# Cahier des charges V1

Valar Morghulis
by
Birna Gérald, Sergent Pierre-Louis & Launay Matthieu

15-06-2020

# Contents

# 1 Introduction

## 1.1 Contexte

Organiser la liberté de la presse, protéger les lanceurs d'alerte est un enjeu démocratique à garantir pour nos sociétés. C'est un sujet brulant del'actualité.

Mais comment déjouer la propagande : pour que cette liberté s'exerce pleinement, si les sources doivent être protégées, elles doivent être aussi vérifiées.

Comment sauvegarder, authentifier, partager, soumettre à vérification des documents, des écrits, des photos, des vidéos sans mettre en danger le lanceur d'alerte.

Suite à l'affaire Snowden, plusieurs ébauches de solution ont été mises sur le marché (SecureDrop). Ces solutions s'appuient toujours sur le réseau de navigation anonymisé TOR.

On peut donc constater qu'il y a un réel besoin de protection et de sécurité pour ces gens, en reprenant l'affaire Snowden notamment, qui souhaitent dévoiler d'importantes informations dans l'intérêt commun sans s'exposer à des représailles.

Mais dans une moindre mesure toute personne qui souhaite faire valoir son droit de liberté d'expression, devrait pouvoir le faire sans passer par des réseaux anonymes tels que TOR.

On remarque aussi que de plus en plus de gens aiment s'informer par eux même, indépendamment des médias *main stream*, par le biais des réseaux sociaux. Cependant cela souvent au détriment de la véracité des propos.

## 1.2 Enjeux

Les enjeux seront donc multiples afin de répondre au mieux à la problématique suivante : "Comment **garantir la sécurité des lanceurs d'alertes**, et comment **garantir au grand public la véracité des informations fournies** ?"

# 2 Description

## 2.1 Objectifs

L'objectif principal de ce projet est donc de créer une plateforme web permettant :

- Le recueil de documents, vidéos, photos (tout types de sources), de manière sécurisé (cryptage des données)
- La vérification des articles par des journalistes ou expert agréé
- La consultation de nouvelles vérifiées par la communauté ou par les experts

### 2.1.1 Stockage *cloud* sécurisé

Un espace de stockage *cloud* qui permettra aux utilisateurs de stocker des articles accompagnés de documents.
Ces données seront cryptées sur le serveur et seront lisibles uniquement avec un système de clefs. Ces articles pourront ensuite être mis à la disposition des médias (présent sur le site), qui vérifieront l'exactitude des informations.

Avec ce premier module on pourra donc établir un lien sécurisé entre les lanceurs d'alertes et les médias. Cette première partie sera donc plutôt portée sur la sécurisation des données et sur la relation lanceur d'alerte - média.

Les *news* ne seront pas publiées directement sur l'espace public du site, mais pourront être relayées par les médias après vérification (par leurs propres moyens). L'utilisateur peut quand même décider à tout moment de publier son article sur l'espace public du site, avec ou sans la validation des médias.

### 2.1.2 *Feed* d'actualité

Alors que la première partie met l'accent sur l'intervention d'un organisme tier afin de vérifier la véracité des informations postés, ce module aura un aspect plus communautaire.

Si certains utilisateurs veulent relayer des *news* moins sensibles, ils pourront alors rendre public l'information directement sur le site. Les médias présent sur la plateforme pourront toujours vérifier certaines informations, et auront un poids plus important sur la *réputation* du post.

Car en effet, les utilisateurs lambda pourront aussi contribuer à vérifier les *news* postés et ce de deux manières :

- Un système de upvote / downvote, à la manière de Reddit
- Une possibilité d'ajouter des sources (liens vers d'autres articles, livres, etc) confirmant ou infirmant les informations données (à la manière de Wikipédia)

Ainsi, sur des informations plus légères, les utilisateurs pourront contribuer à leur tour à la véracité des articles. Bien évidemment la vérification par ce biais est moins fiable, et le verdict des médias aura un poids prédominant.

## 2.2 Utilisateurs

La plateforme comportera deux types d'utilisateurs différents:

- **Utilisateur lambda / lanceur d'alerte**:
  Il s'agit de l'utilisateur de base qui pourra consulter le fil d'actualité, contribuer sur les posts en ajoutant des sources, *upvoter / downvoter* des posts. Et surtout c'est lui qui pourra (de manière anonyme), créer de nouveaux articles pour ensuite les soumettre aux médias ou les publier directement.
- **Utilisateur vérifié / média**:
  Il s'agit d'un média ou d'un journaliste agréé. C'est lui qui aura accès aux articles soumis par les utilisateurs lambda. Il pourra vérifier la véracité des informations avec un poids important. Il pourra aussi vérifier les articles postés publiquement, et ce de manière anonyme.

## 2.3 Fonctionnalités principales

Les fonctionnalités principales ont été quelque peu décrite auparavant. Nous allons donc prendre deux scénario pour mieux comprendre l'enchainement des actions.

### 2.3.1 Scénario 1

Louis est un passionné d'informatique et de développement, c'est pourquoi il a assisté à une conférence sur ReactJS la veille. Durant cette conférence il a appris une nouvelle majeure concernant son *framework* favori, il n'a pas vu d'article à ce propos et il décide donc de partager la nouvelle. Cependant il n'a pas pris de vidéo, il n'a donc aucune preuve de ce qu'il avance. Il décide donc de se rendre sur son site de *news* préféré : ***Valar Morghulis***.

Il y écrit un court article détaillant la nouveauté lié à ReactJS, et post directement l'article sur l'espace public du site. Quelques jours plus tard, il retourne sur la plateforme et constate qu'il a reçu plusieurs *upvote* ainsi que des collaborations pour confirmer ses dires. En effet, plusieurs personnes ont partagé des articles en rapport avec la nouveauté qu'il avait reporté, une personne était même présente à l'évènement et a pris une vidéo qu'elle a pu joindre au post.

Ainsi, Louis a pu lui même vérifier ses dires grâce aux articles communiqués par les autres utilisateurs, et l'information qu'il a communiqué a bénéficié directement à la communauté. De plus, la collaboration de la communauté a permis de confirmer la véracité de sa *news*.

***Résumé des fonctionnalités 1***

- Dépôt d'un article sur l'espace personnel
- Publication d'un article
- Consultation des articles sur le *feed* public
- *Upvote / downvote* - ajout de sources aux posts

### 2.3.2 Scénario 2

Pour ce second scénario on peut reprendre l'affaire Snowden (en atténuant certains points).

Edward Snowden, après des années de travails pour la NSA, décide de dévoiler certaines pratiques de l'organisme qui vont à l'encontre des libertés individuelles et du respect de la vie privée. Il décide d'utiliser la plateforme ***Valar Morghulis*** pour sa discrétion et sa sécurité.

Il y cré alors un article, en joignant une grande quantité de fichiers pour appuyer ses propos. Ces fichiers sont hautement confidentiels, mais il sait que la plateforme crypte les données stockées, et qu'il n'a donc rien à craindre. Compte tenu de l'importance des informations, il va bien évidemment se garder de publier cela sur l'espace public.

En revanche, il va effectuer une sélection de journalistes et de médias grands publics, qui possèdent un compte vérifié sur la plateforme, afin de leur transmettre l'information de manière anonyme (directement sur la plateforme). Une fois cela réalisé, les médias vont ensuite valider la *news* et, suite à un accord, vont transmettre une partie de ces informations au grand public. Les médias décident aussi de tourner une vidéo où Snowden apparait, pour rendre la nouvelle encore plus crédible aux yeux du monde. Bien évidemment, la vidéo est alors stocké en toute sécurité sur la plateforme.

Une fois que toutes les précautions ont été prises, que les vidéos ont été tournés et que tous les médias ont vérifié l'information, Edward Snowden décide de publier son article sur l'espace publique du site. Il a ainsi pu, tout au long du processus, protéger ses données, et ajouter du poids à son propos en récoltant la vérification des différents médias.

***Résumé des fonctionnalités 2***

- Dépôt des documents de manière sécurisé
- Encryptage des documents (système à définir, clef publique/privée)
- Mise à des disposition des documents uniquement aux journalistes et experts vérifiés
- Vérification des informations, validation/invalidation
- Possibilité de mettre l'article sur l'espace public du site avec une pastille de vérification (dans le cas d'une validation)

# 3 Spécifications

## 3.1 Technologies

- **Back-end : API GraphQL**
  Pour l'écriture du back-end nous allons utiliser ***FastAPI***, qui permet d'écrire simplement des API très performantes. FastAPI est un framework python, cela nous sera grandement utile pour la partie cryptage des données.
- **Front-end : ReactJs**
  Pour le côté front-end nous utiliserons ReactJs par affinités et pour rester dans une certaine logique étant donné que GraphQL et ReactJs sont deux technologies développées par ***Facebook***.
- **Infrastructure : Docker**
  Le front-end et back-end seront développés indépendamment, et seront tous deux *dockerisé* afin de faciliter le déploiement. Nous hébergerons le tout chez un *cloud provider* tel que ***Digital Ocean***. Afin de synchroniser les différentes briques applicatives nous utiliserons un simple *docker-compose*.
- **Déploiement : Capistrano**
  Il s'agit d'un outil très pratique qui pourra, en s'appuyant sur le docker-compose, livrer automatiquement les applications sur le serveur. Il permet aussi de gérer les versions de livraisons, et sera utile si nous souhaitons revenir à une version livrée antérieure.

*NB : dans le cadre de la réalisation de ce POC nous utiliserons des serveurs cloud. Dans le cas de la mise en oeuvre véritable du projet, il serait plus judicieux d'utiliser des serveurs privées, afin de garantir au mieux la sécurité des données.*

## 3.2 Fonctionnalités détaillées

Les fonctionnalités seront découpés en plusieurs niveaux de priorités :

- **Obligatoire** (FOB) : absolument nécessaire pour la *MVP (Minimum Valuable Product)* du projet
- **Optionnelle** (FOPT) : pas totalement nécessaire mais peuvent être utile pour avoir un produit abouti
- **Future** (FF) : pas vraiment attendues dans le cadre de la réalisation du projet tutoré, mais souhaitable pour le futur

### 3.2.1 Obligatoire

Le module décrit dans la partie *2.1.1 - Stockage cloud sécurisé* sera développé en priorité. Cette partie est le noyau de l'application.

Les utilisateurs pourront s'inscrire, créer des articles et par ce biais déposer des documents sur la plateforme. Ils pourront ensuite soumettre ces articles aux médias et experts pour vérification.

Les médias ou expert pourront créer un compte sur le plateforme puis demander à être vérifiés pour acquérir un statut particulier, qui leur permettra de recevoir les articles que les utilisateurs lambda veulent faire vérifier.

---

**FOB.1 - Module d'authentification**

L'utilisateur pourra créer un compte, qui aura les droits de base. Il devra effectuer une demande spéciale par mail si il souhaite acquérir un compte vérifié.

- **FOB.1.1 - Inscription**
- **FOB.1.2 - Connexion/déconnexion**

**FOB.2 - Module cloud sécurisé**

Une fois connecté l'utilisateur aura accès à une page qui listera tous les articles qu'il a rédigé. Il pourra accéder aux détails de chacun d'entre eux, télécharger les documents qui sont liés à ces derniers et modifier/supprimer les contenus. Il pourra aussi publier ou soumettre son article à vérification auprès des experts.

- **FOB.2.1 - CRUD Article**
  Chaque article aura la possibilité d'avoir une vignette de certification, qui atteste que l'article a été validé par les médias/experts.
  - ***FOB.2.1.1*** : lister les articles
  - ***FOB.2.1.2*** : créer un article (contenu + fichiers)
    Les fichiers et informations devront être cryptés sur le serveur
  - ***FOB.2.1.3*** : détails d'un article
    - FOB.2.1.3.1 : afficher contenu
    - FOB.2.1.3.2 : télécharger fichiers liés
  - ***FOB.2.1.4*** : modifier un article
  - ***FOB.2.1.5*** : supprimer un article

- **FOB.2.2 - Soumettre article**
  Sélection des médias/experts qui auront accès à l'article ainsi que tous les fichiers joints, et qui pourront attester la véracité des informations. A savoir que si un article est soumis à plusieurs médias/experts, l'article devra recevoir une majorité d'avis favorable pour recevoir la vignette de certification.

---

**FOB.3 - Module de vérification**

Si un utilisateur se fait vérifier par la plateforme il gagne alors le statut de média/expert. De ce fait, il a maintenant accès à une nouvelle page qui répertorie les articles mis à sa disposition pour une demande de vérification. Il peut désormais les consulter et rendre son verdict.

- **FOB.3.1 - Consultation articles soumis**
  - ***FOB.3.1.1*** : lister les articles soumis (similaire *FOB.2.1.1*)
  - ***FOB.3.1.2*** : vue détaillée de chaque article (similaire *FOB.2.1.3*)

- **FOB.3.2 - Validation article**

### 3.2.2 Optionnelle

Le module décrit dans la partie *2.1.2 - Feed d'actualité* sera développé dans un second temps, si et seulement si les fonctionnalités obligatoires ont été satisfaites. Ce module sera donc optionnel, tout comme certaines améliorations qui rendront l'utilisation de la plateforme plus agréable.

- **FOB.2.1 - CRUD Article**
  - ***FOB.2.1.1*** : lister les articles
    - FOPT.2.1.1.1 : filtrer / trier les articles avec différents critères
  - ***FOB.2.1.3*** : détails d'un article
    - FOPT.2.1.3.3 : lire / consulter les documents directement en ligne (sans télécharger)

---

- **FOB.3.1 - Consultation articles soumis**
  - ***FOB.3.1.1*** : lister les articles soumis (similaire *FOB.2.1.1*)
    - FOPT.3.1.1.1 : filtrer / trier les articles avec différents critères (similaire *FOPT.2.1.1.1*)
  - ***FOB.3.1.2*** : détails d'un article
    - FOPT.3.1.2.1 : lire / consulter les documents directement en ligne (sans télécharger) (similaire *FOPT.2.1.3.3*)
  - ***FOPT.3.1.3*** : commentaires / discussion avec l'auteur de l'article
    Le media/expert pourra commenter les articles ou entrer en contact avec le lanceur d'alerte pour demander des informations supplémentaires

- **FOB.3.2 - Validation article**
  - ***FOPT.3.2.1*** : argumenter les raisons de la validation / invalidation
    A la manière de la *FOPT.3.1.3*, le media/expert pourra laisser un commentaire avec le rendu de son verdict

#### 3.2.3 Future

# 4 Réalisation

## 4.1 Critères d'acceptation

Pour un tel projet on peut tenir compte des critères secondaires d'acceptations tels que : la ***rapidité*** et l'***ergonomie***, afin de garantir une expérience fluide.

Cependant le point le plus important sera de garantir un cryptage des données fonctionnel, afin de garantir ***sécurité*** et ***fiabilité*** (pas de perte de données par exemple).

## 4.2 Contraintes

- **Temps** : période de développement assez courte, possibilité de développer uniquement un des deux modules présentés (stockage).
- **Environnement de développement** : travail à distance, ne facilite pas la communication et la coordination des tâches.
- **Environnement technique** : environnement très vaste, avec beaucoup de technologies différentes. Bien insister sur la répartition des tâches en fonction des affinités de chacun.

## 4.3 Points critiques

### 4.3.1 Cryptage des données

L'une des fonctionnalité critique de l'application sera de réussir à *crypter les données* stockées sur le serveur. Il s'agit d'un mécanisme complexe qui est assez nouveau pour l'équipe, aussi bien sur le principe que sur la mise en place techniques (nouveaux outils).

Il sera donc très important de se pencher sur cette problématique très tôt dans la réalisation du projet, afin de choisir la bonne solution technique et la mise en place la plus efficace.

Une possibilité sera de développer d'abord le module d'authentification puis directement le module de dépôt. Plusieurs outils existent afin de nous aider, notamment ***Postman*** qui nous permettra de tester notre API.

Le développement et la resolution de problèmes techniques de ces deux modules se feront en parallèle de l'implémentation du modèle de données classique.

### 4.3.2 API GraphQL

L'écriture de l'API se fera en utilisant ***FastAPI***, avec son implémentation GraphQL. Il s'agit d'une technologie très performante mais assez récente. Il y aura donc une période d'apprentissage, qui se doit de rester courte afin de mener à bien le projet.

L'écriture de cette API va aussi impacter la façon donc les appels depuis le front seront effectués. Ils devraient être grandement simplifiés, mais encore une fois, un petit apprentissage sera nécessaire.

Le bon déroulement du projet sera donc dépendant de la capacité du groupe à s'adapter vite à des difficultés et à bien organiser la partie *monté en compétences* et *développement*.

Pistes en cas de grandes difficultés :

- Revenir à l'utilisation d'un *framework* mieux maitrisé (Django)
- Abandonner l'implémentation GraphQL et développer une API REST
- Revenir sur une architecture plus classique
- Diminuer le temps de travail sur l'automatisation de déploiement et la mise en place de l'infrastructure