Solid'Ark

Création d'une FAQ pour les besoins de l'association

Travail de Bachelor

Werner Schmid

Juin 2022

Supervisé par :

Prof. Dr. Jacques Pasquier-Rocha
Software Engineering Group

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Groupe Génie Logiciel
Département d'Informatique
Université de Fribourg (Suisse)

Remerciements

Je souhaiterais remercier le Professeur Jacques Pasquier pour son encadrement et ses conseils durant ce travail. Je souhaiterais également remercier l'association Solid'Ark pour leur disponibilité et les idées qu'ils ont apportées.

Abstract

L'association Solid'Ark souhaiterais développer un forum pour permettre à ses utilisateurs d'interagir sur leur platforme. Etant l'ami d'un membre de l'association, je suis engagé à les aider dans le développement de celle-ci. Le but de ce travail de bachelor est de définir correctement les besoins des utilisateurs et de créer un prototype. Ceci leur permettra de disposer d'une base solide lorsque les fonds seront levés pour concrétiser le projet.

Mots clés : Symfony, REST, MySQL

Table des matières

Liste des figures

Liste des tableaux

Liste des codes source

1 Introduction

1.1. Présentation de l'association

Solid'Ark est une association fribourgeoise dont le but est de développer une plateforme digitale, décentralisée et permettant à ses utilisateurs d'interagir sur un marché local. Ceux-ci pourront y acheter des titres, financer des entreprises et/ou des entrepreneurs locaux et être rémunérés en fonction de la performance financière de leurs investissements. Les utilisateurs pourront également y consommer des services.

L'objectif à long terme de l'association est la création de marchés régionaux qui s'auto-évalueront de la même manière qu'un marché boursier, en fonction de l'offre et de la demande de titres d'entreprises et du retour sur investissement de leurs projets. Les transactions monétaires se feront à l'aide de crypto-monnaies utilisant la technologie blockchain. Ces monnaies seront indexées sur un jeton développé sur la Smart Chain de Binance et dont une réserve non échangeable sera liée à des actifs intangibles garantissant sa stabilité, comme par exemple des biens immobiliers. Le site de l'association [1] explique de manière plus détaillée leurs objectifs.

Etant ami avec l'une des personnes faisant partie du comité, je me suis engagé à mettre à contribution mes connaissances acquises durant mes cours dans le cadre de l'un de leurs sous-projets. Ceci me permettrait de simultanément fournir un travail m'aidant à avancer dans mes études et de potentiellement contribuer au succès de leur aventure.

1.2. But du travail

L'association souhaiterait, à long terme, travailler de manière collaborative pour mener à bien ses projets. Ceci permettrait de faire participer toute personne souhaitant s'impliquer dans son développement.

[1] Page d'accueil : `https://ico.solidark.org/`

Les participants auront la possibilité de proposer des projets de développement ou de législation pour améliorer la plateforme, comme des idées d'amélioration des fonctionnalités ou du design. Une fois un projet approuvé par la majorité, il sera ensuite possible de le financer, d'y créer des tâches, des cercles de discussion ou de travail, d'y accomplir des tâches, ... On souhaiterait également pouvoir y rémunérer les participants en crypto-monnaies pour leur participation à un projet.

Le forum doit également permettre aux utilisateurs d'y poser des questions dans une FAQ afin d'y éclaircir des points qui ne sont pas clairs et d'y interagir avec les autres. Ces questions devraient pouvoir être adressées à tout participant ou à des groupes restreints parmi ceux-ci, comme la team de l'association, les experts dans un domaine, les membres d'un même projet ou d'un même cercle de discussion.

Mon travail va se concentrer sur la FAQ. Dans un premier temps, celui-ci consistera à modéliser les fonctionnalités requises dans la FAQ et la base de données associée à celui-ci. Dans un deuxième temps, celui-ci montrera l'implémentation d'un prototype de backend à l'aide du framework Symfony. Ce prototype est uniquement une exploration personnelle du framework et ne sera pas mis en production.

Ce qui aura été fait dans ce travail devra permettre à l'association de disposer d'une base conceptuelle et d'une méthodologie qui va faciliter l'avancement de leurs futurs projets.

1.3. Organisation du travail

Le rapport est divisé en quatre chapitres et une annexe.

Introduction

L'introduction explique le but, la mise en forme et la structure du travail.

Besoin des utilisateurs

Ce chapitre explore les différents forums existants, définit les différents rôles que peuvent avoir les utilisateurs et présente les besoins fonctionnels de la FAQ.

Serveur et base de données

Dans un premier temps, ce chapitre explique les concepts associés au développement d'un serveur web (architecture REST, ORM, endpoints, ...) et les technologies utilisées dans ce travail pour le créer (Symfony, Swagger, ...). Il définit ensuite la base de données et les endpoints utilisés pour implémenter le prototype.

Conclusion

La conclusion énumère les points positifs et les difficultés rencontrées dans le cadre de ce travail, ainsi que les points à prendre en compte dans la concrétisation future du forum.

Annexes

L'annexe contient une description détaillée de la base de données et des endpoints créés dans ce travail.

1.4. Structure et mise en forme

— Conventions de mise en forme :

- Abréviations et acronymes : Asynchronous JavaScript And XML (AJAX) lors d'un premier usage et AJAX pour les usages suivants ;
- `http://localhost:9090/eHealthServer` est utilisée pour les adresses web ;
- Référencer du code est formatté de cette façon :

```
1 public double division(int _x, int _y) {
2     double result;
3     result = _x / _y;
4     return result;
5 }
```

- Le travail est divisé en quatre chapitres, qui contiennent eux-mêmes des sections et des sous-sections. Chaque section ou sous-section est divisée en paragraphes, signalant des ruptures logiques.
- Les tableaux, les figures et les codes sources sous numérotés à l'intérieur d'un chapitre : référencer la Figure j du chapitre i sera noté *Figure i.j*.

2
Besoins des utilisateurs

2.1. Aperçu des forums existants

Pour avoir une idée globale de ce qu'on pourrait développer pour l'association, on va tout d'abord explorer les principales plateformes de discussion existantes.

Après la découverte d'un article [5] énumérant les 30 forums les plus populaires, il a été décidé d'explorer et d'expliquer le fonctionnement de trois plateformes pertinentes pour ce projet, qui sont *Reddit*, *Quora* et *StackOverFlow*.

Chaque sous-section sera conclue par un tableau récapitulatif de l'ensemble des fonctionnalités pertinentes de chaque plateforme.

2.1.1. Reddit

Reddit[10] est un «site web communautaire publiant des contenus choisis par les membres eux-mêmes selon un système de vote»[11]. Les contenus sont organisés par thèmes (subreddits), définis comme thèmes par défaut par la plateforme ou créés par les utilisateurs eux-mêmes.

Page d'accueil

La figure 2.1 nous montre un aperçu de l'interface lorsqu'on tombe sur la page d'accueil de *Reddit*. La page web est composée d'un en-tête et d'un fil d'actualité comportant les thèmes les plus en vogue aujourd'hui et d'une liste de postes pertinents pour l'utilisateur. Il est possible de défiler le fil d'actualité vers le bas.

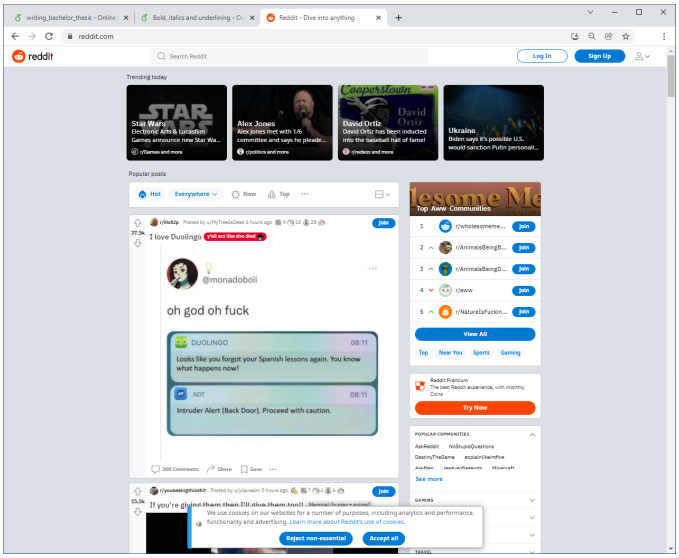

FIGURE 2.1. – Page d'accueil de la plateforme *Reddit*

L'en-tête (figure 2.2) est composé d'une barre de recherche, d'un bouton pour se connecter à la plateforme et d'un autre pour s'enregistrer.

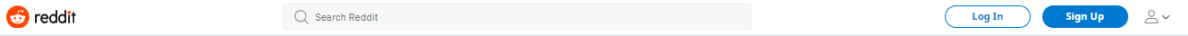

FIGURE 2.2. – Barre de recherche de *Reddit*

La barre de recherche permet de rechercher un thème par son nom ou un utilisateur (figure 2.3) : une liste de suggestions apparait en dessous de celle-ci lorsqu'on commence à taper du texte.

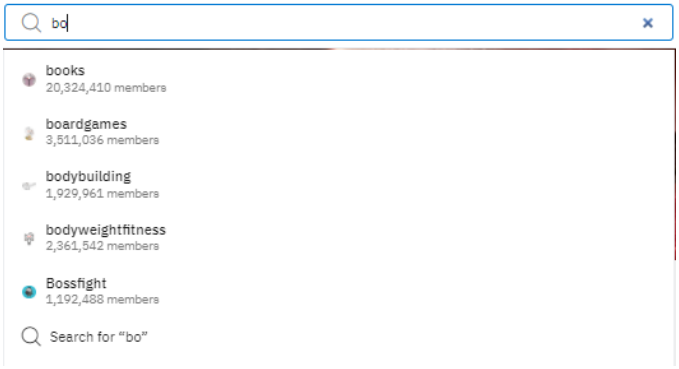

FIGURE 2.3. – Recherche d'un thème ou d'un utilisateur sur *Reddit*

La page d'accueil d'un thème (figure 2.4) ne diffère pas énormément de la page d'accueil principale : elle comporte une liste de postes en lien avec le thème recherché. L'en-tête du menu principal est néanmoins différent : on peut y voir le nom du thème ainsi qu'un bouton permettant de le rejoindre.

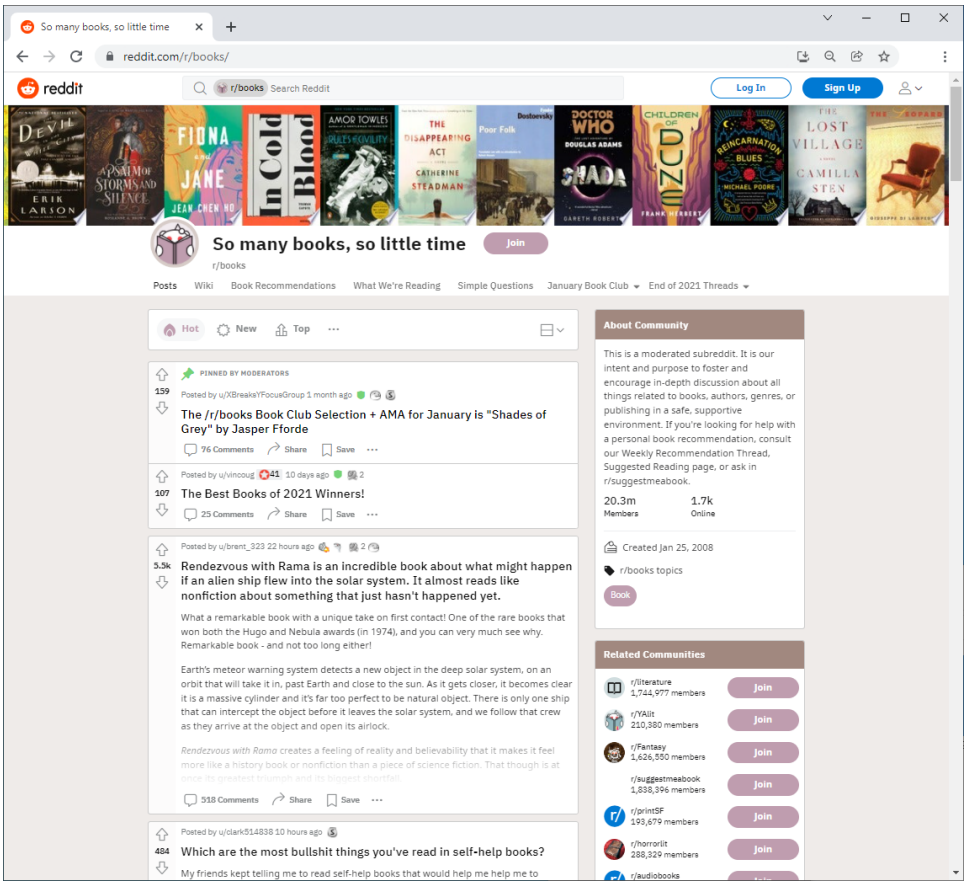

FIGURE 2.4. – Page d'accueil d'un thème sur *Reddit*

Fonctionnalités de l'utilisateur

Lorsqu'un utilisateur s'inscrit sur la plateforme et se connecte pour la première fois, il sera automatiquement abonné aux thèmes par défaut de *Reddit* et les postes les plus pertinents associés à ceux-ci seront visibles sur sa page d'accueil. L'utilisateur a ensuite la possibilité de procéder à une personnalisation : il peut décider de se désabonner de ces thèmes par défaut et de s'abonner à ceux de son choix. Ceci lui permettra de voir défiler uniquement les postes en lien avec les abonnements qu'il aura souscrits.

Il est également possible de modifier la manière dont le tri des postes *Reddit* est fait par le biais de la barre d'outils illustrée par la figure 2.5.

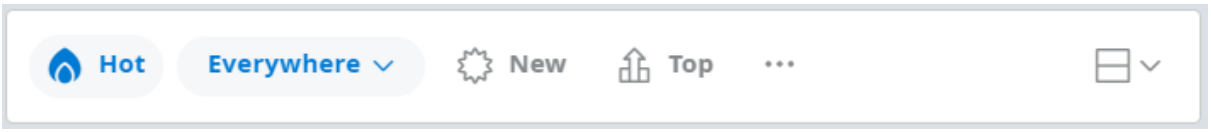

FIGURE 2.5. – Tri des postes sur *Reddit*

On peut y constater que les postes sont triés par le paramètre *Hot*, qui signifie que le critère de pertinence est le nombre de likes obtenus dans un temps récent. L'utilisateur peut également les trier par le paramètre *Top* - nombre total de likes - ou *New* - temps d'émission le plus récent. Lorsqu'on choisit l'une de ces trois méthodes de triage, il est possible d'affiner ce tri par des paramètres supplémentaires, comme la zone d'émission géographique du poste, la date d'émission, ect.

Lorsqu'un utilisateur crée un poste, il doit obligatoirement lui assigner un thème. Le poste peut être émis sous la forme d'un texte, d'un lien url, d'une image ou d'une vidéo. La figure 2.6 nous montre un aperçu visuel d'un poste *Reddit* sous forme textuelle.

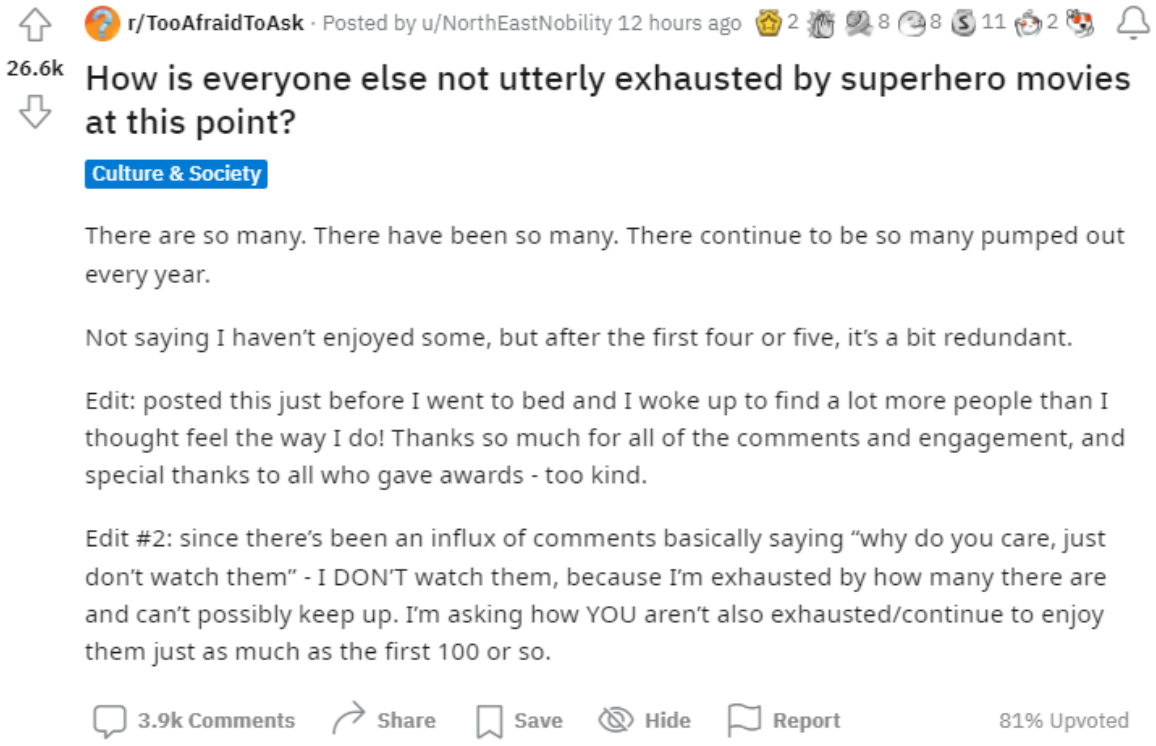

FIGURE 2.6. – Poste *Reddit*

On peut voir que, pour un poste existant, un utilisateur a la possibilité de le commenter, de le liker, de le partager ou de le sauvegarder.

Profil d'utilisateur

Lorsqu'on consulte un profil d'utilisateur *Reddit* (voir figure 2.7), on s'aperçoit qu'on peut obtenir des informations sur la date de création du profil et connaître ses contributions personnelles (publications, commentaires, thèmes modérés, ...). Il est également possible de s'abonner à un utilisateur grâce au bouton "Follow".

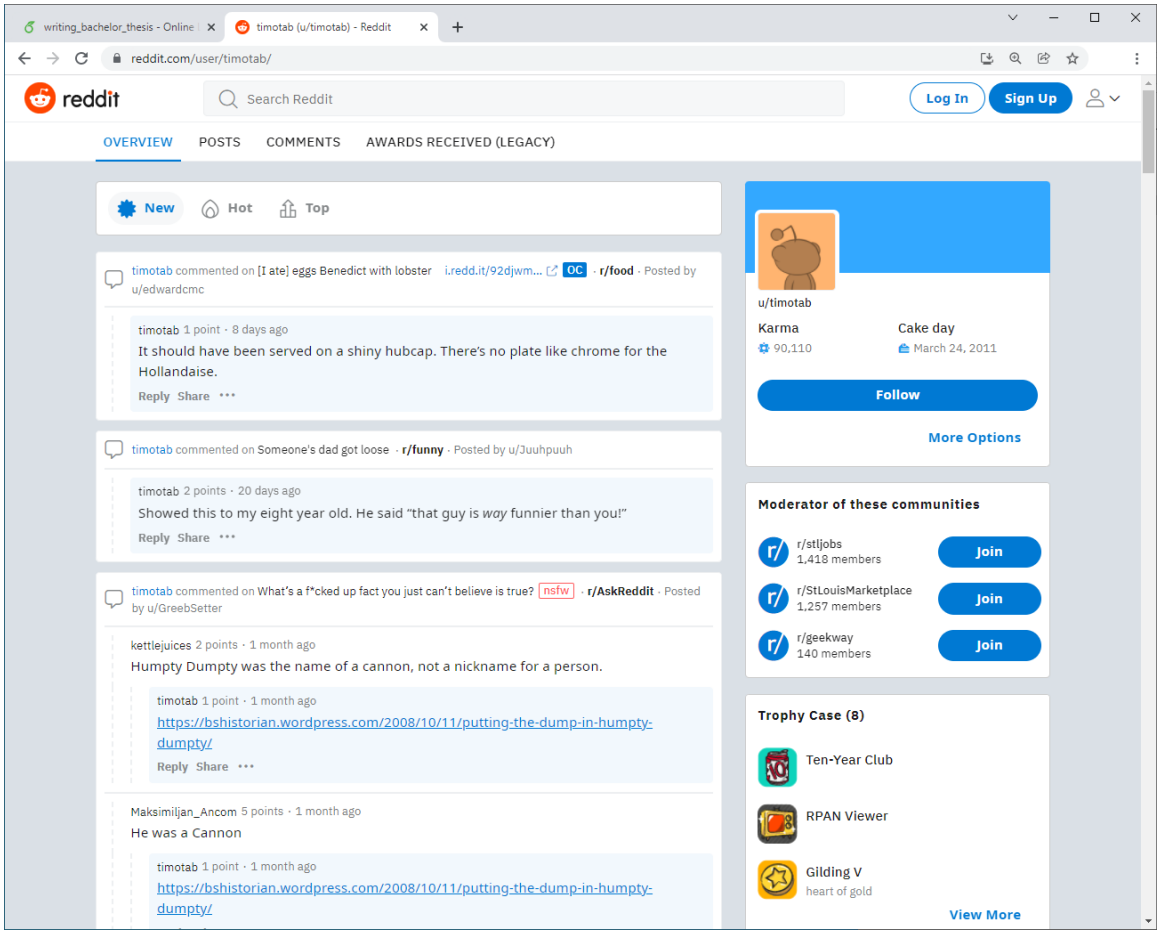

FIGURE 2.7. – Profil d'utilisateur *Reddit*

Tableau récapitulatif des fonctionnalités

Le tableau 2.1 récapitule les fonctionnalités pertinentes de *Reddit* qui pourront être utilisés et adaptés aux besoins du projet.

Role	Fonctionnalité
Utilisateur	Créer un compte
	Se connecter
	Rechercher un thème
	S'abonner à un thème
	Se désabonner d'un thème
	Créer un thème
	Créer un poste
	Liker un poste
	Disliker un poste
	Commenter un poste
	Commenter un commentaire
	Consulter un profil
	Suivre un profil
	Signaler un poste
	Signaler un thème
Créateur d'un thème	Censurer un poste
	Censurer un commentaire
	Bannir un utilisateur d'un thème
	Rajouter un modérateur
Administrateur	Censurer un thème
	Bannir utilisateur de *Reddit*

TABLE 2.1. – Tableau récapitulatif des fonctionnalités de *Reddit*

Lors de la création d'un poste, l'utilisateur a l'obligation de mentionner le thème auquel celui-ci sera lié. Pour notre FAQ, nous allons déterminer dans la section 2.4 si une question peut faire partie de plusieurs thèmes à la fois.

2.1.2. Quora

Quora[8] est un «réseau de questions-réponses»[9] lancé en 2010. Le but initial de *Quora* est de permettre aux utilisateurs de poser des questions - plutôt d'ordre de culture générale ou scientifique - auxquelles d'autres utilisateurs, plus ou moins spécialistes, peuvent répondre par une explication plus ou moins détaillée.

Page d'accueil

Quora nécessite une inscription pour accéder à sa page d'accueil, que l'on peut faire en utilisant directement son adresse e-mail ou un de ses comptes de réseau social existant comme Facebook ou Google. Durant le processus d'inscription, *Quora* demande de sélectionner ses centres d'intérêts - littérature, mathématiques, ... - et les langues maitrisées

afin que les questions liées à ces choix s'affichent dans le fil d'actualité une fois qu'on accède au forum.

L'utilisateur tombe, une fois le processus d'inscription terminé, sur la page d'accueil illustrée par la figure 2.8.

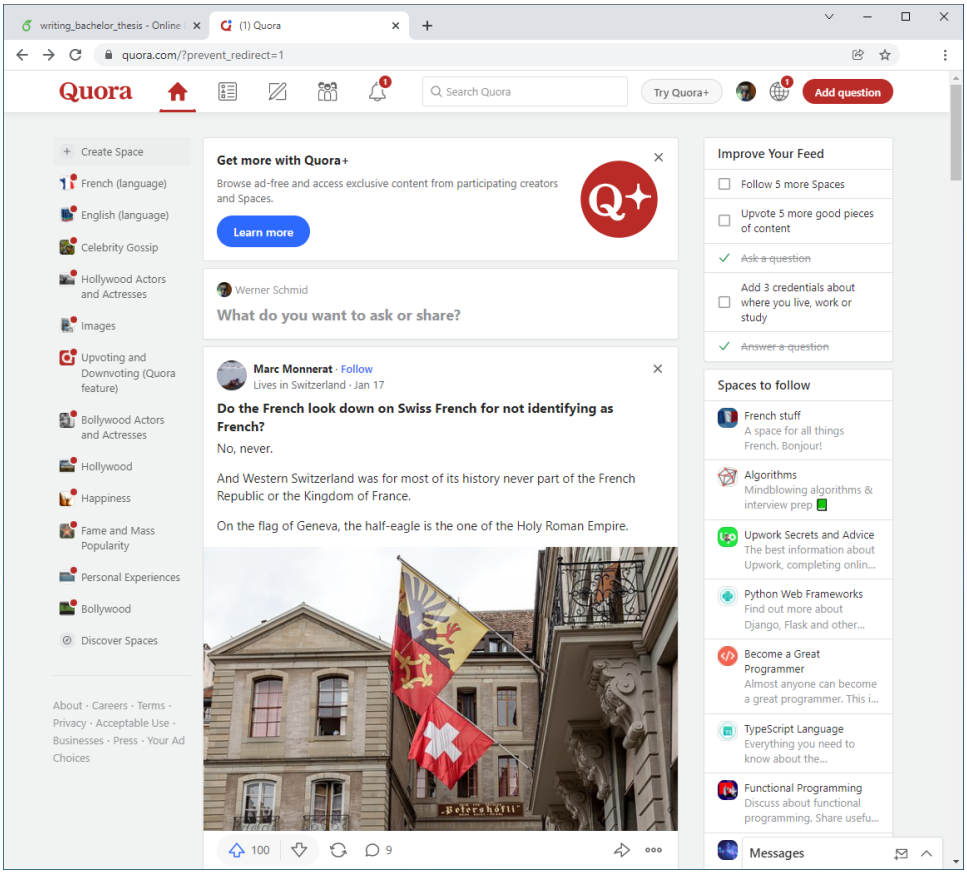

FIGURE 2.8. – Page d'accueil de *Quora*

Comme dans le cas de *Reddit*, *Quora* se compose également d'une barre de recherche permettant de rechercher des questions, des thèmes ou des utilisateurs de la plateforme. La page d'accueil permet également de pouvoir y poser une question, de créer un thème (espace) ou de s'abonner aux thèmes existants. L'utilisateur reçoit une notification par e-mail lorsque des questions pertinentes apparaissent sur le forum.

Fonctionnalités de l'utilisateur

Lorsqu'un utilisateur souhaite poser une question, une fenêtre (figure 2.9) s'ouvre dans le navigateur. L'utilisateur peut décider si la question aura un statut public - visible dans le fil d'actualité de l'utilisateur - ou privé - non visible dans le fil d'actualité de l'utilisateur. La question doit être formulée en 1 phrase et ne comporte pas de champ de contenu pour expliquer de manière plus détaillée la problématique.

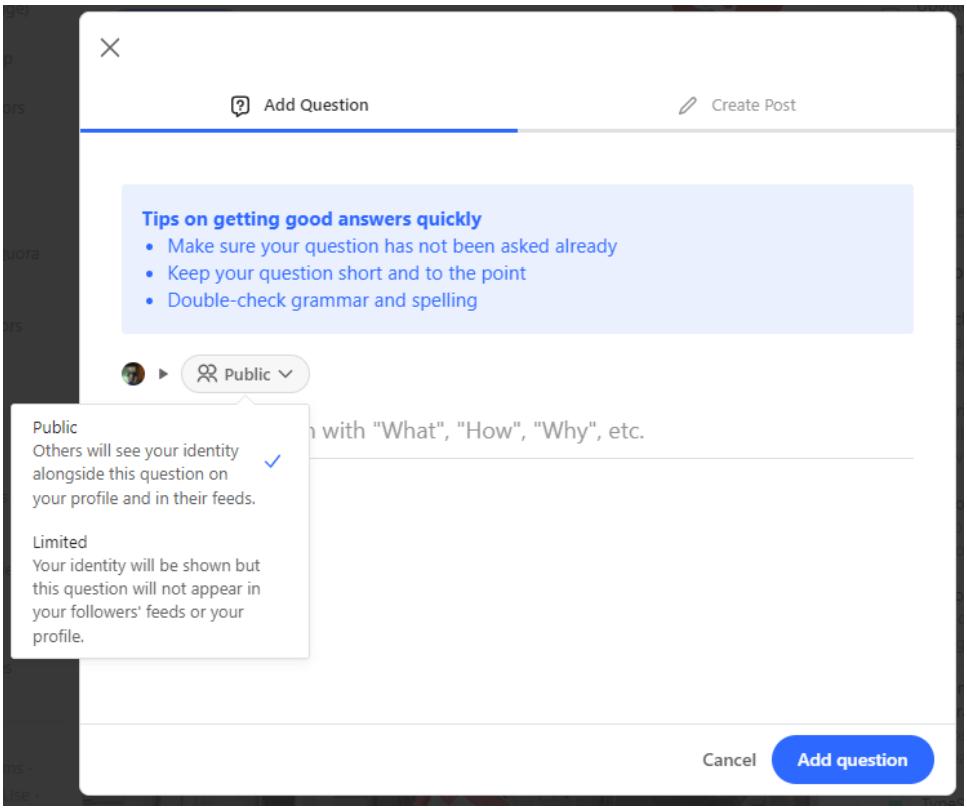

FIGURE 2.9. – Formulaire pour poser une question de *Quora*

Lorsqu'on est en train de rédiger le titre d'une question, une liste suggérant des questions comportant des titres similaires apparait en dessous du champ, comme illustré par la figure 2.10 : il est ensuite possible de cliquer sur une de ces questions et de suivre un des liens suggérés. *Quora* n'autorise pas un utilisateur à publier une question ayant exactement le même titre qu'une autre : lorsqu'on essaye de le faire, la plateforme nous retourne un message comme quoi une question similaire existe déjà et il suffit juste ensuite de cliquer dessus pour y être redirigé. Ce processus évite à l'utilisateur de poser une question qui aura déjà été posée et répondue par la communauté *Quora*.

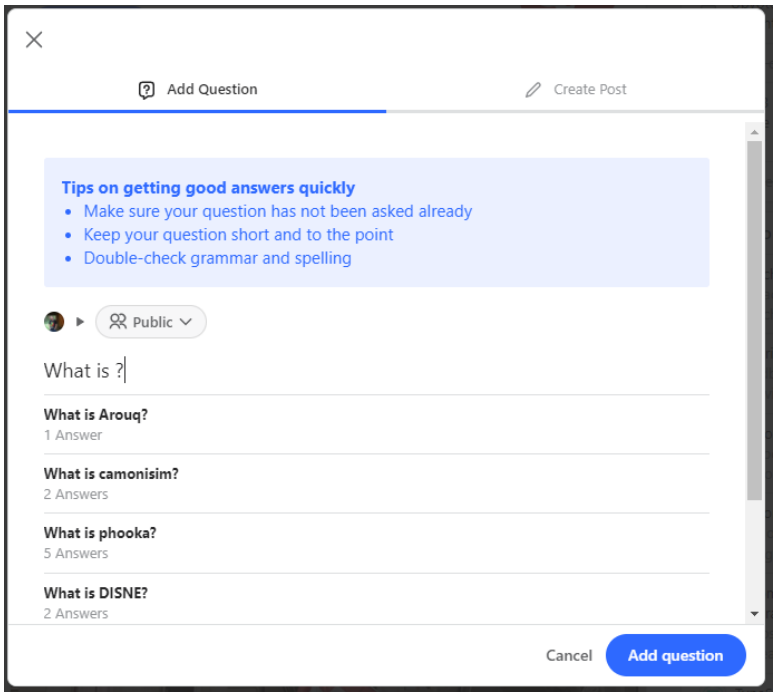

FIGURE 2.10. – Liste de suggestions pour une question sur *Quora*

Si la question est nouvelle, *Quora* demande à l'utilisateur de lui associer des thèmes, comme illustré par la figure 2.11. Une fois que l'on a cliqué sur "Terminé", la question devient disponible sur la plateforme et sur le fil d'actualité.

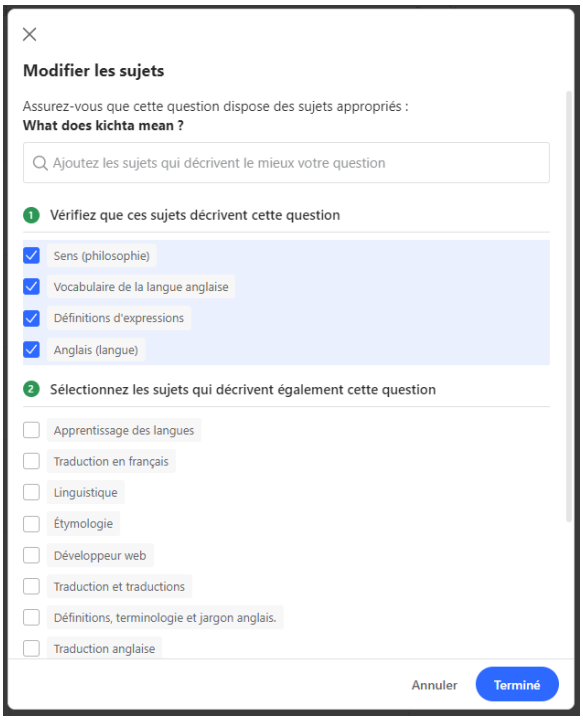

FIGURE 2.11. – Choix des thèmes pour une question *Quora*

La figure 2.12 nous montre qu'il est possible, pour une question qu'on a publiée, de modifier les thèmes associés, d'y répondre, de la liker, de la supprimer ou de la partager.

On peut également la traduire dans une autre langue, l'éditer, la commenter ou liker les réponses émises par les utilisateurs.

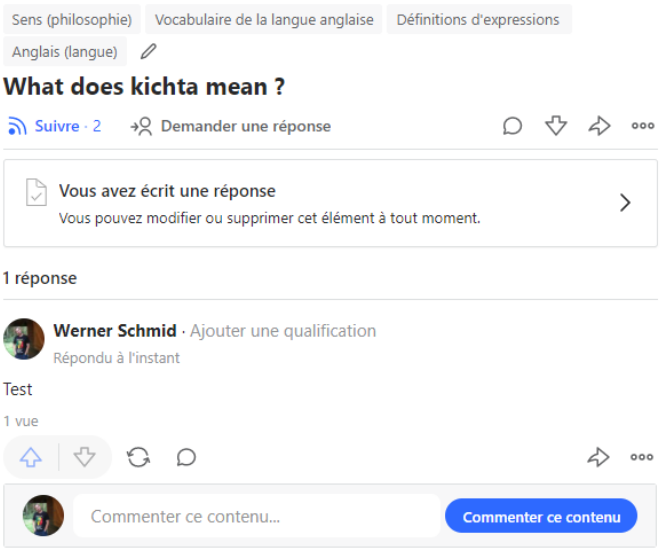

FIGURE 2.12. – Page de présentation d'une question *Quora*

Profil d'utilisateur

La figure 2.13 illustre à quoi ressemble un profil d'utilisateur *Quora*. Sa page contient des informations personnelles (poste, lieu de résidence, popularité et compétences) ainsi que ses contributions sur le forum (questions posées, réponses émises, followers et followings). Il est possible de suivre l'utilisateur, de lui envoyer un message ou de le signaler.

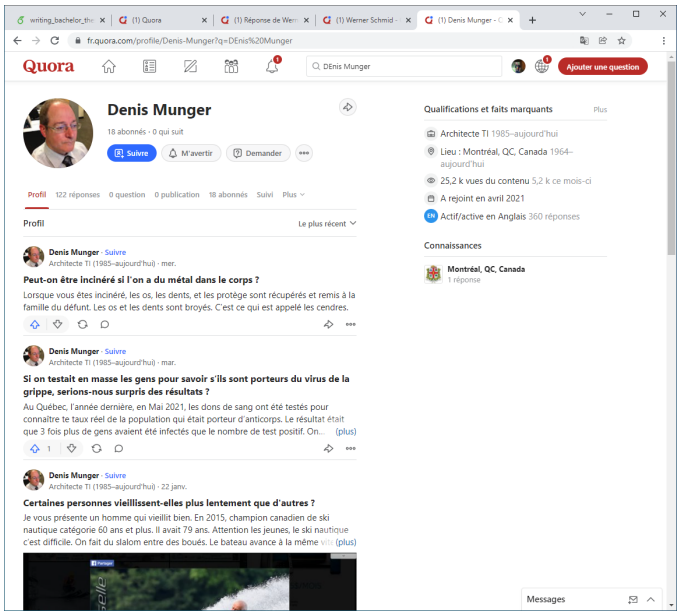

FIGURE 2.13. – Profil d'utilisateur sur *Quora*

Tableau récapitulatif des fonctionnalités

Le tableau 2.2 récapitule les fonctionnalités pertinentes de *Quora* qui pourront être utilisées et adaptées aux besoins du projet.

Role	Fonctionnalité
Utilisateur	Créer un compte
	Créer un compte avec ses réseaux sociaux
	Se connecter
	Créer un thème
	Rechercher un thème
	S'abonner à un thème
	Se désabonner d'un thème
	Rechercher un utilisateur
	Poser une question publique
	Poser une question privée
	Répondre à une question
	Rechercher une question
	Commenter une réponse
	Commenter un commentaire
	Liker une réponse
	Liker un commentaire
	Modifier une question
	Modifier une réponse
	Modifier un commentaire
	Supprimer une question
	Supprimer une réponse
	Supprimer un commentaire
	Suivre un utilisateur
	Envoyer un message à un utilisateur
	Signaler un utilisateur

TABLE 2.2. – Tableau récapitulatif des fonctionnalités de *Quora*

2.1.3. StackOverFlow

StackOverFlow[13] est un forum de questions/réponses relatifs à des problématiques de développement logiciel ou de programmation. La plateforme est extrêmement réputée dans le monde informatique et permet souvent à des programmeurs de trouver des postes répondant déjà aux problèmes qu'ils rencontrent.

Page d'accueil

La figure 2.14 nous montre un aperçu de l'interface lorsqu'on tombe sur la page d'accueil de *StackOverFlow*. La page web est composée d'une barre de recherche en en-tête, d'un fil d'actualité comportant les questions les plus en vogue aujourd'hui et d'un bouton pour poser une nouvelle question. Il est possible de défiler le fil des questions vers le bas. Comme dans les fils d'actualités des forums précédents, il est possible de modifier la pertinence

d'affichage des questions en les triant par leur date d'ajout, leur nombre de réponses ou par l'utilisateur qui aura posé la question.

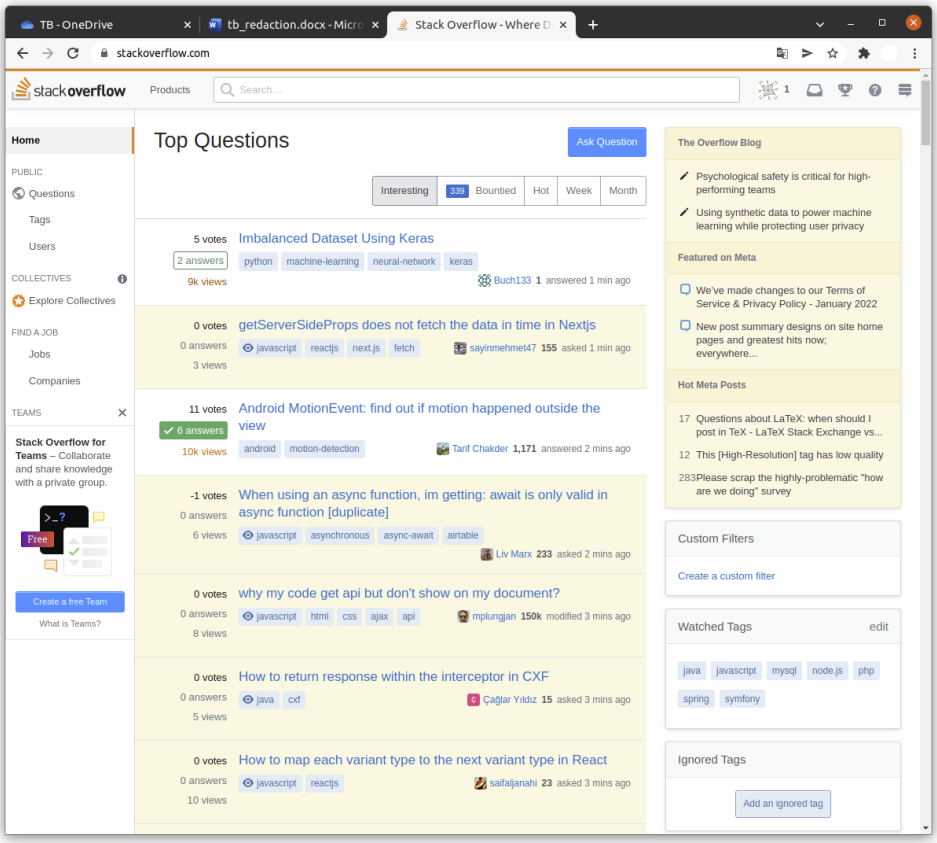

FIGURE 2.14. – Page d'accueil de *StackOverFlow*

Lorsqu'on sélectionne la barre d'outils, comme illustrée par la figure 2.15, un menu devient visible et explique à l'utilisateur comment procéder dans ses recherches pour trouver un utilisateur, un tag ou des questions triés par des paramètres pertinents comme le nombre de réponses.

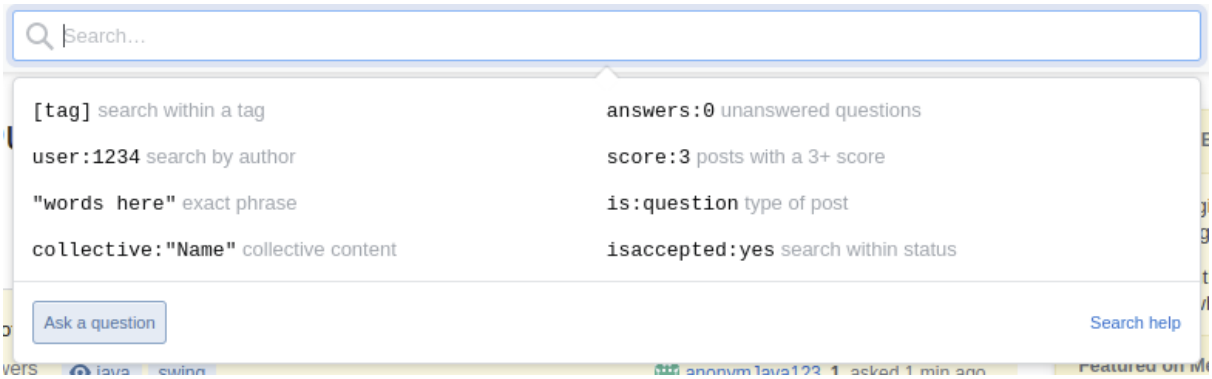

FIGURE 2.15. – Page d'accueil de *StackOverFlow*

La page d'accueil d'un tag, illustré par la figure 2.16, a comme en-tête le nom du tag et une description de celui-ci. Le reste de la page ne diffère pas de la page d'accueil de base, avec un fil d'actualité des questions reliés au tag recherché.

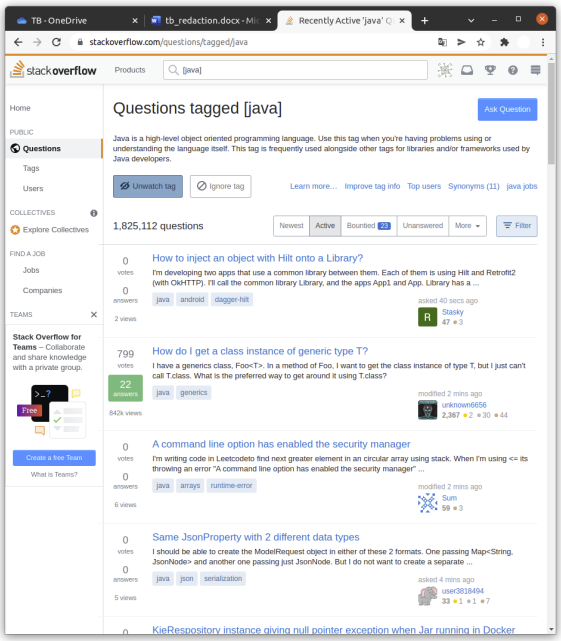

FIGURE 2.16. – Page du tag Java sur *StackOverFlow*

Fonctionnalités de l'utilisateur

Lorsqu'on souhaite poser une question sur *StackOverFlow*, une nouvelle page, illustrée par la figure 2.17, s'affiche dans le navigateur. L'utilisateur doit préciser le titre et le contenu de la question et lui associé jusqu'à 5 tags existants.

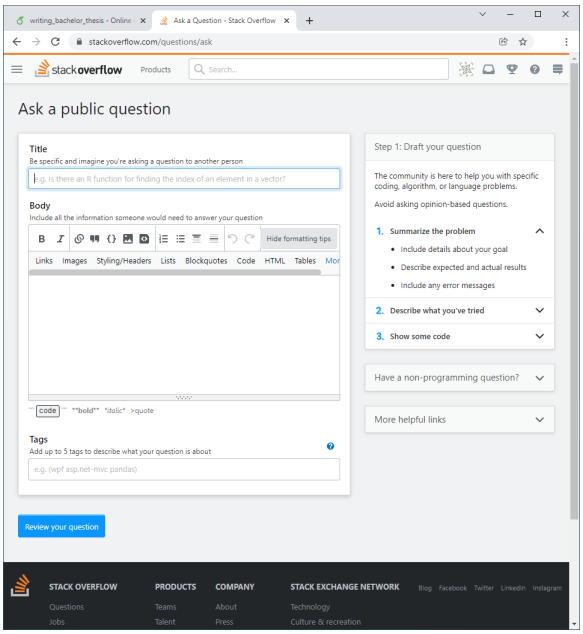

FIGURE 2.17. – Formulaire pour poser une question sur *StackOverFlow*

Une fois une question publiée sur le forum, il est ensuite possible de la liker ou de publier une réponse à celle-ci. Il est également possible de commenter une question ou une réponse : un commentaire ne correspond pas à la résolution de la problématique, mais, par exemple, à des demandes d'éclaircissement. Lorsqu'une réponse est publiée, il est également possible de la liker.

Profil d'utilisateur

Lorsqu'on sélectionne un profil sur *StackOverFlow*, une page similaire à la figure 2.18 apparait dans le navigateur. La page contient comme en-tête le nom de l'utilisateur, une photo de profil, des informations sur le moment où l'utilisateur s'est inscrit et la dernière fois qu'il s'est connecté à la plateforme. Elle contient également des informations sur les contributions de l'utilisateur à la plateforme par le biais des onglets "Profile" et "Activity". L'onglet "Developper Story" retrace le parcours professionnel de l'utilisateur en tant que développeur informatique.

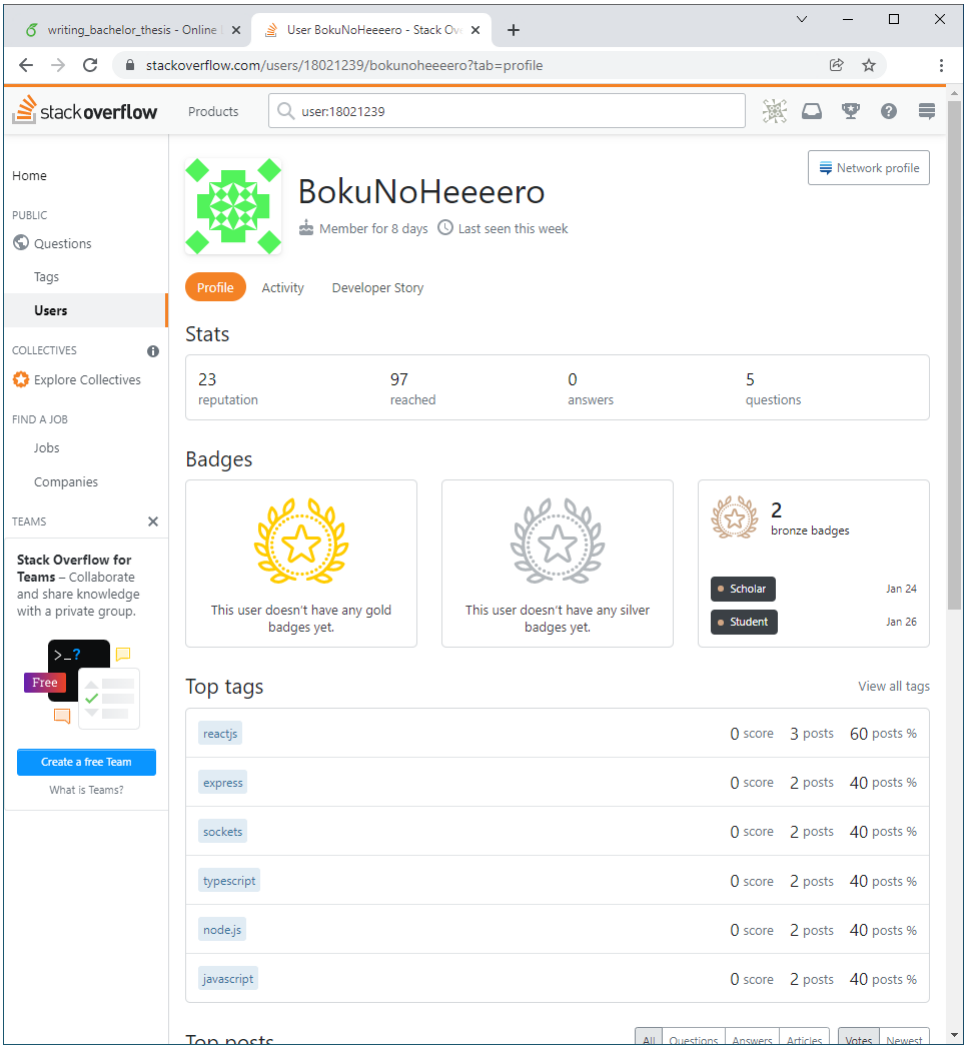

FIGURE 2.18. – Page de profil sur *StackOverFlow*

Tableau récapitulatif des fonctionnalités

Le tableau 2.3 récapitule les fonctionnalités pertinentes de *StackOverFlow* qui pourront être utilisés et adaptés aux besoins du projet.

Role	Fonctionnalité
Utilisateur	Créer un compte
	Créer un compte avec ses réseaux sociaux
	Se connecter
	Créer un thème (tag)
	Rechercher un thème
	Rechercher un utilisateur
	Poser une question
	Répondre à une question
	Rechercher une question
	Commenter une question
	Commenter une réponse
	Commenter un commentaire
	Liker une question
	Liker une réponse
	Modifier une question
	Modifier une réponse
	Modifier un commentaire
	Supprimer une question
	Supprimer une réponse
	Supprimer un commentaire

TABLE 2.3. – Tableau récapitulatif des fonctionnalités de *StackOverFlow*

2.1.4. En résumé

On a pu s'apercevoir durant l'exploration des trois plateformes précédentes que les fonctionnalités pertinentes de chacune d'entre elles ne diffèrent pas énormément dans leur globalité et que les divergences sont surtout dues aux buts spécifiques voulus par celles-ci. Les prochains chapitres vont reprendre le travail qui aura été accompli dans cette partie explorative pour définir les besoins des utilisateurs, tout en adaptant ceux-ci aux exigences de l'association.

2.2. Définition des rôles sur la plateforme

Suite à des discussions avec les membres de l'association et grâce au travail entrepris dans la section 2.1, nous avons réussi à identifier trois rôles d'utilisateurs différents sur la FAQ. Ces rôles sont l'utilisateur, le modérateur et l'administrateur.

Un utilisateur est un simple participant au forum : il peut y poser des questions, donner des réponses ou liker des postes. Il peut également signaler des questions ou des réponses inappropriés violant les règles de participation.

Un modérateur a un rôle de superviseur d'un cercle de discussion. Par le biais d'un tableau de bord spécifique, un superviseur pourra voir les questions et les postes signalés par les utilisateurs. Il pourra ensuite procéder à l'effacement de la question ou annuler le signalement en le classant comme signalement abusif. Ce tableau de bord contient également des informations sur les utilisateurs ayant subi le plus de signalements ou ayant le plus abusé de cette fonctionnalité. Si le nombre de cas avérés atteint 5 infractions, le modérateur peut suspendre l'utilisateur d'un cercle de discussion comme premier avertissement. En cas de récidive après une première suspension, le modérateur pourra signaler l'utilisateur aux administrateurs pour demander son bannissement.

Un administrateur est le compte possédant le plus d'accès sur la plateforme. Lors de la création de la base de données de la FAQ, un compte administrateur de base avec des accès par défaut est créé d'office. Ceci permet de disposer directement d'un compte avec ces droits qui est utilisable par le comité de l'association. Le tableau de bord de l'administrateur permet de créer des cercles de discussion et de modifier les droits d'accès des utilisateurs. L'administrateur peut également consulter les signalements de comptes émis par les modérateurs et procéder au bannissement des utilisateurs posant problème.

2.3. Définition des cercles de discussion

Un cercle de discussion correspond à la catégorie de sujet qu'une question ou un projet aura sur le forum de Solid'Ark. Il est équivalent aux thèmes des forums explorés et l'utilisateur devra obligatoirement assigner un cercle d'appartenance lors de la création d'une question. Ceci a pour but de rassembler les personnes compétentes dans des projets et des discussions où ceux-ci y trouveront leur intérêt et pourront y faire valoir leurs connaissances et leur savoir-faire dans un domaine spécifique.

Six cercles de discussion différents ont déjà été définis par les membres de l'association, auprès desquelles d'autres pourront potentiellement se greffer. Ces cercles sont :

- le cercle médiatique, comportant les problématiques liées à la communication et à la promotion de Solid'Ark.
- le cercle technique, comportant les problématiques d'implémentation technique de la plateforme.
- le cercle administratif, comportant les questions en lien avec la gestion administrative de Solid'Ark.
- le cercle exécutif, qui concerne tout ce qui est en lien avec le management de Solid'Ark.
- le cercle législatif, comportant les problématiques d'ordre juridique de Solid'Ark.
- le cercle financier, qui concerne le financement et les dépenses de Solid'Ark.

2.4. Cas d'utilisation

Un cas d'utilisation est une action qu'un utilisateur peut effectuer dans un système informatique, comme créer un poste, commenter une publication ou se connecter. Il permet d'identifier, de clarifier et d'organiser les besoins interactifs de l'utilisateur.

Un diagramme de cas d'utilisation représente graphiquement l'ensemble des interactions fonctionnelles que peuvent faire les types d'utilisateurs existants dans un système informatique. Ceux-ci sont classés en différents rôles qui leur donneront accès à certains groupes de fonctionnalités.

Les diagrammes utilisés dans ce travail respectent la notation UML[16]. Voici une explication de chaque objet utilisé :

— une bulle (figure 2.19) représente un cas d'utilisation, c'est-à-dire une fonctionnalité disponible dans le système.

— un système (figure 2.20) représente un regroupement de cas d'utilisation ayant une interaction commune.

— un acteur (figure 2.21) représente un rôle d'utilisateur interagissant avec le système. Cet acteur peut être une personne physique ou un autre système informatique.

— un lien de communication (figure 2.22) désigne l'exécution d'un cas d'utilisation par un acteur. Tous les cas d'utilisation doivent obligatoirement être reliés à un acteur.

FIGURE 2.19. – Représentation d'un cas d'utilisation

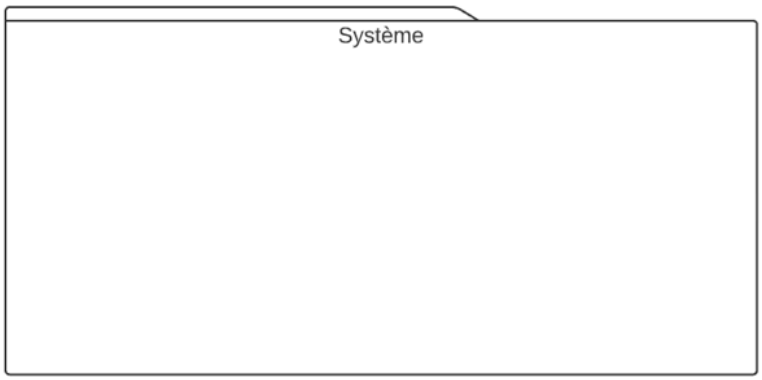

FIGURE 2.20. – Représentation d'un système

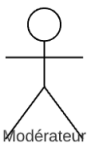

FIGURE 2.21. – Représentation d'un acteur

FIGURE 2.22. – Représentation d'un lien de communication

Les cas d'utilisation sont généralement accompagnés d'une description textuelle, expliquant de manière plus détaillée leur procédure et leurs étapes de validation.

L'utilisation de diagramme de cas d'utilisation facilite la communication et la représentation du système avec des personnes ne faisant pas partie du monde de l'informatique et de la programmation. La description textuelle permet de détailler le fonctionnement d'un cas d'utilisation.

Les sous-sections suivantes vont détailler l'ensemble des cas d'utilisation existants de la FAQ.

2.4.1. Inscription et connexion

La figure 2.23 représentent les cas d'utilisation de base de tout utilisateur. Pour avoir accès au forum et y interagir, il est nécessaire d'être inscrit à la plateforme et de se connecter avec ses accès. La plateforme permet donc à l'utilisateur de s'inscrire uniquement avec son adresse e-mail ou en utilisant l'un de ses comptes de réseaux sociaux, qui peut être *Gmail* ou *LinkedIn*.

Une fois le processus d'inscription terminé, l'utilisateur pourra ensuite se connecter à la plateforme. Il existe également un cas d'utilisation de résiliation de mot de passe si l'utilisateur a oublié ses accès : la plateforme Solid'Ark enverra un lien de résiliation à son adresse e-mail, que celui-ci devra cliquer pour donner un nouveau mot de passe à son compte.

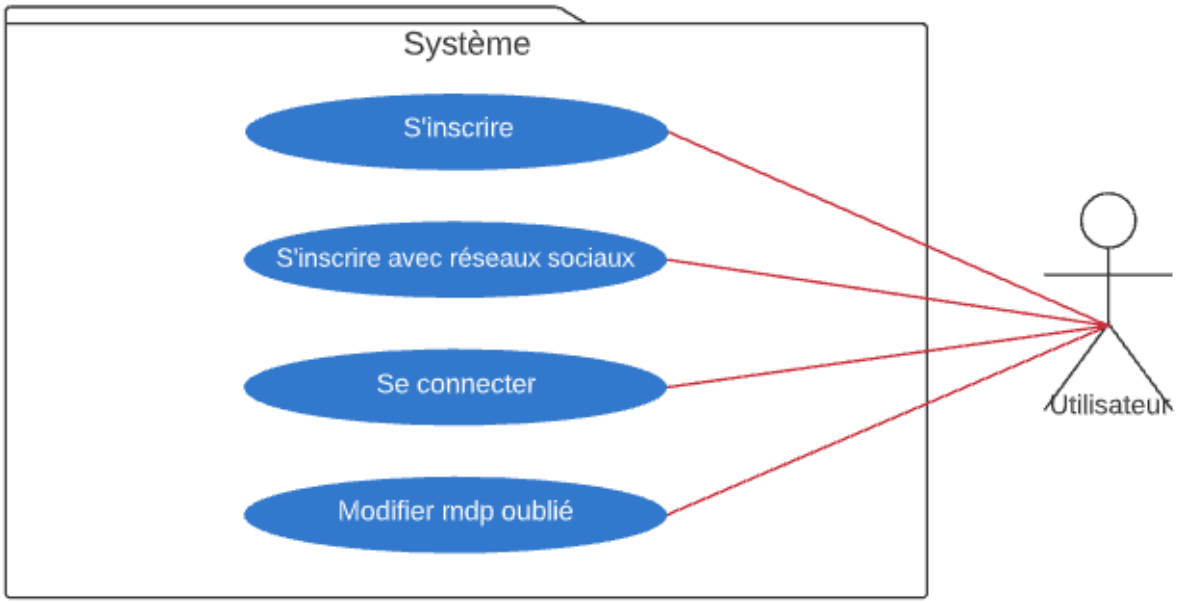

FIGURE 2.23. – Cas d'utilisation pour l'inscription et la connexion à la plateforme

Une fois que l'utilisateur est connecté, il aura accès aux autres fonctionnalités du forum.

2.4.2. Cas d'utilisation des utilisateurs

Gestion du compte personnel

L'utilisateur doit pouvoir gérer son compte personnel sur la plateforme une fois qu'il sera inscrit : il pourra mettre à jour sa photo de profil et son CV, ce qui n'est pas demandé lors du processus d'inscription afin de l'alléger. L'ensemble des cas est illustré par la figure 2.24.

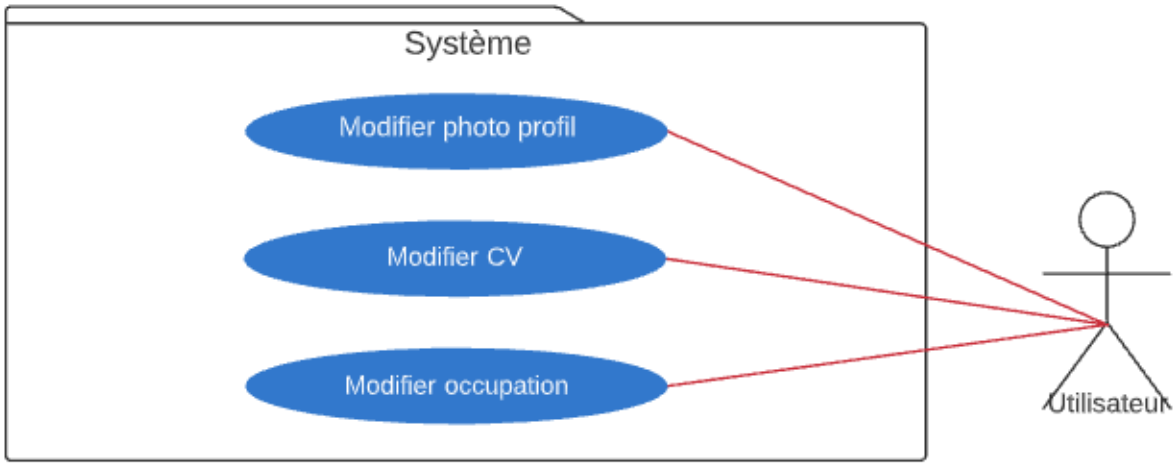

FIGURE 2.24. – Cas d'utilisation pour la gestion du compte personnel

Interaction sur le forum

La FAQ doit permettre à tout utilisateur connecté de pouvoir y interagir : il pourra y poser des questions, donner des réponses à des questions existantes, liker les questions pertinentes ou commenter brièvement une question ou une réponse donnée. La figure 2.25 énumère de manière détaillée l'ensemble des fonctionnalités liées aux interactions des utilisateurs sur le forum.

FIGURE 2.25. – Cas d'utilisation pour les interactions sur le forum

Les postes ne peuvent pas être modifiés une fois publiés sur la FAQ : les commentaires ont pour rôle de préciser ou de corriger des points qui ne sont pas clairs dans une réponse ou une question déjà émise.

Signalement de contributions

Une FAQ peut également contenir des contributions violant les règles d'utilisation du forum. Comme il est parfois difficile pour les modérateurs de recenser l'ensemble des postes ne respectant pas l'usage voulu, La plateforme doit également permettre aux utilisateurs de signaler les contributions problématiques. Le processus de signalement se déroule en deux étapes : l'utilisateur clique tout d'abord sur le bouton de signalement de la contribution en question avant de préciser le type de violation rencontré (spam, message de haine, ...) et de donner une brève description personnelle du signalement. La figure 2.26 énumère l'ensemble des fonctionnalités de signalement que peut faire l'utilisateur.

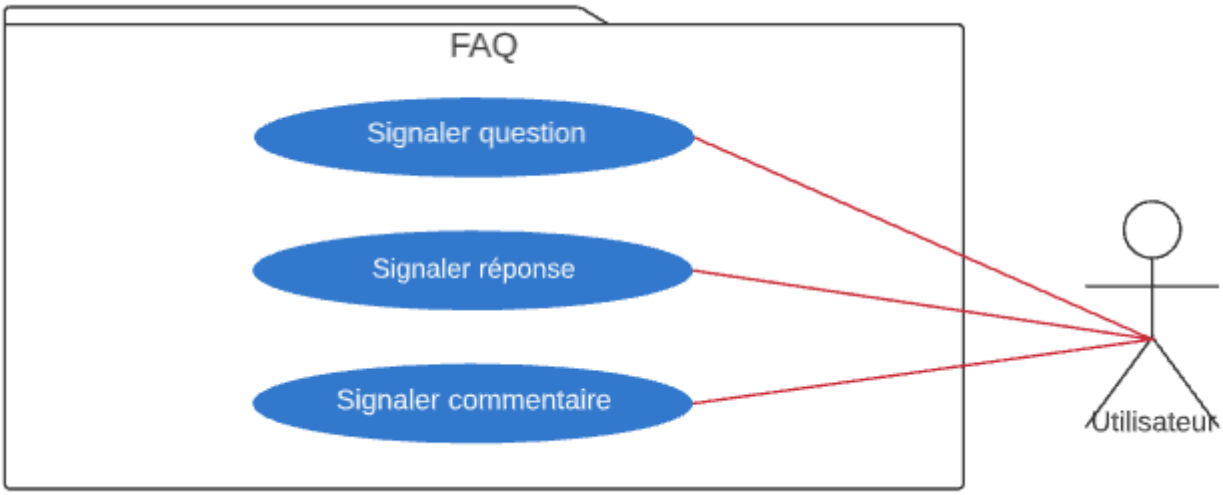

FIGURE 2.26. – Cas d'utilisation pour les signalements sur le forum

2.4.3. Cas d'utilisation des modérateurs

La FAQ est divisée en cercles de discussion, qui correspondent à des catégories mutuellement exclusives dans lesquelles on va classer une question lors de sa création. L'administrateur va assigner des modérateurs à chaque cercle de discussion, qui auront la tâche de veiller à leur bon fonctionnement.

Gestion d'un cercle de discussion

Un modérateur dispose d'un tableau de bord pour chaque cercle de discussion géré. Celui-ci lui permet d'avoir un aperçu des postes signalés et des utilisateurs problématiques. Il peut supprimer une contribution signalée s'il estime qu'elle est réellement problématique ou reporter un signalement abusif si ça n'est pas le cas.

Pour chaque suppression ou signalement abusif reporté, le compteur du nombre de fautes de l'utilisateur concerné est incrémenté et est visible sur le tableau de bord du modérateur. Lorsque le nombre de fautes atteint 5 cas, le modérateur peut suspendre l'utilisateur du cercle de discussion, ce qui l'empêchera d'interagir dessus pendant une certaine période donnée. Après avoir été suspendu, si l'utilisateur récidive et atteint de nouveau 5 cas, le modérateur va signaler l'utilisateur à l'administrateur pour demander son bannissement général de la FAQ.

La figure 2.27 nous donne un aperçu de l'ensemble des fonctionnalités que peut faire un modérateur.

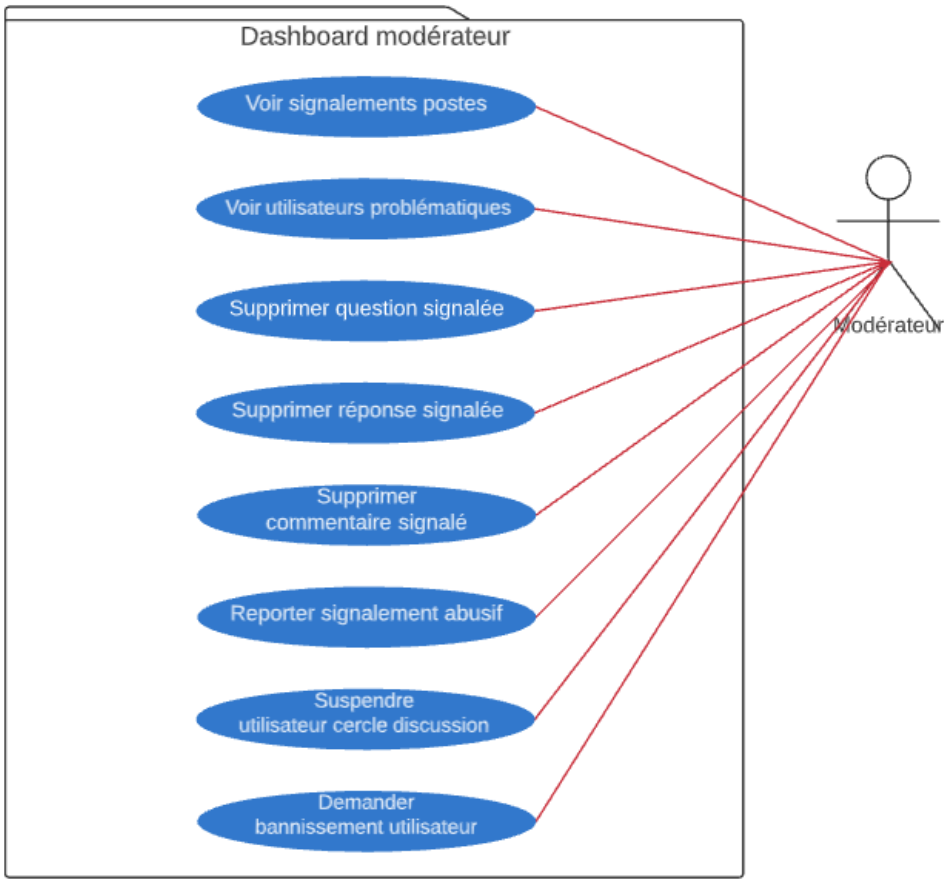

FIGURE 2.27. – Cas d'utilisation pour la gestion d'un cercle de discussion

2.4.4. Cas d'utilisation de l'administrateur

Le tableau de bord de l'administrateur permet de créer des cercles de discussion et de modifier les droits d'accès des utilisateurs. L'administrateur peut également consulter les signalements de comptes émis par les modérateurs et procéder au bannissement des utilisateurs posant problème.

Gestion des droits d'accès

Un administrateur a la possibilité de modifier le rôle d'un utilisateur existant sur la plateforme : il peut le désigner comme modérateur, utilisateur simple ou administrateur. Une fois qu'un compte aura été désigné comme administrateur, il ne sera plus possible pour les autres de modifier son rôle.

Un administrateur s'occupe également de créer les cercles de discussion existants et d'assigner des utilisateurs ayant au moins un rôle de modérateur à ceux-ci. Il peut également supprimer des modérateurs d'un cercle de discussion.

La figure 2.28 nous résume les fonctionnalités de gestion des droits d'accès d'un administrateur.

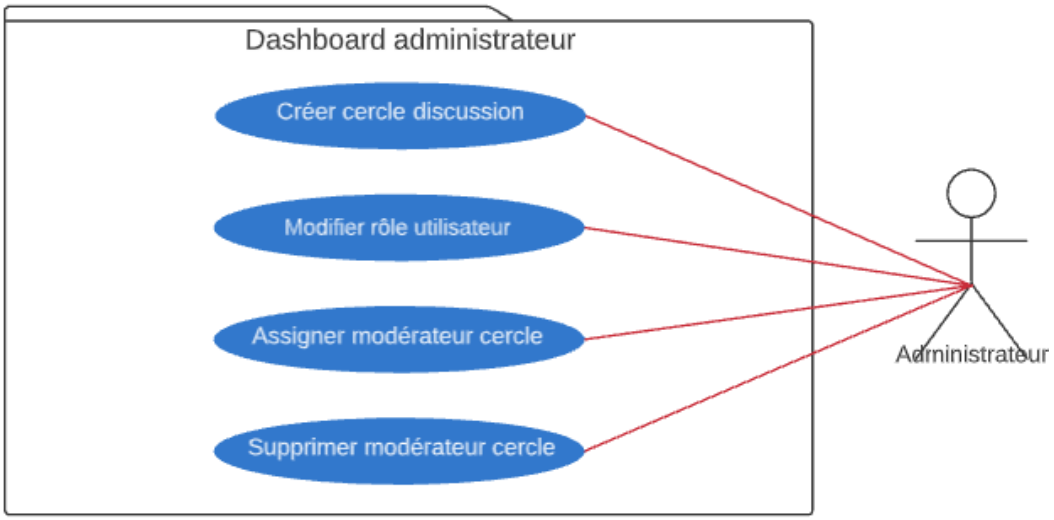

FIGURE 2.28. – Cas d'utilisation pour la gestion des droits d'accès

Bannissement des utilisateurs

Un administrateur peut consulter les signalements de compte faits par les modérateurs et procéder au bannissement de ceux-ci, comme illustré par la figure 2.29.

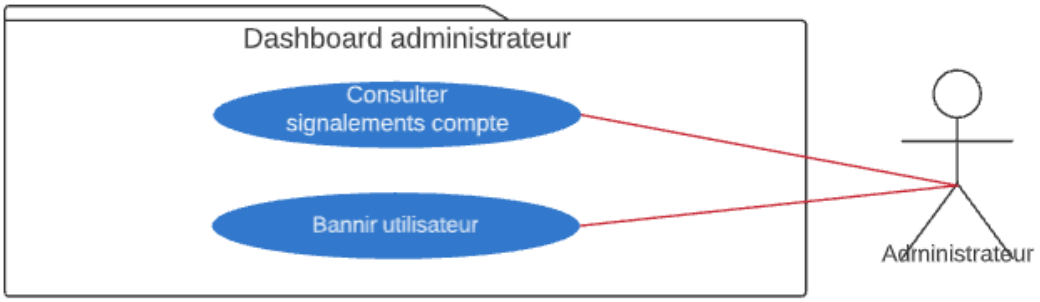

FIGURE 2.29. – Cas d'utilisation pour le bannissement des utilisateurs

2.5. Conclusion de l'étude des besoins des utilisateurs

Nous avons vu l'ensemble des cas d'utilisation des utilisateurs, des modérateurs et des administrateurs de la FAQ. L'étude des cas d'utilisation nous a permis de subdiviser le système total de la FAQ en sous-systèmes de gestion et de clarifier la manière dont on souhaiterait le structurer. On a donc pu déterminer les fonctionnalités de chaque système de gestion et expliquer et présenter plus facilement les fonctionnalités requises à des personnes ne faisant pas partie du monde de l'informatique.

Pour la partie programmation, seulement un sous-ensemble de ces fonctionnalités va être implémenté : il s'agit des cas d'utilisation "S'inscrire" et "Se connecter" dans l'ensemble "Inscription et connexion", ainsi que les cas d'utilisation du groupe "Interaction sur le forum". Le reste des fonctionnalités seront développées dans la suite du projet et dépasse le cadre de ce travail.

3
Serveur et base de données

Ce troisième chapitre va aborder la manière dont le backend a été implémenté. Nous allons expliquer des concepts théoriques utiles dans la création d'une application web et les technologies qui ont été choisies. En ce qui concerne Symfony, le framework utilisé dans ce travail, nous allons illustrer la mise en place d'une API REST et l'accès à une base de données relationnelle à l'aide de son ORM.

3.1. L'architecture REST

Dans l'idéal, une application web doit réussir à rendre le client complètement indépendant du serveur, afin pouvoir les implémenter comme nous le souhaitons. Il devient alors possible de créer des clients différents, comme des sites web ou des applications mobiles, qui communiquent avec un même serveur. Les appels au serveur se font grâce à un protocole de communication qui traite les requêtes des clients. Le serveur leur envoie une réponse et se mettra éventuellement à jour.

Un service REST est une API avec un protocole de communication séparant le client du serveur. Une application est RESTful si elle respecte les contraintes architecturales

énumérés dans cet article[12]. La première contrainte est justement la **séparation du client et du serveur**. Les cinq autres contraintes sont :

- d'avoir des communications entre le client et le serveur qui sont **sans connaissance de l'état**. Les informations concernant la session d'utilisateur sont stockées côté client et elles doivent être transmises au serveur lorsque celui-ci a besoin d'accéder à des données nécessitant une autorisation.
- la possibilité pour le client de **mettre en cache** les données du serveur lorsqu'une requête similaire se présente. Ceci permet leur réutilisation.
- la possibilité de décomposer l'API en **plusieurs couches** indépendantes qui facilitent son évolution et renforcent sa sécurité. Un client ne sait pas à quelle couche il s'adresse lorsqu'il fait une requête au serveur.
- l'utilisation d'une méthode de communication **uniformisée** pour accéder aux ressources existantes. On doit pouvoir identifier les ressources à l'aide d'un URI et les représenter. De plus, celles-ci doivent être autodescriptives. Leur accès se fait à l'aide des différentes requêtes HTTP existantes (GET, POST, PUT, PATCH et DELETE), qui seront traitées par le serveur. Celui-ci retourne ensuite une réponse contenant un en-tête et un corps.
- l'envoi de **code à la demande** au client lorsqu'il le souhaite, afin d'étendre ses fonctionnalités. Cette dernière contrainte est facultative.

L'architecture REST permet donc d'atteindre l'idéal énoncé dans le premier paragraphe de cette section. Ceci explique grandement son utilisation dans une bonne partie des projets web actuels.

3.2. Jeton d'authentification JWT

Un jeton d'authentification est une méthode d'identification ne nécessitant pas d'état. Lors de l'authentification d'un client sur un serveur, celui-ci reçoit un jeton d'accès unique et bénéficie alors d'un accès aux ressources autorisées pour une durée limitée. Lorsque le client se déconnecte ou que la durée de validité est expirée, il ne devient alors plus possible d'utiliser le jeton qui a été transmis précédemment.

Le token JWT est un type de jeton d'authentification et dont ses informations stockées peuvent être décomposées en trois parties, comme illustré par la figure 3.1. Lors de l'authentification d'un utilisateur, un token encodé par le serveur est renvoyé. Le client va ensuite lui transmettre ce token à chaque requête nécessitant des droits d'accès. Le token sera alors décodé par le serveur et parmi les informations sur l'identité du client, il va checker s'il dispose de l'autorisation d'accéder à la ressource.

Les trois parties constituant un token JWT sont :

1. un **header**. Il contient des informations sur l'algorithme utilisé pour générer la signature et le type de token.
2. un **payload**. Il stocke les informations utilisées pour donner l'accès à l'utilisateur, comme son nom ou son rôle. Les données contenues dans cette partie sont choisies par le développeur.
3. une **signature**. Elle permet de vérifier l'authenticité du token, par le biais d'un hash utilisant le header, le payload et une clé secrète.

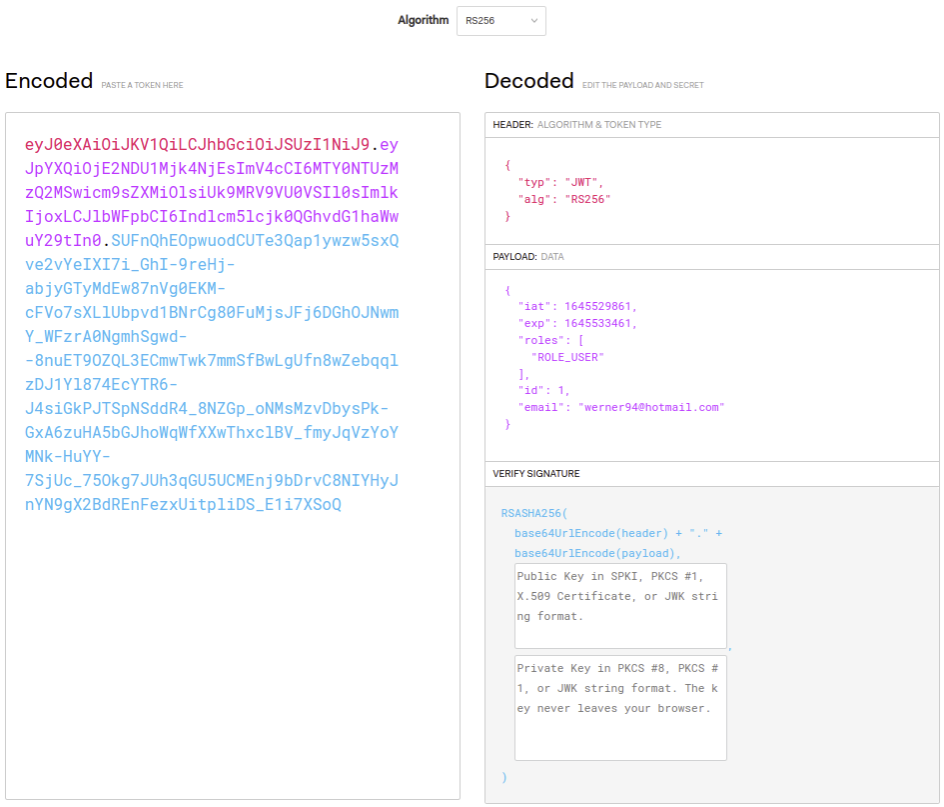

FIGURE 3.1. – Structure d'un token JWT

A chaque requête nécessitant un droit d'accès, le client envoie le token dans son en-tête sous la forme **Authorization : Bearer <token>**. Le serveur vérifie ensuite sa validité et transmet les informations voulues si c'est le cas.

3.3. ORM

En POO, un Object Relational Mapping (ORM) est un ensemble de classes permettant de manipuler des tables dans une base de données relationnelle. Les classes correspondent à des tables et leurs attributs aux attributs des tables ou à des relations entre tables. L'ORM donne au logiciel une couche d'abstraction supplémentaire où l'utilisateur s'adressera à des objets pour modifier les entités stockées dans le DBMS. L'avantage de cette utilisation est qu'elle évite le souci de devoir comprendre le fonctionnement du DBMS car l'ORM prend la responsabilité de transformer les modifications faites aux objets en requêtes SQL modifiant les entités.

Symfony utilise Doctrine comme ORM. Son fonctionnement sera expliqué et illustré par un exemple lorsque nous aborderons le framework.

3.4. MySQL

MySQL[6] est un DBMS open source appartenant à Oracle. Son utilisation est aujourd'hui extrêmement populaire, particulièrement pour les sites crées avec WordPress.

Symfony offre la possibilité de choisir quel DBMS utiliser et associer avec son ORM. MySQL a été choisi en raison des nombreux cours universitaires enseignés à son sujet et de sa grande popularité dans le monde professionnel.

3.5. Endpoints

3.5.1. Qu'est-ce qu'un Endpoint ?

Les endpoints, ou points de terminaison en français, correspondent à l'emplacement numérique d'une ressource API sur le web. Un client s'adresse à ceux-ci à l'aide des requêtes HTTP existantes (GET, POST, ...), traitées et retournées sous le format demandé (JSON, XML, ...).

FIGURE 3.2. – Structure d'un Endpoint

Comme illustré par la figure 3.2, Le chemin d'URI d'un endpoint est composé :

- **d'un protocole**, qui correspond à des règles de communication entre deux ordinateurs à travers le réseau. Le protocole utilisé dans le cas d'une API REST est HTTP, qui permet la communication avec un serveur web pour obtenir des documents, ou son équivalent sécurisé HTTPS.
- **d'un nom de domaine**, qui permet au serveur DNS de trouver l'adresse IP de l'URI. Un nom de domaine permet de mémoriser plus facilement la ressource et de ne pas écrire l'adresse IP (par exemple : 192.168.1.0).
- **d'un chemin d'accès**, qui correspond à l'emplacement de la ressource. Le chemin d'accès est composé du **nom de la ressource**, très souvent donné au pluriel, et éventuellement d'un **identifiant** lorsqu'on demande d'accéder à un élément précis parmi l'ensemble.

On peut également rajouter des **filtres** au chemin d'URI en les rajoutant sous la forme *?filtre=valeur* à la fin de l'URL. L'utilité de rajouter des filtres dépend de la ressource et des possibilités de tri qu'on veut lui associer, comme la pagination ou l'ordre.

Dans le cadre de notre serveur, lorsqu'un utilisateur fait une requête au serveur, une réponse sous format JSON est renvoyée, avec un code de réponse HTTP. Les différents codes utilisés sont :

- **2xx**, qui signifient que la requête a été traitée avec succès.
- **4xx**, qui indiquent que l'erreur provient du client, comme une ressource non trouvée ou qu'il n'a pas de droits d'accès.
- **500**, qui indique une erreur interne du serveur. Ce code est généré automatiquement par le serveur et nous indique qu'un problème se situe dans notre programmation, alors que les deux autres types de code ont été programmés.

3.5.2. Swagger UI

Swagger[14] est un IDL permettant de documenter et de tester le fonctionnement d'une API REST et dont le format de données utilisé est JSON. Swagger est justement utilisé par Symfony lorsque nous souhaitons y créer une API. La figure 3.3 illustre comment Swagger se présente sur le framework.

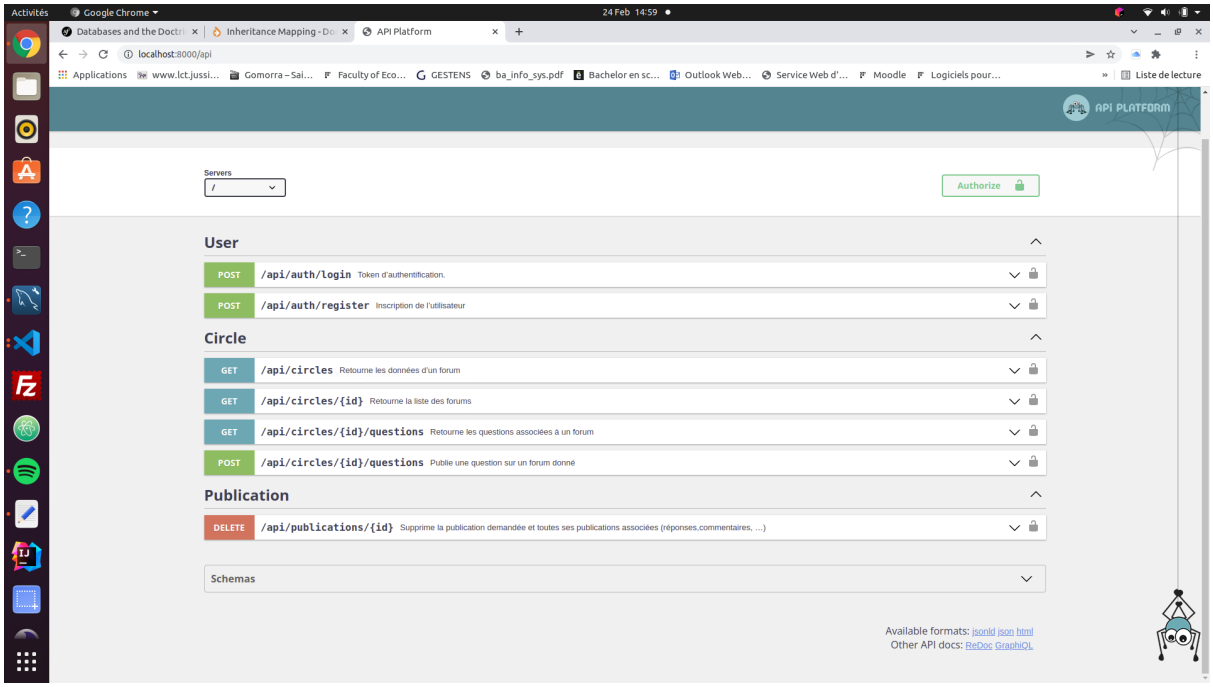

FIGURE 3.3. – Interface utilisateur de Swagger sur Symfony

Lorsqu'on déroule le menu accordéon d'un endpoint, comme montré par la figure 3.4, on peut voir que sa description contient :

- le **type de la requête** (POST, GET).
- le **chemin d'accès** de la ressource.
- un **cadenas** précisant si l'accès nécessite ou pas une authentification.
- la liste des **paramètres** à envoyer à la ressource, comme l'identifiant ou le numéro de page voulu.
- la structure du **corps de la requête** HTTP à envoyer, dans le format choisi (application/json, application/ld+json, application/xml).
- les **réponses** possibles qui peuvent être renvoyés par le serveur, avec leur code.

Le bouton "Try it out" permet de tester l'envoi d'une requête sur l'API : on peut y spécifier les valeurs des paramètres énumérés et customiser le corps de la requête. Une fois l'envoi effectué, la réponse s'affiche sur l'interface avec le code et le contenu retourné par le serveur.

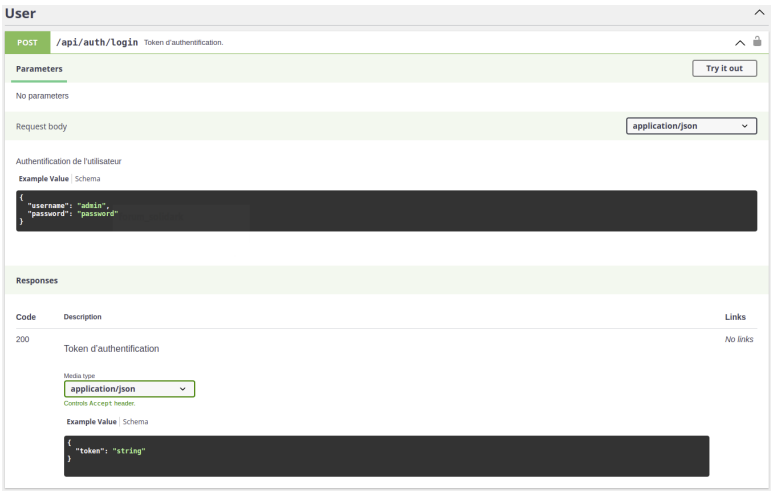

FIGURE 3.4. – Description d'un Endpoint sur Swagger

Comme illustré par la figure 3.5 représentant l'en-tête de la page d'accueil de Swagger, il est également possible de configurer les droits d'accès aux endpoints par le bouton "Authorize".

FIGURE 3.5. – En-tête de la page d'accueil de Swagger

Lorsque l'on clique sur "Authorize", une fenêtre modale comme illustrée par la figure 3.6 s'ouvre et demande de rentrer les valeurs des paramètres permettant l'accès. La figure 3.6 illustre une valeur à entrer pour accompagner le champ "Authorization" transmis dans l'en-tête de la requête à envoyer au serveur. Une fois que l'on clique sur le bouton "Authorize", chaque requête effectuée dans Swagger la transmettra automatiquement et permettra d'accéder aux ressources voulues.

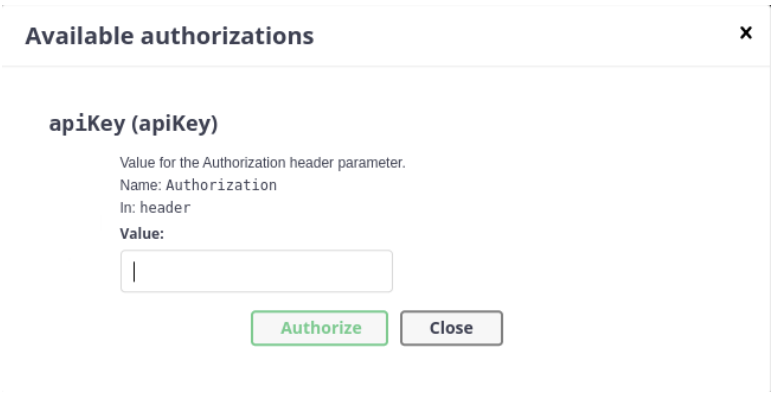

FIGURE 3.6. – Formulaire pour rentrer les clés d'accès à l'API

3.6. Symfony

Symfony[15] est un framework pour la création de projets web. Il a été développé avec PHP et contient des fonctionnalités facilitant le développement dans ce langage de programmation. Il est divisé en modules que l'on peut décider d'intégrer dans notre application, comme la création et la gestion d'une base de données, l'envoi d'e-mail, le routage ou l'authentification. Symfony permet de créer des sites classiques à l'aide de son outil de templating Twig, mais également des services REST ou GraphQL. C'est le framework utilisé sur le CMS Drupal.

Dans cette partie du rapport, Deux outils puissants de Symfony seront présentés : API Platform, qui permet de créer une API REST, et Doctrine, l'ORM gérant la création et la modification d'une base de données. L'utilisation des deux outils sera illustrée avec un exemple de création de ressource API à la fin de cette section.

3.6.1. API Platform

API Platform[1] est un composant de Symfony pour créer des API REST ou GraphQL. Il permet de générer automatiquement des ressources à l'aide de son système de configuration et d'annotations. Les annotations écrites par le développeur permettent de rajouter des endpoints personnalisés ou de rendre certaines requêtes (GET, POST, ...) inacessibles. Il génère également la documentation de l'API avec Swagger et on la peut personnaliser à l'aide d'un Decorator sur la classe responsable de la créer.

3.6.2. Doctrine

Doctrine[3] est l'ORM utilisé par Symfony pour accéder à une base de données relationnelle : elle traduit en requêtes SQL les opérations faites sur les objets représentant les entités stockées dans les tables du DBMS. Doctrine supporte tous les DBMS existants, que ce soit MySQL, utilisé dans le cadre de ce travail, PostGreSQL, Oracle ou SQLite. Il assure également une sécurité contre les failles traditionnelles atteignant les requêtes, comme les injections SQL. Etant indépendant de Symfony, il peut être installé sur n'importe quel projet PHP existant.

3.6.3. Exemple : création d'une ressource API associée à une table stockée dans la base de données

On part du présupposé que l'on a installé Composer sur notre machine, processus qui ne sera pas expliqué dans le cadre de ce travail et qui est décrit sur ce lien. Une fois Composer installé, ouvrez l'invite de commande dans le dossier de votre choix et tapez l'instruction suivante, en remplaçant *my_project_directory* par le nom que vous souhaitez donner au projet :

```
1 composer create-project symfony/skeleton my_project_directory
```

Code Source 3.1 – Création d'un projet Symfony

Ouvrez ensuite le nouveau dossier *my_project_directory* : comme illustré par la figure 3.7, celui-ci contient alors la structure de base d'une application Symfony.

A l'intérieur du dossier, ouvrez à nouveau la console et tapez les instructions 3.2 afin d'installer API Platform et Doctrine dans notre projet :

```
composer require api && composer require doctrine && composer require maker-bundle --
    dev
```

Code Source 3.2 – Installation d'API Platform et de Doctrine dans le projet Symfony

Ouvrez le fichier *.env* et définissez la base de données que vous allez utiliser. Remplacez *user* et *password* par vos accès personnels au DBMS :

```
...
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
DATABASE_URL="mysql://user:password@127.0.0.1:3306/database_name?serverVersion=5.7"
# DATABASE_URL="postgresql://symfony:ChangeMe@127.0.0.1:5432/app?serverVersion=13"
...
```

Code Source 3.3 – Définition de l'accès à la base de données

Pour créer une nouvelle entité *Product*, tapez l'instruction suivante :

```
bin/console make:entity Product
```

Code Source 3.4 – Création d'une entité Product dans Symfony

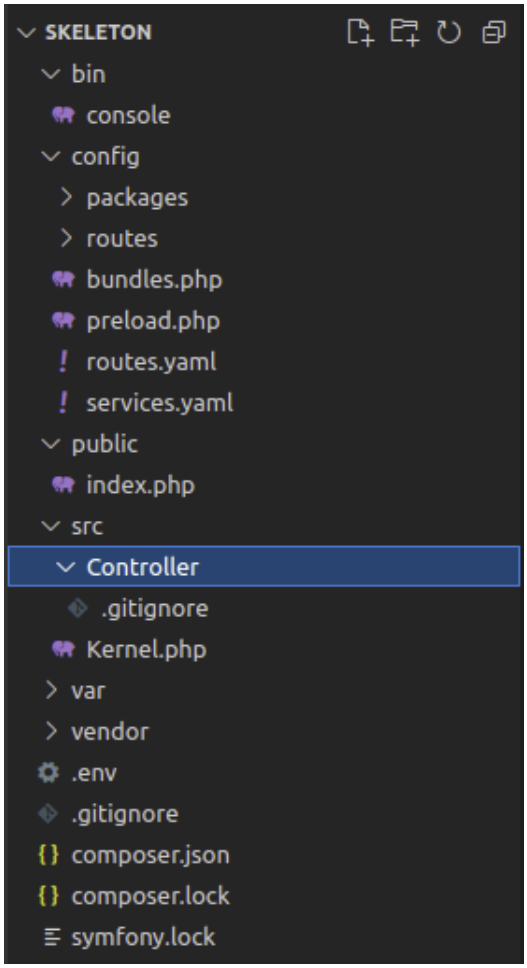

FIGURE 3.7. – Structure d'un Projet Symfony

Comme illustré par la figure 3.8, la console nous demande alors de manière interactive comment nous souhaitons procéder pour définir les champs associés à notre entité (leur type, la possibilité d'être nul, ...). Une fois une entité créée, il est possible de la retravailler en tapant à nouveau l'instruction 3.4.

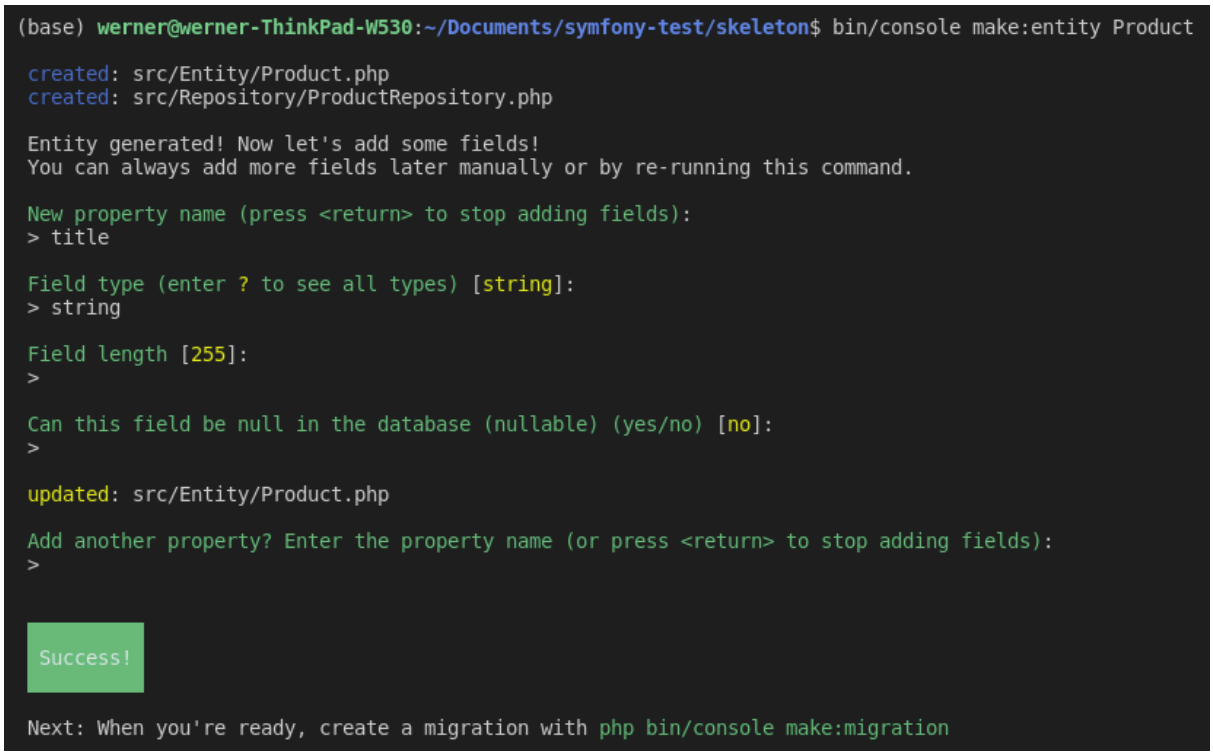

FIGURE 3.8. – Création d'une entité sur Symfony

Une fois le processus de création terminé, ouvrons le fichier Product.php, situé à l'emplacement src/Entity. Son contenu est illustré par le code source 3.5. Les lignes commençant par un "#" correspondent aux annotations que l'ORM va utiliser pour générer les tables dans la base de données.

```
1  <?php
2
3  namespace App\Entity;
4
5  use App\Repository\ProductRepository;
6  use Doctrine\ORM\Mapping as ORM;
7
8  #[ORM\Entity(repositoryClass: ProductRepository::class)]
9  #[ORM\Table(name: 'products')]
10 class Product
11 {
12     #[ORM\Id]
13     #[ORM\GeneratedValue]
14     #[ORM\Column(type: 'integer')]
15     private $id;
16
17     #[ORM\Column(type: 'string', length: 255)]
18     private $title;
19
```

```
20     public function getId(): ?int
21     {
22         return $this->id;
23     }
24
25     public function getTitle(): ?string
26     {
27         return $this->title;
28     }
29
30     public function setTitle(string $title): self
31     {
32         $this->title = $title;
33
34         return $this;
35     }
36 }
```

Code Source 3.5 – Contenu du fichier Product.php

Une fois toutes les entités et leurs relations définies, créez fichier de migration à l'aide de l'instruction 3.6 et ouvrez le fichier VersionXXX.php situé à l'emplacement migrations. Comme illustré par le code source 3.7, on voit qu'il contient les requêtes permettant de créer les tables de la base de données.

```
1 bin/console make:migration
```

Code Source 3.6 – Création du fichier de migration

```
1  <?php
2
3  declare(strict_types=1);
4
5  namespace DoctrineMigrations;
6
7  use Doctrine\DBAL\Schema\Schema;
8  use Doctrine\Migrations\AbstractMigration;
9
10 /**
11  * Auto-generated Migration: Please modify to your needs!
12  */
13 final class Version20220227150127 extends AbstractMigration
14 {
15     public function getDescription(): string
16     {
17         return '';
18     }
19
20     public function up(Schema $schema): void
21     {
22         // this up() migration is auto-generated, please modify it to your needs
23         $this->addSql('CREATE TABLE products (id INT AUTO_INCREMENT NOT NULL, title
               VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4
               COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
24     }
25
26     public function down(Schema $schema): void
```

```
27     {
28         // this down() migration is auto-generated, please modify it to your needs
29         $this->addSql('DROP TABLE products');
30     }
31 }
```

Code Source 3.7 – Contenu du fichier VersionXXX.php

On peut ensuite procéder à la migration à l'aide de l'instruction 3.8 et voir son résultat dans la base de données : la table *products* a bien été créée et contient les champs *id* et *title*, comme illustrée par la figure 3.9.

```
1 bin/console doctrine:migrations:migrate -n -q
```

Code Source 3.8 – Création de la base de données

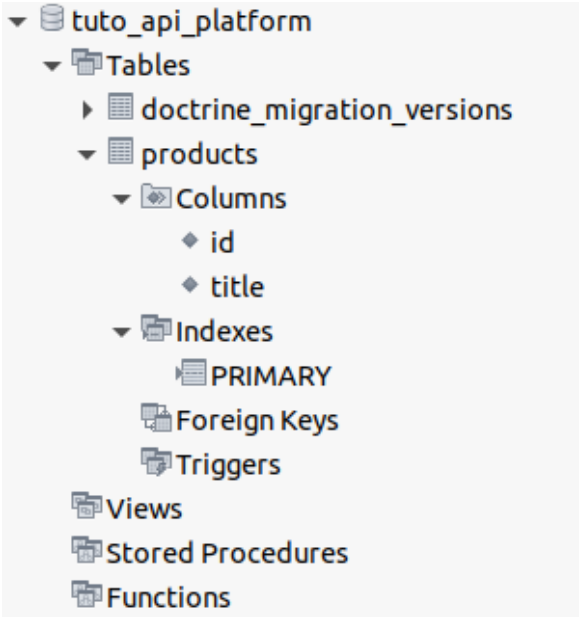

FIGURE 3.9. – Base de données après avoir procédé à la migration

Pour transformer cette entité en ressource API, il suffit d'annoter la classe Product.php, comme illustré par le code source 3.9.

```
1 ...
2 #[ApiResource()]
3 class Product
4 {
5 ...
```

Code Source 3.9 – Annotation de la classe Product.php

Lancez le serveur en local avec l'instruction 3.10 et allez à l'adresse `http://localhost:8000/api` sur votre navigateur. On constate, comme montré par la figure 3.10, que la description de la ressource et de ses endpoints a été automatiquement créée. On peut ensuite customiser la ressource pour avoir les endpoints et les requêtes que nous souhaitons, point qui ne sera pas abordé dans cette section.

```
1 php -S 127.0.0.1:8000 -t public
```

Code Source 3.10 – Lancement du serveur en local

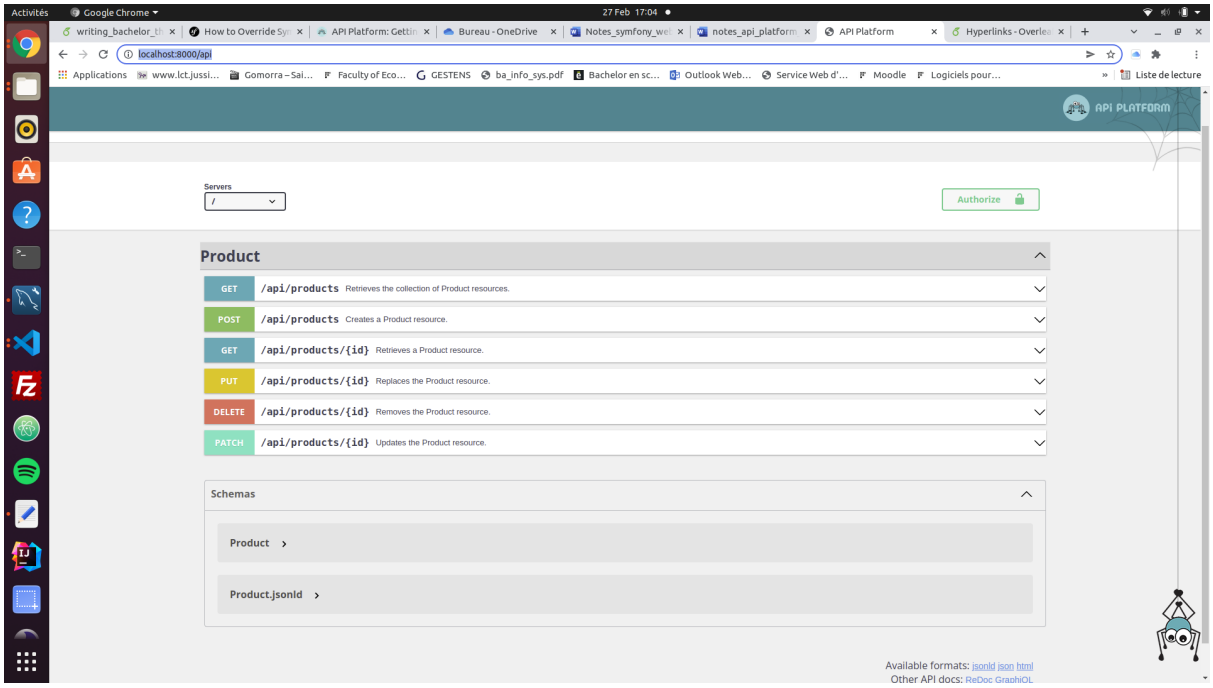

FIGURE 3.10. – Description de l'API générée

3.7. Conceptualisation de notre serveur

Maintenant que l'on a abordé les concepts théoriques et expliqué comment fonctionnent les technologies que l'on va utiliser, nous allons maintenant voir comment notre serveur a été défini. L'implémentation technique des endpoints et de la base de données ne sera pas abordée dans ce rapport.

3.7.1. Modélisation de la base de données

La structure de la base de données du prototype est représentée par la figure 3.12, qu'on nomme un diagramme entité-association. La signification des liens est détaillée par la figure 3.11. Un rectangle représente une entité, un losange une relation et un triangle une relation d'héritage.

En tenant compte de ces informations, on peut lire la relation Publication-Utilisateur de la figure 3.12 comme ceci : «un utilisateur publie aucune, une ou plusieurs publications et une publication est publiée par un utilisateur ». Les deux triangles du diagramme sont annotés par le commentaire «disjoint/complet», qui signifie que l'entité mère est forcément une instance de l'une des deux filles et ne peut pas être les deux à la fois.

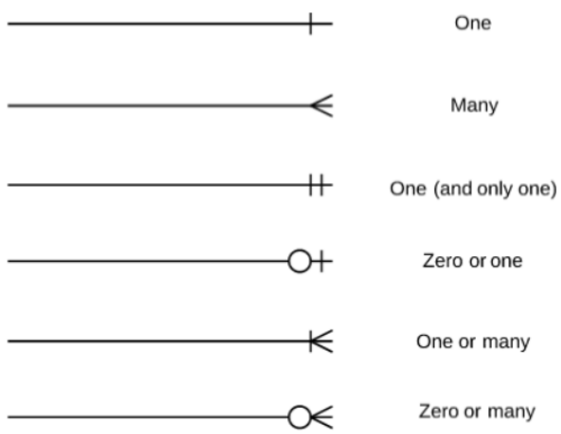

FIGURE 3.11. – Significations de la notation

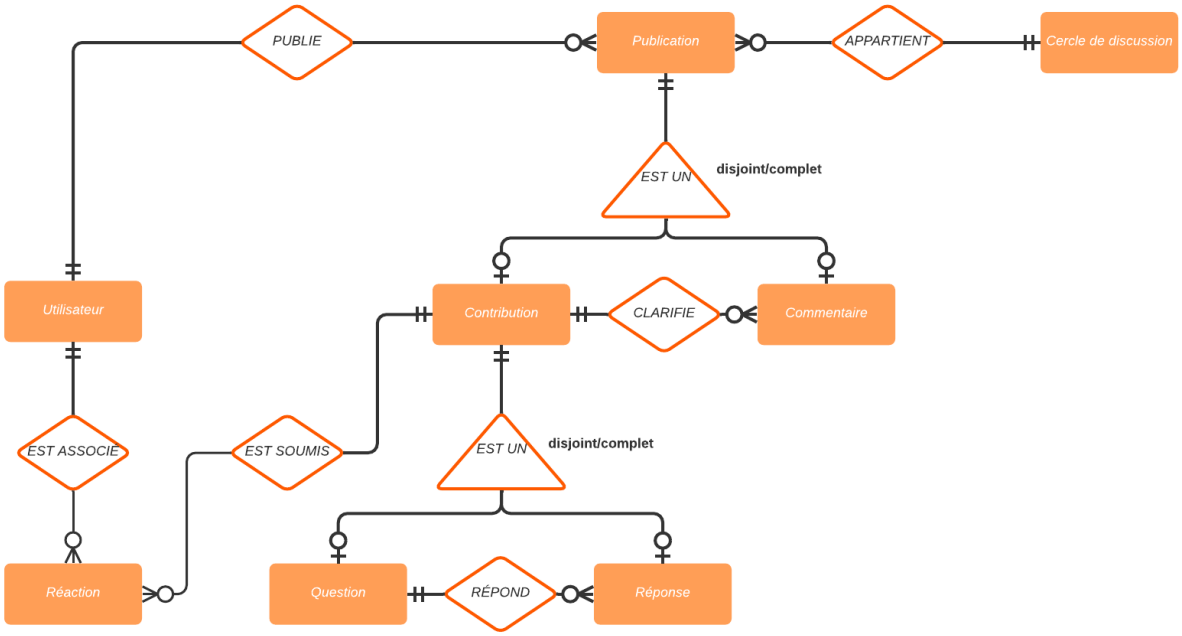

FIGURE 3.12. – Diagramme Entité-Association

Utiliser un modèle entité-association comporte l'avantage de décrire précisément la structure d'une base de données, sans avoir à rentrer dans des détails techniques. On peut facilement s'y référer lorsqu'on commence l'implémentation.

Voici une explication textuelle du diagramme :

— un user est un participant à la FAQ.

— une publication est un texte publié sur la FAQ.

— une publication est soit un commentaire, soit une contribution.

— une contribution est soit une question, soit une réponse.

— un cercle de discussion est un thème associé à une publication sur la FAQ.

— une réaction est un like/dislike fait par un user sur une publication.

— un user publie aucune, une ou plusieurs publications et une publication est publiée par un user.

— une publication appartient à un cercle de discussion et un cercle de discussion comporte aucune, une ou plusieurs publications.

— une réaction est associée à un utilisateur et un utilisateur peut avoir fait aucune, une ou plusieurs réactions.

— une réaction concerne une contribution et une contribution peut être soumise à aucune, une ou plusieurs réactions.

— un commentaire clarifie un poste et un poste est clarifié par aucun, un ou plusieurs commentaires.

— une réponse répond à une question et une question comporte aucune, une ou plusieurs réponses.

Vous trouverez au point A.1 une description détaillée des tables et de leurs champs.

3.7.2. Endpoints définis dans le cadre de ce travail

Les endpoints ont été définis selon les six critères d'exigence suivants :

— la **description** comporte le titre que l'on veut donner au endpoint.

— l'**accès** énumère les utilisateurs autorisés à demander du contenu au endpoint.

— le **corps de la requête** décrit le contenu que le client doit envoyer au endpoint.

— les **paramètres** décrivent les identifiants à rajouter au endpoint.

— les **filtres** décrivent les possibilités de filtre du endpoint, comme la pagination ou l'ordonnancement.

— la **réponse** décrit le contenu à renvoyer au client en cas de succès de la requête.

Vous trouverez au point A.2 une description détaillée des endpoints.

4
Conclusion

Ce travail de bachelor m'a permis de comprendre des concepts utiles dans la mise en place d'une application web, comme savoir ce qu'est un endpoint, un ORM ou un service REST. J'ai pu également découvrir comment fonctionne un framework, en particulier Symfony. J'ai appris à me référer à la documentation, installer des nouvelles dépendances ou automatiser la création de nouvelles fonctionnalités grâce à l'invite de commandes.

L'association Solid'Ark dispose à présent d'une base de travail solide pour mener à bien leur projet, surtout en ce qui concerne l'exploration des forums existants et l'analyse des besoins des utilisateurs. La structure et la méthodologie du rapport pourront être reprises pour modéliser de nouvelles fonctionnalités et étendre le backend de la plateforme.

Quelques difficultés ont été rencontrées lors de la création du prototype. Il a d'abord fallu découvrir le framework, comprendre ses concepts et son utilisation. Les tutoriels et cours en ligne ont été d'une aide précieuse pour comprendre le fonctionnement global. En ce qui concerne l'implémentation de points plus précis, j'ai dû procéder à une lecture plus détaillée de la documentation et parfois tester plusieurs solutions d'architecture pour voir lesquels arrivent au résultat voulu.

Pour finir, voici les pistes d'amélioration possibles de la plateforme :

- l'association souhaite donner la possibilité aux participants de financer, proposer ou participer à des projets de développement. Pour ces points-là, on pourrait donc procéder à une exploration des plateformes existantes et modéliser les besoins des utilisateurs.
- on pourrait également implémenter le frontend du prototype, afin de disposer d'une bonne piste d'exploration sur le design qu'on souhaiterait avoir sur le site.
- on pourrait étendre le backend du prototype en procédant à la modélisation complète de la base de données et des endpoints. On reprendrait les cas d'utilisation qui n'ont pas été implémentés dans le cadre de ce travail.

A
Annexes

A.1. Tables de la base de données

CHAMP	TYPE	SPÉCIFICITÉS
ID	INTEGER	CLÉ PRIMAIRE
USERNAME	STRING	UNIQUE
PASSWORD	STRING	MOT DE PASSE HASHÉ
EMAIL	STRING	UNIQUE DE TYPE EMAIL

TABLE A.1. – Table users

CHAMP	TYPE	SPÉCIFICITÉS
ID	INTEGER	CLÉ PRIMAIRE
NAME	STRING	UNIQUE
DESCRIPTION	TEXT	-

TABLE A.2. – Table circles

CHAMP	TYPE	SPÉCIFICITÉS
ID	INTEGER	CLÉ PRIMAIRE
CONTENT	TEXT	-
CREATOR	INTEGER	FK USERS(ID)
CREATION_DATE	DATETIME	PAS NULL
BELONGING_CIRCLE	INTEGER	FK CIRCLES(ID)

TABLE A.3. – Table publications

CHAMP	TYPE	SPÉCIFICITÉS
CONTRIBUTION_ID	INTEGER	CLÉ PRIMAIRE FK PUBLICATIONS(ID)

TABLE A.4. – Table contributions

CHAMP	TYPE	SPÉCIFICITÉS
COMMENT_ID	INTEGER	CLÉ PRIMAIRE FK PUBLICATIONS(ID)
RELATED_CONTRIBUTION	INTEGER	FK CONTRIBUTIONS(ID)

TABLE A.5. – Table comments

CHAMP	TYPE	SPÉCIFICITÉS
QUESTION_ID	INTEGER	CLÉ PRIMAIRE FK CONTRIBUTIONS(ID)
TITLE	STRING	UNIQUE

TABLE A.6. – Table questions

CHAMP	TYPE	SPÉCIFICITÉS
ANSWER_ID	INTEGER	CLÉ PRIMAIRE FK CONTRIBUTIONS(ID)
RELATED_QUESTION	INTEGER	FK QUESTIONS(ID)

TABLE A.7. – Table answers

CHAMP	TYPE	SPÉCIFICITÉS
USER_ID	INTEGER	CLÉ PRIMAIRE FK USERS(ID)
CONTRIBUTION_ID	INTEGER	CLÉ PRIMAIRE FK CONTRIBUTIONS(ID)
STATUS	INTEGER	0 : LIKE 1 : DISLIKE

TABLE A.8. – Table reactions

A.2. Endpoints

A.2.1. Authentification

POST auth/register

Description	Inscription de l'utilisateur
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	Données de base de l'utilisateur
Paramètres	-
Filtres	-
Réponse	Utilisateur créé

TABLE A.9. – POST auth/register

POST auth/login

Description	Authentification de l'utilisateur
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	E-mail et mot de passe de l'utilisateur
Paramètres	-
Filtres	-
Réponse	Token d'authentification

TABLE A.10. – POST auth/login

A.2.2. Circle

GET circles

Description	Liste des cercles de discussion existants
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	-
Paramètres	-
Filtres	-
Réponse	Liste des cercles de discussion

TABLE A.11. – GET circles

GET circles/{id}

Description	Données d'un cercle de discussion existant
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	-
Paramètres	Identifiant du cercle de discussion
Filtres	-
Réponse	Données du cercle de discussion demandé

TABLE A.12. – GET circles/{id}

GET circles/{id}/questions

Description	Questions associées à un cercle de discussion
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	-
Paramètres	Identifiant du cercle de discussion
Filtres	Pagination, Ordonnancement
Réponse	Liste des questions du cercle de discussion demandé

TABLE A.13. – GET circles/{id}/questions

POST circles/{id}/questions

Description	Publication d'une question sur un cercle de discussion
Accès	Utilisateur authentifié
Corps de l'en-tête	Token d'authentification
Corps de la requête	Données de la question à publier
Paramètres	Identifiant du cercle de discussion
Filtres	-
Réponse	Question créée

TABLE A.14. – GET circles/{id}/questions

A.2.3. Publication

DELETE publications/{id}

Description	Suppression d'une publication et de ses publications associées (réponses et commentaires)
Accès	Utilisateur authentifié
Corps de l'en-tête	Token d'authentification
Corps de la requête	-
Paramètres	Identifiant du cercle de discussion
Filtres	-
Réponse	Publication supprimée ou erreur si l'utilisateur n'est pas l'owner

TABLE A.15. – DELETE publications/{id}

A.2.4. Contribution

GET contributions/{id}/comments

Description	Liste des commentaires d'une contribution
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	-
Paramètres	Identifiant de la contribution
Filtres	-
Réponse	Liste des commentaires associés à la contribution

TABLE A.16. – GET contributions/{id}/comments

GET contributions/{id}/like

Description	Nombre de likes associés à une contribution
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	-
Paramètres	Identifiant de la contribution
Filtres	-
Réponse	Nombre de likes associés à la contribution

TABLE A.17. – GET contributions/{id}/like

GET contributions/{id}/dislike

Description	Nombre de dislikes associés à une contribution
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	-
Paramètres	Identifiant de la contribution
Filtres	-
Réponse	Nombre de dislikes associés à la contribution

TABLE A.18. – GET contributions/{id}/dislike

POST contributions/{id}/comments

Description	Publication d'un commentaire sur un poste donné
Accès	Utilisateur authentifié
Corps de l'en-tête	Token d'authentification
Corps de la requête	Données du commentaire à publier
Paramètres	Identifiant de la contribution
Filtres	-
Réponse	Commentaire créé

TABLE A.19. – POST contributions/{id}/comments

POST contributions/{id}/like

Description	Publication d'un like sur un poste donné
Accès	Utilisateur authentifié
Corps de l'en-tête	Token d'authentification
Corps de la requête	-
Paramètres	Identifiant de la contribution
Filtres	-
Réponse	Like publié

TABLE A.20. – POST contributions/{id}/like

POST contributions/{id}/dislike

Description	Publication d'un dislike sur un poste donné
Accès	Utilisateur authentifié
Corps de l'en-tête	Token d'authentification
Corps de la requête	-
Paramètres	Identifiant de la contribution
Filtres	-
Réponse	Dislike publié

TABLE A.21. – POST contributions/{id}/dislike

DELETE contributions/{id}/like

Description	Suppression d'un like sur un poste donné
Accès	Utilisateur authentifié
Corps de l'en-tête	Token d'authentification
Corps de la requête	-
Paramètres	Identifiant de la contribution
Filtres	-
Réponse	Like supprimé

TABLE A.22. – DELETE contributions/{id}/like

DELETE contributions/{id}/dislike

Description	Suppression d'un dislike sur un poste donné
Accès	Utilisateur authentifié
Corps de l'en-tête	Token d'authentification
Corps de la requête	-
Paramètres	Identifiant de la contribution
Filtres	-
Réponse	Dislike supprimé

TABLE A.23. – DELETE contributions/{id}/dislike

A.2.5. Question

GET questions

Description	Liste de toutes les questions existantes
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	-
Paramètres	-
Filtres	Pagination, Ordonnancement
Réponse	Liste des questions existantes

TABLE A.24. – GET questions

GET questions/{id}

Description	Données d'une question existante
Accès	Utilisateur non authentifié
Corps de l'en-tête	-
Corps de la requête	-
Paramètres	Identifiant de la question
Filtres -	
Réponse	Données de la question demandée

TABLE A.25. – GET questions/{id}

POST questions/{id}/answers

Description	Publication d'une réponse à une question donnée
Accès	Utilisateur authentifié
Corps de l'en-tête	Token d'authentification
Corps de la requête	Données de la réponse à publier
Paramètres	Identifiant de la question
Filtres -	
Réponse	Réponse créée

TABLE A.26. – POST questions/{id}/answers

B
Acronymes utilisés

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CMS	Content Management System
CSS	Cascading Style Sheet
DBMS	Database Management System
DNS	Domain Name System
DOM	Document Object Model
IDL	Interfae Description Language
ERM	Entity Relationship Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
ORM	Object Relational Mapping
PHP	PHP : Hypertext Processor
POO	Programmation Orientée Objet
REST	Representational State Transfer
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language
URI	Unified Resource Identifier
URL	Uniform Resource Locator
XML	eXtensible Markup Language

Sites Web

[1] API Platform : REST and GraphQL framework on top of Symfony and React. https://api-platform.com/ (dernière consultation le March 07, 2022). 34

[2] Comment installer et utiliser Composer sur Ubuntu 20.04. https://www.digitalocean.com/community/tutorials/how-to-install-and-use-composer-on-ubuntu-20-04-fr (dernière consultation le February 27, 2022).

[3] Object Relational Mapper - Doctrine : PHP Open Source Project. https://www.doctrine-project.org/projects/orm.html (dernière consultation le March 07, 2022). 34

[4] Apprendre à développer des applications web avec PHP et Symfony. https://www.editions-eni.fr/open/mediabook.aspx?idR=d201d593a72dd2c8a78478f7b6529709 (dernière consultation le February 27, 2022).

[5] Best Most Popular Forums, Message Boards Online Communities – Top 30. https://it-maniacs.com/best-and-most-popular-forums-message-boards-and-online-communities-top-30 (dernière consultation le January 26, 2022). 5

[6] MySQL. https://www.mysql.com/ (dernière consultation le March 07, 2022). 30

[7] CakePHP 3 - Le framework PHP pour le développement de vos applications web - Qu'est-ce qu'un ORM ? | Editions ENI. https://www.editions-eni.fr/open/mediabook.aspx?idR=4714ebb7e87f7c09fb040952bfd0b871 (dernière consultation le February 23, 2022).

[8] Quora. https://www.quora.com/ (dernière consultation le March 07, 2022). 10

[9] Qu'est-ce que Quora ? Tout savoir sur le portail de questions - IONOS. https://www.ionos.fr/digitalguide/web-marketing/analyse-web/quora/ (dernière consultation le January 29, 2022). 10

[10] Reddit - Dive into anything. https://www.reddit.com/ (dernière consultation le March 07, 2022). 5

[11] Que signifie Reddit ? - Definition IT de Whatis.fr. https://whatis.techtarget.com/fr/definition/Reddit (dernière consultation le January 26, 2022). 5

[12] Une API REST, qu'est-ce que c'est ? https://www.redhat.com/fr/topics/api/what-is-a-rest-api (dernière consultation le February 22, 2022). 29

[13] Stack Overflow - Where Developers Learn, Share, Build Careers. https://stackoverflow.com/ (dernière consultation le March 07, 2022). 15

[14] API Documentation Design Tools for Teams | Swagger. https://swagger.io/ (dernière consultation le March 07, 2022). 32

[15] Symfony, High Performance PHP Framework for Web Development. https://symfony.com/ (dernière consultation le March 07, 2022). 34

[16] Welcome To UML Web Site! https://www.uml.org/ (dernière consultation le March 08, 2022). 21

Index