/fiches-d-actualite-scientifique/374-morsures-de-serpents-un-probleme-de-sante-publique-en-afrique.

[4] World Health Organization. Snakebite envenoming [en ligne]. 08 avril 2019. [Consulté le 19 octobre 2019]. Disponible à l'adresse suivante : https://www.who.int/news-room/fact-sheets/detail/snakebite-envenoming.

[5] Pierre-Etienne Joye/ani. Une application permettra d'identifier les serpents à l'origine d'une morsure [en ligne]. 06 juillet 2019. [Consulté le 19 octobre 2019]. Disponible à l'adresse suivante : https://www.rts.ch/info/sciences-tech/10548798-une-application-permettra-d-identifier-les-serpents-a-l-origine-d-une-morsure.html.

[6] Inconnu. Morsures de serpents [en ligne]. 14 novembre 2019. [Consulté le 19 octobre 2019]. Disponible à l'adresse suivante : https://www.ameli.fr/assure/sante/urgence/morsures-griffures-piqures/morsure-serpent.

[7] Inconnu. What is Ionic Framework [en ligne]. Inconnu. [Consulté le 10 septembre 2019]. Disponible à l'adresse suivante : https://ionicframework.com/docs/intro.

[8] Inconnu. Spring Boot [en ligne]. Inconnu. [Consulté le 10 septembre 2019]. Disponible à l'adresse suivante : https://spring.io/projects/spring-boot.

[9] Emily Grace Adiseshiah. Personas, scenarios, user stories and storyboards: what's the difference? [en ligne]. 18 Juillet 2017. [Consulté le 21 août 2019]. Disponible à l'adresse suivante : https://www.justinmind.com/blog/user-personas-scenarios-user-stories-and-storyboards-whats-the-difference/.

[10] Matthew Tyson. What is JPA? Introduction to the Java Persistence API [en ligne]. 02 avril 2019. [Consulté le 12 septembre 2019]. Disponible à l'adresse suivante : https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html.

[11] Médéric Munier. Le modèle DAO [en ligne]. 10 mars 2019. [Consulté le 20 novembre 2019]. Disponible à l'adresse suivante : https://openclassrooms.com/fr/courses/626954-creez-votre-application-web-avec-java-ee/624784-le-modele-dao.

[12] Pankaj. Spring RestController [en ligne]. 28 février 2019. [Consulté le 21 novembre 2019]. Disponible à l'adresse suivante : https://www.journaldev.com/21536/spring-restcontroller.

[13] Chintan Jain. Using Mobile Device IDs as additional factor of authentication (something you have) [en ligne]. 13 Octobre 2017. [Consulté le 21 octobre 2019]. Disponible à l'adresse suivante : https://medium.com/@cjainn/using-mobile-device-ids-as-additional-factor-of-authentication-something-you-have-80b6bd885950.

[14] Doug Stevenson. What is firebase? The complete story, abridged [en ligne]. 25 Septembre 2018. [Consulté le 21 octobre 2019]. Disponible à l'adresse suivante : https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0.

7. Annexes

A. Analyse des risques

Cette analyse de risques aura pour but d'identifier tous les problèmes que je pourrais rencontrer durant ce travail de Bachelor. L'identification de ces risques permettra de réduire leur impact ou leur probabilité (ou les deux) et donc d'assurer au mieux le bon déroulement de ce projet. Pour chaque risque j'ai tout d'abord estimé son impact ainsi que sa probabilité, puis j'ai déterminé son type et enfin j'ai mis en place une mitigation. Voici tout d'abord les catégories de risques qui seront utilisées dans cette analyse.

Risques de développement

Tous les problèmes que je pourrais rencontrer liés au développement de l'application ou aux technologies utilisées seront classés dans cette catégorie.

Risques organisationnels

Toutes les difficultés rencontrées concernant l'organisation ou la planification seront classées dans cette catégorie.

Risques humains

Dans cette catégorie seront classés tous les problèmes liés à la communication, la motivation ou les problèmes personnels.

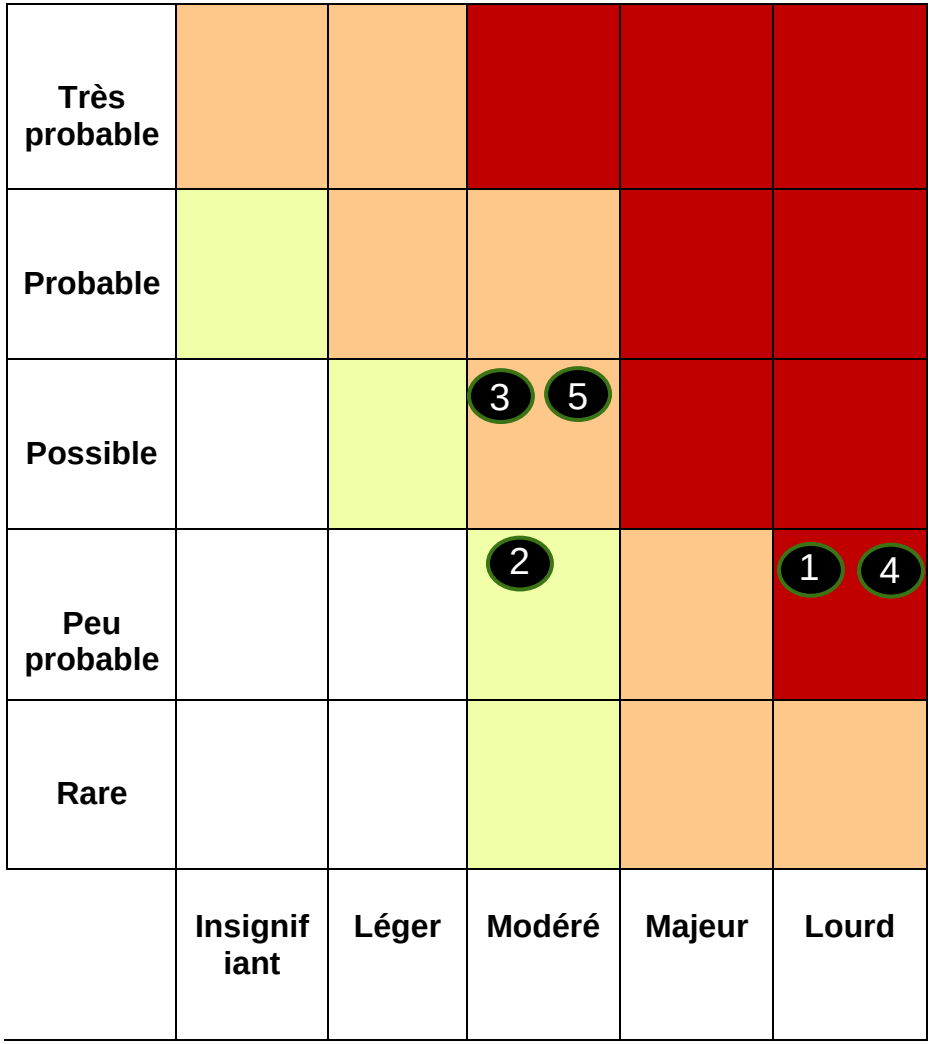
N°	Risques	Description du risque	Type de risque	Risque brut		Mitigation	Risque net / résiduel	
				Impact[1]	Probabilité[2]		Impact	Probabilité
1	Besoins du mandant non satisfaits	Si le mandant n'est pas satisfait lors de la livraison de l'application alors le projet sera un échec.	Développement	5	2	Livrer des parties de fonctionnalités et faire des démonstrations au mandant de façon régulière. Favoriser la communication afin de savoir s'il y a des changements à effectuer.	5	1
2	Bugs dans l'application	Dans le cas où toutes les fonctionnalités n'auraient pas été testées, il pourrait y avoir des bugs lors de la mise en production.	Développement	3	2	Faire des jeux de tests pour chaque fonctionnalité et les exécuter avant et lors de la mise en production.	2	1
3	Technologies utilisées trop complexes	Les outils utilisés pour le développement, tel que le langage utilisé pourrait être trop complexe et causerait du retard sur l'avancement du projet.	Développement	3	3	S'informer sur la meilleure technologie à utiliser et suivre une formation ou des cours à l'avance si nécessaire.	3	1

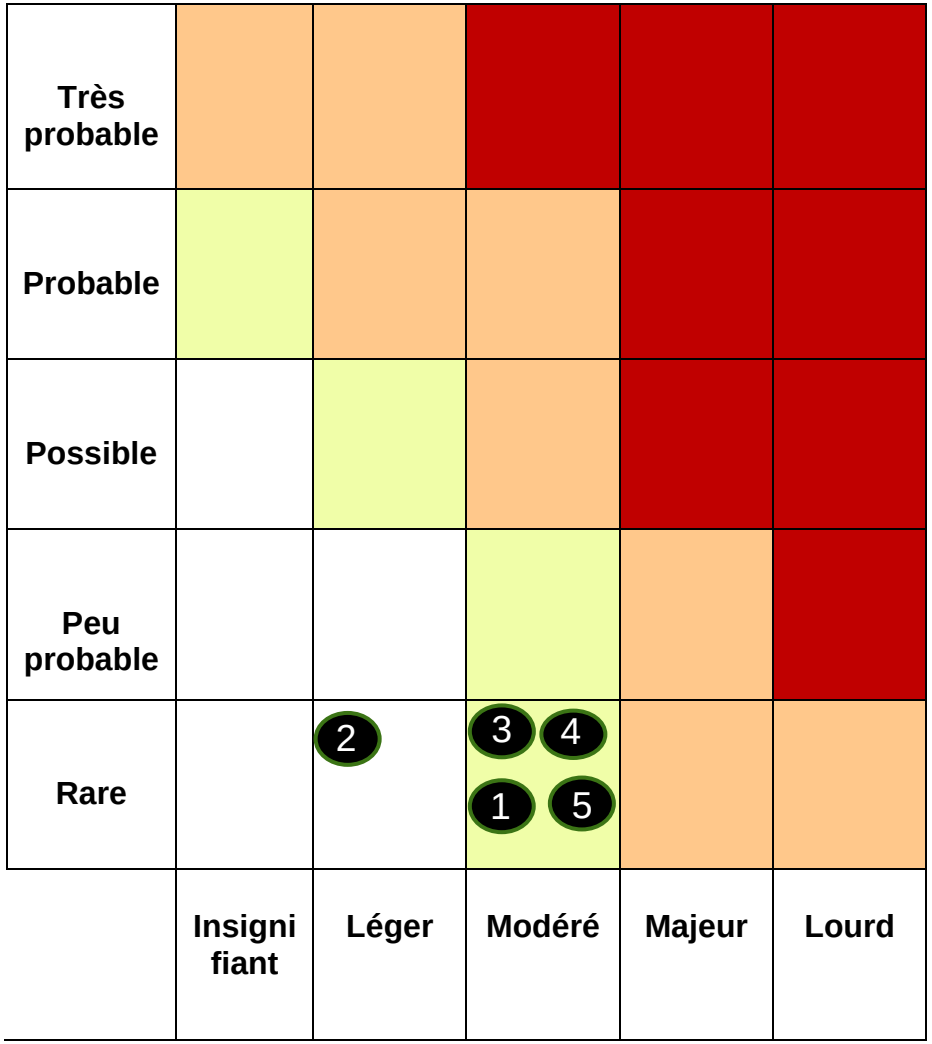
[1] **1** = insignifiant, **2** = léger, **3** = modéré, **4** = majeur, **5** = lourd
[2] **1** = rare, **2** = peu probable, **3** = possible, **4** = probable, **5** = très probable

4	Failles de sécurité de la solution informatique	Si le serveur n'est pas ou est mal protégé contre tout type d'attaque, celui-ci pourrait subir des attaques.	Développement	5	2	Sécuriser le système au mieux et exécuter des tests.	3	1
5	Temps de réponse trop lent	En cas de surcharge, le temps de réponse de notre application pourrait être trop long.	Développement	3	3	Prévoir une architecture et un serveur adapté au type de service proposé afin de supporter la charge conséquente.	3	1
6	Manque de communication	Un manque de communication entre les membres de l'équipe peut causer une baisse de la qualité du travail et donc du retard sur le projet.	Humain	3	3	Favoriser une communication régulière et des mises au points afin de voir l'avancement du travail et proposer des solutions en cas de problème.	2	1
7	Conflit entre certains membres de l'équipe	Il est possible qu'il y ait des conflits entre certains membres de l'équipe.	Humain	3	2	En cas de conflits les autres membres doivent intervenir de manière neutre, régler le problème et réconcilier les personnes si besoin.	3	1
8	Perte de motivation de certains membres de l'équipe	Selon l'avancement du projet, les difficultés du projet ou pour d'autres	Humain	3	2	Encourager la personne, la rassurer et	3	1

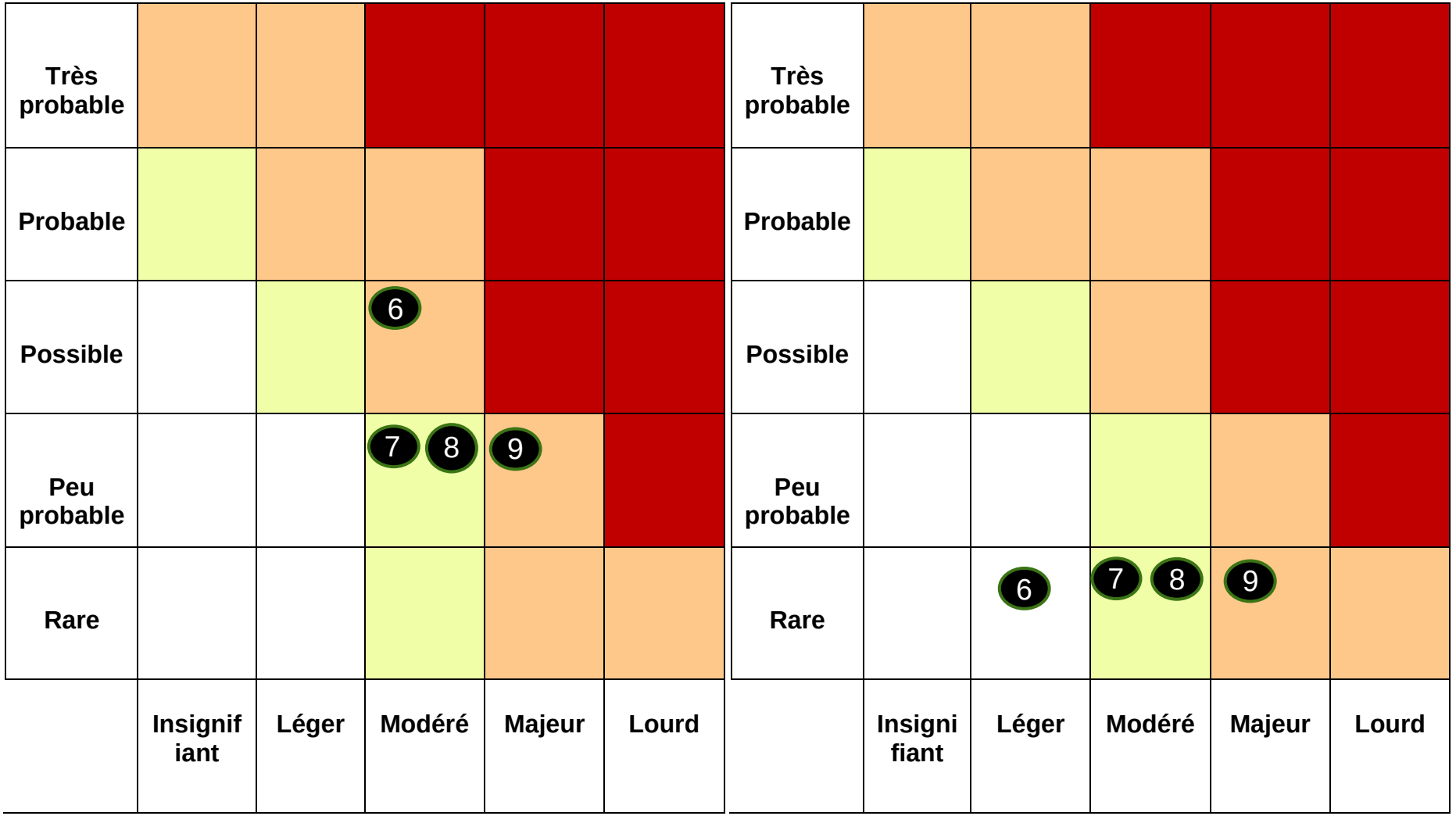
		raisons, certains membres pourraient perdre leur motivation et cela réduirait potentiellement leur qualité de travail.				éventuellement voir s'il y a possibilité d'arranger le problème s'il y en a un. (Il est possible de motiver avec des récompenses)		
9	Abandon de certains membres de l'équipe	Un ou plusieurs membres pourraient partir suite à une perte de motivation ou pour d'autres raisons.	Humain	4	2	Essayer de discuter avec la personne pour voir s'il est possible de la faire changer d'avis.	4	1
10	Retard sur le projet	Si nous rencontrons des problèmes externes ou personnels, nous pourrions prendre du retard.	Organisationnel	3	2	Organiser le travail avec des délais et respecter les délais. Prévoir les problèmes qui pourraient survenir et essayer d'empêcher qu'ils arrivent. Prévoir des mitigations s'ils se produisent.	3	1
11	Mauvaise organisation	Si nous ne définissons pas correctement une méthode de travail commune et des délais, nous pourrions avoir des difficultés au niveau de l'organisation et prendre du retard.	Organisationnel	3	2	Organiser le travail avec des délais et imposer le respect des délais	3	1

Grille des risques de catégorie « Développement » avant et après mitigation :





Grille des risques de catégorie « Humain » avant et après mitigation :



Grille des risques de catégorie « Organisationnel » avant et après mitigation :

Très probable					
Probable					
Possible					
Peu probable			10 11		
Rare					
	Insignifiant	**Léger**	**Modéré**	**Majeur**	**Lourd**

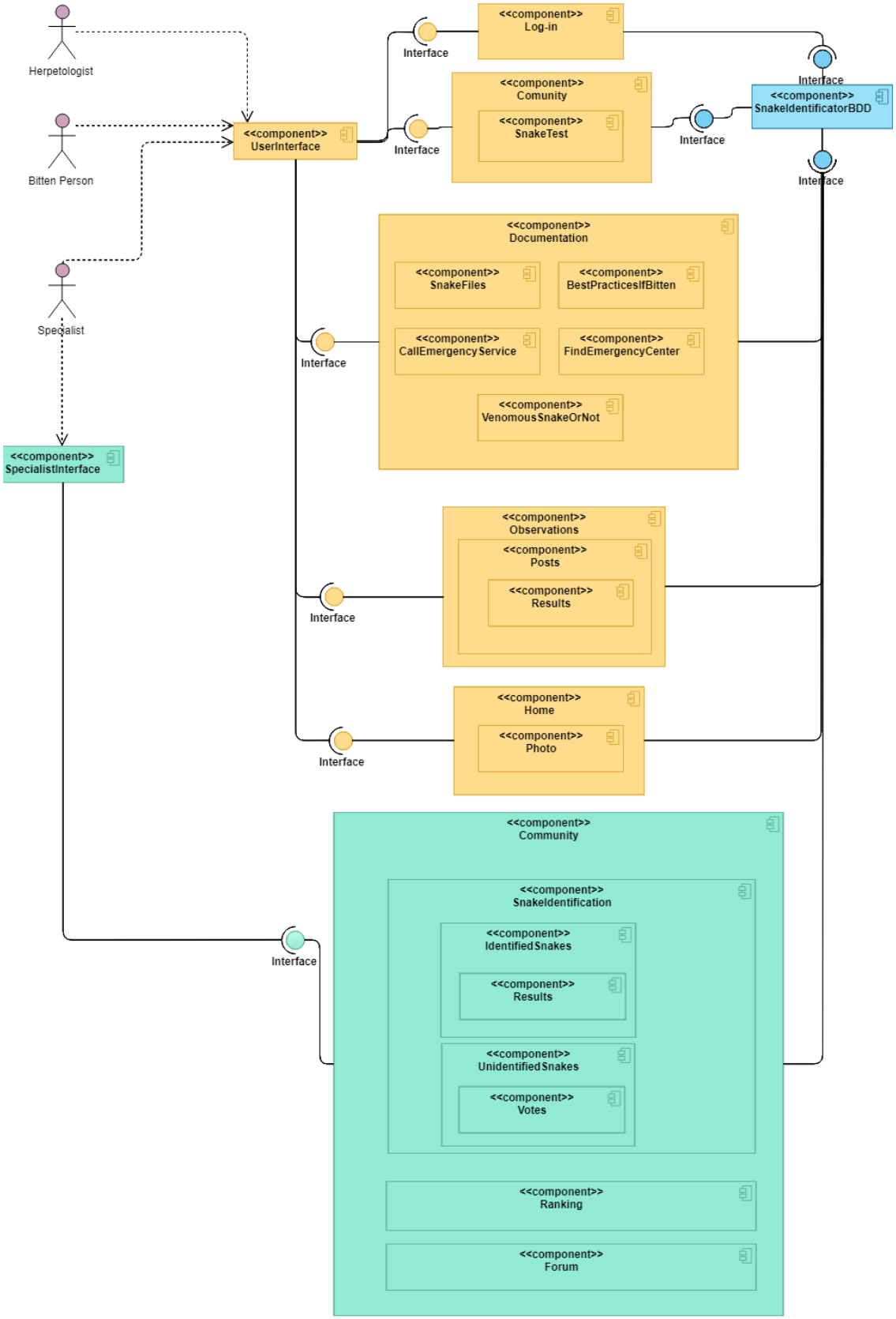
Très probable					
Probable					
Possible					
Peu probable					
Rare			10 11		
	Insignifiant	**Léger**	**Modéré**	**Majeur**	**Lourd**

B. Planification du projet

Activité	Début	Fin	Description
Début du projet	**05.08.2019**		
Semaine 1	05.08.2019	12.08.2019	Planifier le déroulement du projet, définir un Product Backlog et créer des stories
Semaine 2	12.08.2019	19.08.2019	Créer les users personnas, les usecases, le diagramme de composants ainsi que le modèle logique de données
Semaine 3	19.08.2019	26.08.2019	Définir les API, planifier les stories et identifier les tâches pour chaque stories
Semaine 4	26.08.2019	02.09.2019	Effectuer la mise en place l'environnement, créer les tables avec les insertions de données et structurer l'architecture du serveur
Semaine 5	02.09.2019	09.09.2019	Gérer les droits d'accès avec les Unique Device Id, enregistrer une photo de serpent (en base 64), la localisation et envoyer une notification aux spécialistes lors d'un nouveau post
Semaine 6	09.09.2019	16.09.2019	
Semaine 7	16.09.2019	23.09.2019	Retourner de façon paginée la liste des espèces de serpents non-identifiés et identifiés pour un utilisateur et pour un spécialiste, gérer les rôles et
Semaine 8	23.09.2019	30.09.2019	

			les accès et enregistrer le vote d'un vote d'un spécialiste
Semaine 9	30.09.2019	07.10.2019	Revoir le code du back-end, documenter les méthodes de chaque classe pour générer la java doc et ajouter des tests unitaires
Semaine 10	07.10.2019	14.10.2019	
Semaine 11	14.10.2019	21.10.2019	Intégrer les fonctionnalités sur le front-end, ajouter le plugin pour la géolocalisation et le plugin Geocoder pour récupérer le pays à partir des coordonnées X et Y au niveau du front-end
Semaine 12	21.10.2019	28.10.2019	
Semaine 13	28.10.2019	04.11.2019	Récupérer et gérer l'affichage de la liste des espèces de serpents non-identifiés et identifiés pour un utilisateur et pour un spécialiste. Ajouter la page de résultats pour un serpent identifié et la page permettant de voter pour un serpent non-identifié.
Semaine 14	04.11.2019	11.11.2019	
Semaine 15	11.11.2019	18.11.2019	Finaliser le rapport écrit du travail de Bachelor et compléter la documentation des méthodes de chaque classe pour générer la java doc
Semaine 16	18.11.2019	22.11.2019	
Fin du projet		**22.11.2019**	

C. Diagramme de composants



D. Diagramme de classe du projet Spring Boot