Université de Rouen

Amicale GIL

GIL-2

Document d'Architecture Logicielle

Rédigé par :
Islam GUETTOUCHE

Relu par :
L'ensemble de l'équipe

Date : 25 Janvier 2017
Version: 2.0

1 MISE A JOUR

Version	Date	Modifications réalisées
0.1	12/11/2016	Création du document d'architecture
0.2	30/11/2016	Ajout du contenu et des modifications
1.0	10/01/2017	Modifications finales
2.0	25/01/2017	Modifications suite aux retours de Mr Godart

Table des matières

1. OBJET

Ce document est le Document d'Architecture Logicielle de l'application web Amicale GIL.

Dans ce présent document, sont exposées les modifications apportées à ce qui existe déjà comme architecture et les différents composants où on doit intervenir, avec notamment l'ajout de quelques composants et la modification de quelques autres.

Pour l'architecture, l'ancienne version se basait sur l'architecture MVC avec l'emploi du Framework Symfony 2 d'où la nécessité de se reposer la dessus pour continuer le développement et arriver au résultat escompté et satisfaire les besoins clients.

Par ailleurs, des modifications s'imposent au niveau du diagramme de classe, et cela dans le but d'intégrer de nouvelles fonctionnalités que souhaite le client.

Pour conclure, on a intégré de nouveaux diagrammes de séquences, notamment ceux qui concernent les nouvelles fonctionnalités qu'on doit développer ainsi que ceux concernant les fonctionnalités à modifier.

Ainsi donc, ce document consiste à mettre à jour l'architecture déjà mise en place, ainsi qu'à bâtir une architecture évolutive pour pouvoir ajouter de nouvelles fonctionnalités sans trop de peine dans le futur.

2. DOCUMENTS APPLICABLES ET DE REFERENCES

- Spécification Technique des Besoins (STB)
- Documentation officielle de Symfony2
- Document d'architecture déjà existant (l'ancienne version)

3. TERMINOLOGIES ET SIGLES UTILISES

Back End : Partie traitant les données dans une application. Face cachée de l'application.

BDD : Base de données.

Front End : Partie affichant les données dans une application. Face visible de l'application.

GIL : Génie de l'Informatique Logicielle.

MVC : Model View Controller (Design d'application Modèle Vue Controlleur).

PHP : PHP Hypertext Preprocessor.

Responsive Web Design : Interface graphique qui s'adaptent à toute taille d'écran (mobile, smartphone, tablette).

Route : correspond à une URL relative particulière (ex : /login).

SGBD : Système de Gestion de Bases de Données.

STB : Spécification Technique des Besoins.

URL: Uniform Resource Locator.

4. CONFIGURATION REQUISE

4.1 ENVIRONNEMENT

Matériel

L'application web sera installée sur un serveur physique au sein de l'Université de Rouen.

Le bon fonctionnement de l'application nécessite une connexion Internet stable et activée 99% du temps.

Logiciel

Le système d'exploitation du serveur sera de type Ubuntu Serveur 16.04 LTS x64 et fonctionnera 99% du temps.

Le serveur web utilisé sur ce système d'exploitation sera de type : Nginx 1.4.6 et sera fonctionnel 99% du temps.

Le système de gestion de base de données (SGBD) sera de type : MySQL (5.5.44) et sera fonctionnel 99% du temps.

Le langage supporté sur le serveur web est le suivant : PHP (5.5.9).

L'application Web sera optimisée pour les navigateurs Chrome et Firefox dans leurs versions les plus à jour.

4.2 OUTILS NECESSAIRES

Symfony : Framework PHP (version 2.7).

Composer : Gestionnaire de dépendances pour PHP.

FOSUserBundle : Bundle à intégrer dans Symfony pour gérer la partie utilisateur et droits d'accès.

TinyMCEBundle : Bundle à intégrer dans Symfony pour bénéficier d'un éditeur de texte pour les messages du forum.

5. ARCHITECTURE STATIQUE

6.1 STRUCTURE

L'application suit le patron de conception MVC. Ce dernier est constitué de plusieurs couches :

Base de données contient toutes les données de l'application sous MySQL 5.5.44 et fonctionnera grâce à Doctrine qui fera l'interface entre le code PHP et la base de données.

Modèle regroupe les données à travers des entités et interagit avec la base de données pour échanger ces données.

Vue présente les résultats des modèles et des différentes opérations à travers des interfaces visuelles présentes sur le navigateur, en lien direct avec l'utilisateur.

Contrôleur gère les évènements de synchronisation des vues et des modèles. Il permet de rendre l'application entièrement dynamique.

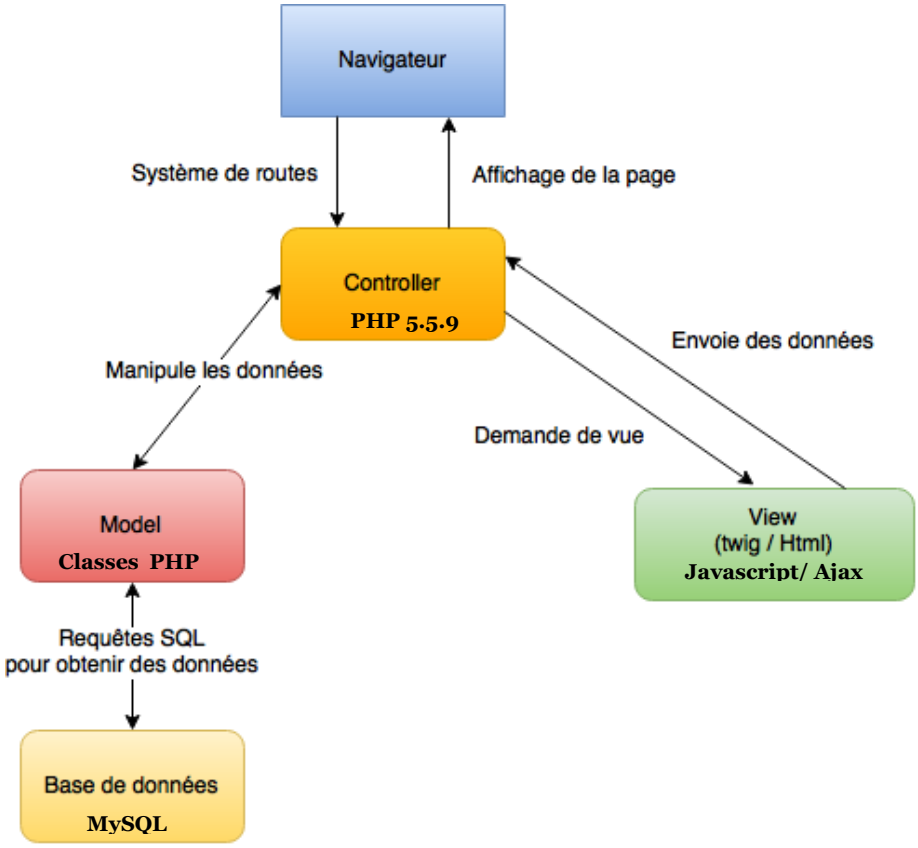

Figure 1 – Architecture générale (sous Symfony)

6.2 DESCRIPTION DES CONSTITUANTS

6.2.1 ARCHITECTURE DU PROJET SYMFONY 2

Le framework Symfony dispose d'une architecture assez précise et bien organisée afin de respecter le modèle MVC et d'intégrer d'autres bibliothèques. Il est composé de différents fichiers de paramètres et de dossiers, dont voici les principaux :

```
app/
|
|----AppKernel.php # sert à inclure les bundles du    projet
|
|----config/
|      |
|      |----parameters.yml # configuration de la base de   données
|      |----routing.yml       # configuration des routes du système
|      |----security.yml      # configuration des paramètres de   sécurité
|
|----Resource/
|      |
|      |----views
```

```
|    |    |
|    |    |----base.html.twig
```

```
src/
|
|----AppBundle/ # Bundle par défaut du site
|
|----NomQuelconque/ # Bibliothèque créé par le développeur
|    |
|    |----NQ/
|    |    |
|    |    |----BundleQuelconque/ # Bundle créé par le développeur
|    |    |    |
|    |    |    |----Controller/ # Contient les contrôleurs du bundle
|    |    |    |----Resources/
|    |    |    |    |
|    |    |    |    |----config/
|    |    |    |    |    |
|    |    |    |    |    |----routing.yml
|    |    |    |    |    |----services.yml
|    |    |    |    |
|    |    |    |    |----views/
```

vendors/ # contient les fichiers géré par Composer

web/ # dossier public, seul dossier accessible par le visiteur de l'application

```
|
|----app.php        # fichier d'index de l'application (équivalent à index.php)
|----app_dev.php # identique à app.php mais utilisé lors du développement
|----js/            # contient les fichiers JavaScript
|----css/           # contient les fichiers CSS
|----img/           # contient les images du site
```

Le dossier **src** contient ce qu'on appelle des bundles. Les bundles sont tout simplement des sortes de "briques" d'un projet. Symfony2 utilise ce concept novateur qui consiste à regrouper dans un même endroit, le bundle, tout ce qui concerne une même fonctionnalité. Par ailleurs voici la liste des bundles développés par l'équipe précédente :

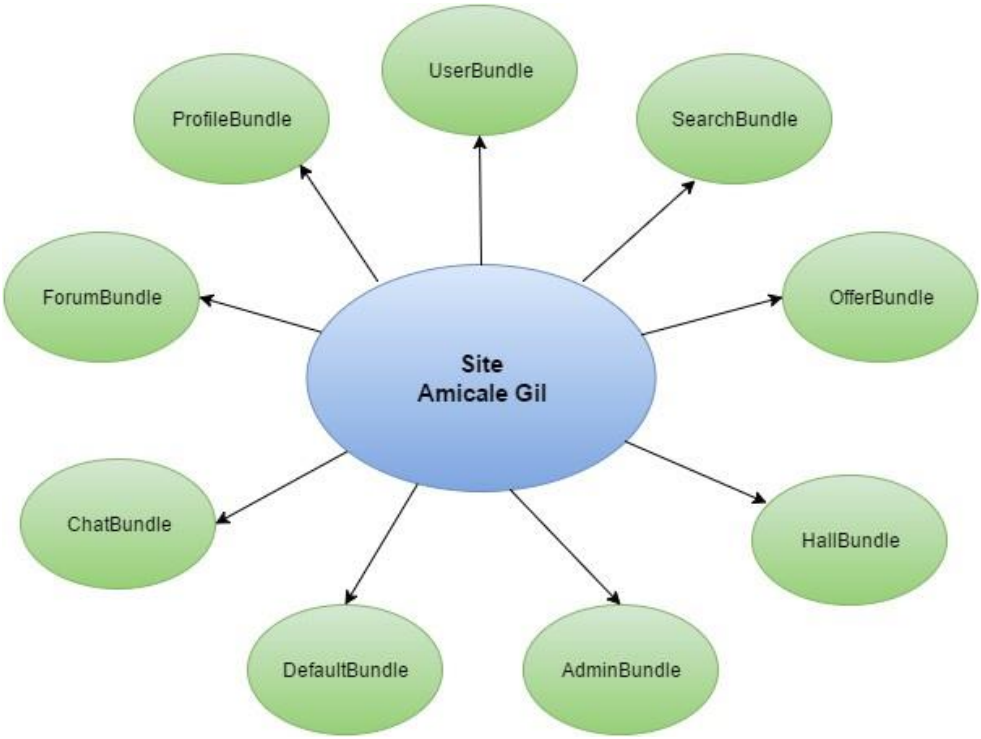

Figure 2 – Les bundles de notre application

AdminBundle Bundle qui contient les fonctionnalités de l'administrateur, certaines sont à corriger et d'autres sont à améliorer.

ChatBundle Bundle qui gérera le salon avec toutes les tables de discussions, qu'on va améliorer par la suite.

DefaultBundle Bundle qui contiendra la gestion de la page d'accueil (en mode connecté, non connecté), ainsi que les éléments qui sont en lien avec tout le site (comme les tags par exemple) mais qui nécessite une modification notamment l'ajout de nouveaux liens.

ForumBundle Bundle qui gérera le forum, avec les catégories, les sujets et les réponses, dans lequel on va intervenir pour ajouter la possibilité d'interagir avec le forum en autorisant à donner son avis sur les sujets et les commentaires.

HallBundle Bundle qui contiendra le hall d'exposition de l'application.

OfferBundle Bundle qui gérera les annonces avec l'espace pour les déposer, dans lequel on va intervenir pour permettre l'ajout de plusieurs offres à la fois.

ProfileBundle Bundle qui contiendra l'espace profil ainsi que l'espace pour le modifier ainsi que de le paramétrer.

SearchBundle Bundle qui gérera la recherche sur l'application de membre, sujets de forum, annonces, qu'on améliorera avec l'ajout de la recherche par annuaire.

UserBundle Bundle qui gérera toute la partie utilisateur (connexion, oublie de mot de passe, etc.).

En gros, on va intervenir dans tous les bundles soit pour modifier des fonctionnalités, ajouter des nouvelles ou bien réparer les bugs.

Ainsi qu'on va rajouter un nouveaux bundles qui sont :

MailingListBundle le bundle qui gérera les listes de diffusion avec les abonnés et les les annonces.

6.2.2 STRUCTURE DE LA BASE DE DONNEES ET MODELES

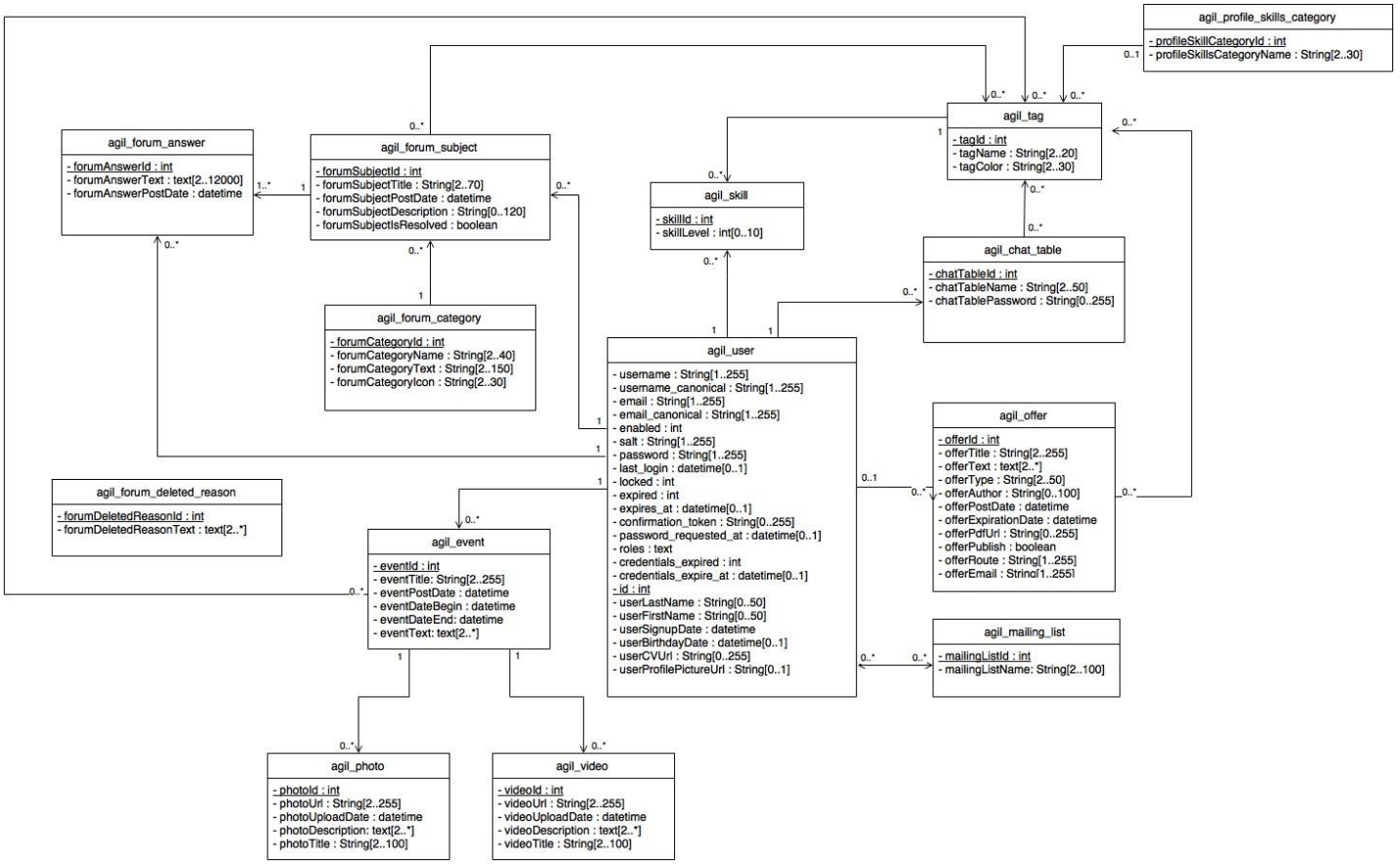

Figure 3 – Diagramme de classe de la BDD

Les modèles sont les entités de l'application qui feront l'échange de données entre le contrôleur et la base de données. Symfony intègre un outil ORM nommé Doctrine. Ce dernier se charge de faire la relation et le mapping entre les "objets PHP" (les modèles) et les données dans la BDD. Nous aurons donc dans notre application un modèle par table de la base de données. Les attributs du modèle seront tout simplement les attributs de la table.

Pour ce qui est de ce diagramme de classe, on a jugé utile de rajouter quelques liens et de le modifier par rapport aux nouvelles fonctionnalités qu'on doit développer, voici le diagramme de classe correspondant à l'amélioration.

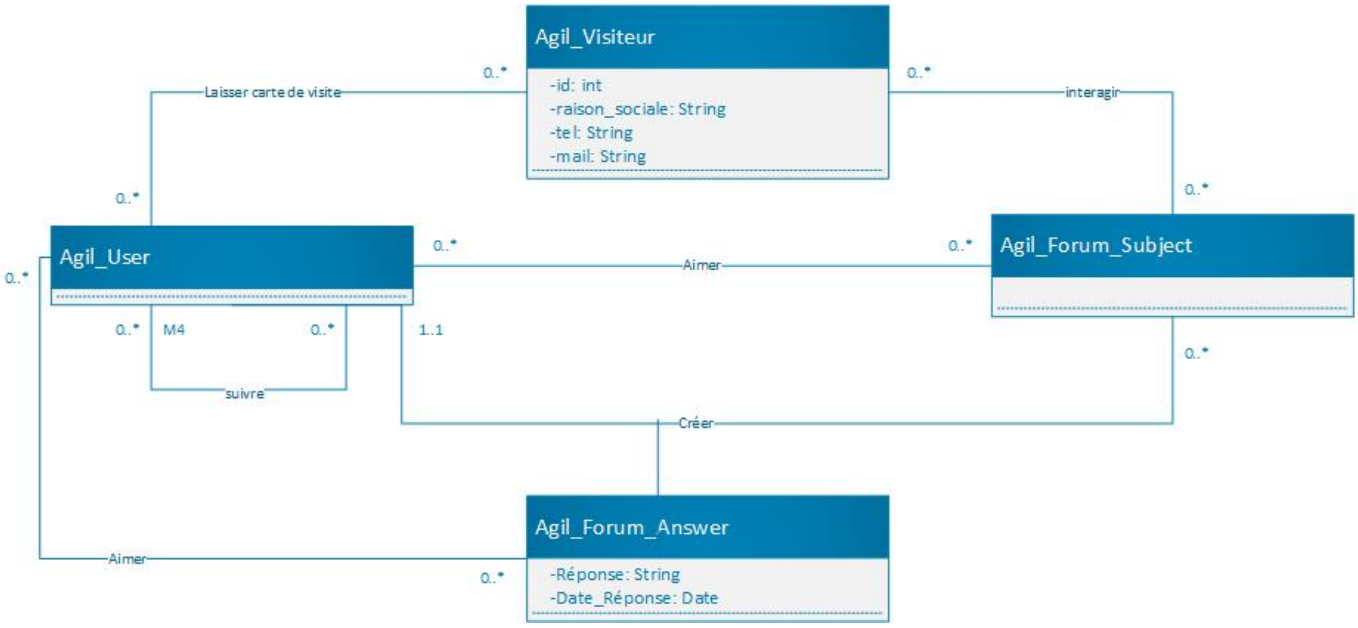

Figure 4 – Diagramme de classe de la partie à améliorer

Modifications apportées:

- **Visiteur :** Il aura la possibilité de :
  Laisser une carte de visite pour un ou plusieurs utilisateur(s) inscrit(s) à la plateforme.
  Consulter le profil réduit des membres inscrits.
  Voir les sujets publics du forum et pouvoir laisser une réponse.
- **Membre :** Il aura la possibilité de :
  Gérer un espace personnel (publier des billets).
  Suivre un/plusieurs membre(s).
  Signaler un message dans un forum.
  S'abonner ou se désabonner à un fil d'informations (mailing liste).
  Créer un sujet public ou privé.
- **Modérateur :** Il aura la possibilité de :
  Gérer les messages signalés.
  Maintenance des mailings liste.

- **Tous les acteurs :** Ils auront la possibilité de :
  Effectuer une rechercher par annuaire et en temps réel.
  Déposer plusieurs offres d'emplois et/ou stages.
  Aimer les sujets du forum.
  Aimer les réponses à un sujet d'un forum.

6.2.3 LES VUES

Les vues sont tout simplement les pages finales que l'utilisateur pourra voir sur son navigateur Web. Elles correspondent à la partie front-end de l'application et seront développées à l'aide du moteur de templating Twig intégré dans Symfony. De plus, nous utiliserons le framework CSS Twitter Bootstrap afin de posséder une architecture organisée et responsive.

Voici, un exemple d'interface de la page d'accueil en mode connecté (avec le menu ouvert):

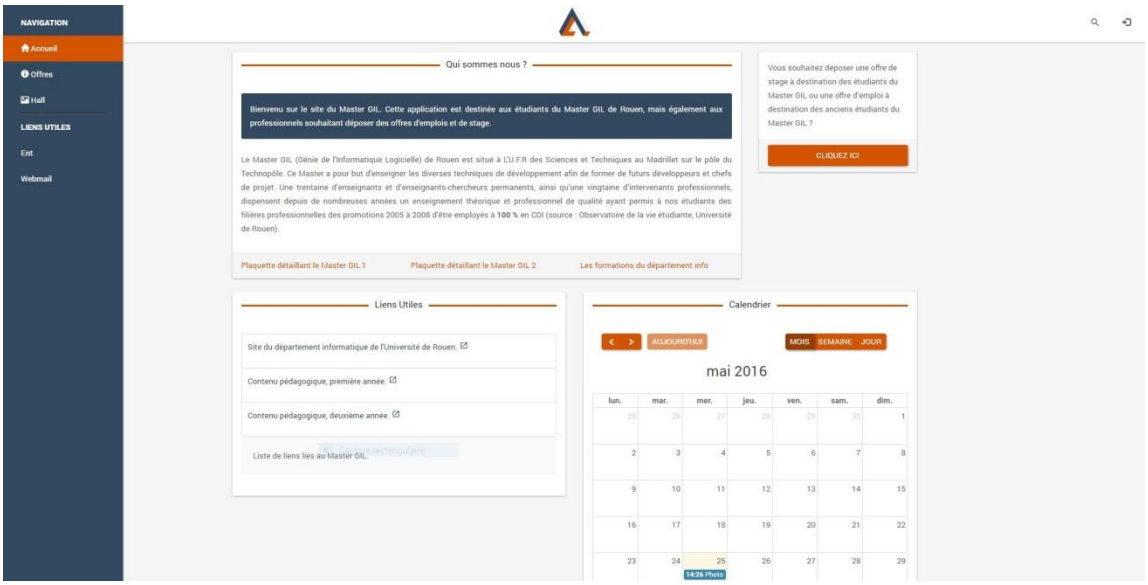

Figure 5 – Exemple d'interface (page d'accueil en mode non connecté)

Sur la partie gauche, nous retrouvons le menu qui permet d'accéder aux fonctionnalités princi- pales. Les items présents seront différents selon un visiteur ou un utilisateur connecté.

En haut, nous pouvons voir l'en-tête du site, le header. Celui-ci contient le logo de l'Université de Rouen, une mention "Amicale GIL" et quelques icônes selon le mode connecté/déconnecté.

Le corps du site contient les informations relatives à la page. Sur la page d'accueil en mode connecté, nous pourrons retrouver par exemple les dernières annonces publiées.

6.2.4 LES CONTROLEURS

Les contrôleurs gèrent les évènements de synchronisation des vues et des modèles. Ils permettent de rendre l'application entièrement dynamique et sont codés en PHP orienté objet.

Chaque URL relative (appelée également route) va exécuter un certain contrôleur de l'applica- tion.

Les routes présentes ci-dessous sont toutes appelées via la méthode GET (méthode HTTP par défaut). Cependant, certaines de ces routes peuvent être également appelées avec la méthode POST (lors d'appels via des formulaires).

6.3 JUSTIFICATION TECHNIQUE

PHP : Du fait des langages supportés sur le serveur Web, et des compétences de l'ensemble de l'équipe, nous avons choisi d'utiliser le langage PHP. Ce dernier est le langage de programmation Web le plus utilisé. Il existe une communauté de développeurs très active qui rend disponible de nombreuses bibliothèques et de documentations. Ces ressources amélioreront la production et le temps d'exécution de l'application.

Symfony 2.7 : En complément du langage web, nous utiliserons un framework PHP populaire en France nommé Symfony. Ce dernier est très utilisé dans les entreprises françaises et adopte une architecture MVC intéressante. Dans l'équipe, plusieurs personnes ont déjà développé avec Symfony, et d'autres ont déjà travaillé avec d'autres framework PHP. Il sera facile pour nous et intéressant de travailler avec Symfony pour ce projet. De plus, ce framework possède des aspects sécuritaires important dans le web, ce qui nous semble primordial pour le développement de l'Amicale GIL.

Twitter Bootstrap : Pour le CSS, nous avons choisi de partir sur le framework Twitter Bootstrap. Après avoir comparé avec d'autres framework CSS, nous avons décidé de partir sur celui de Twitter, qui est plus complet que les autres (Materialize, Foundation). L'idée principale d'utiliser un framework CSS est d'obtenir rapidement un design sobre et moderne, et que ce dernier soit entièrement responsive, c'est à dire qu'il s'adapte à toutes les tailles d'écrans (ordinateur, tablette, téléphone).

7 Diagrammes de séquences

7.1 Cas d'utilisation G01 : Ajouter des offres d'emplois et des stages

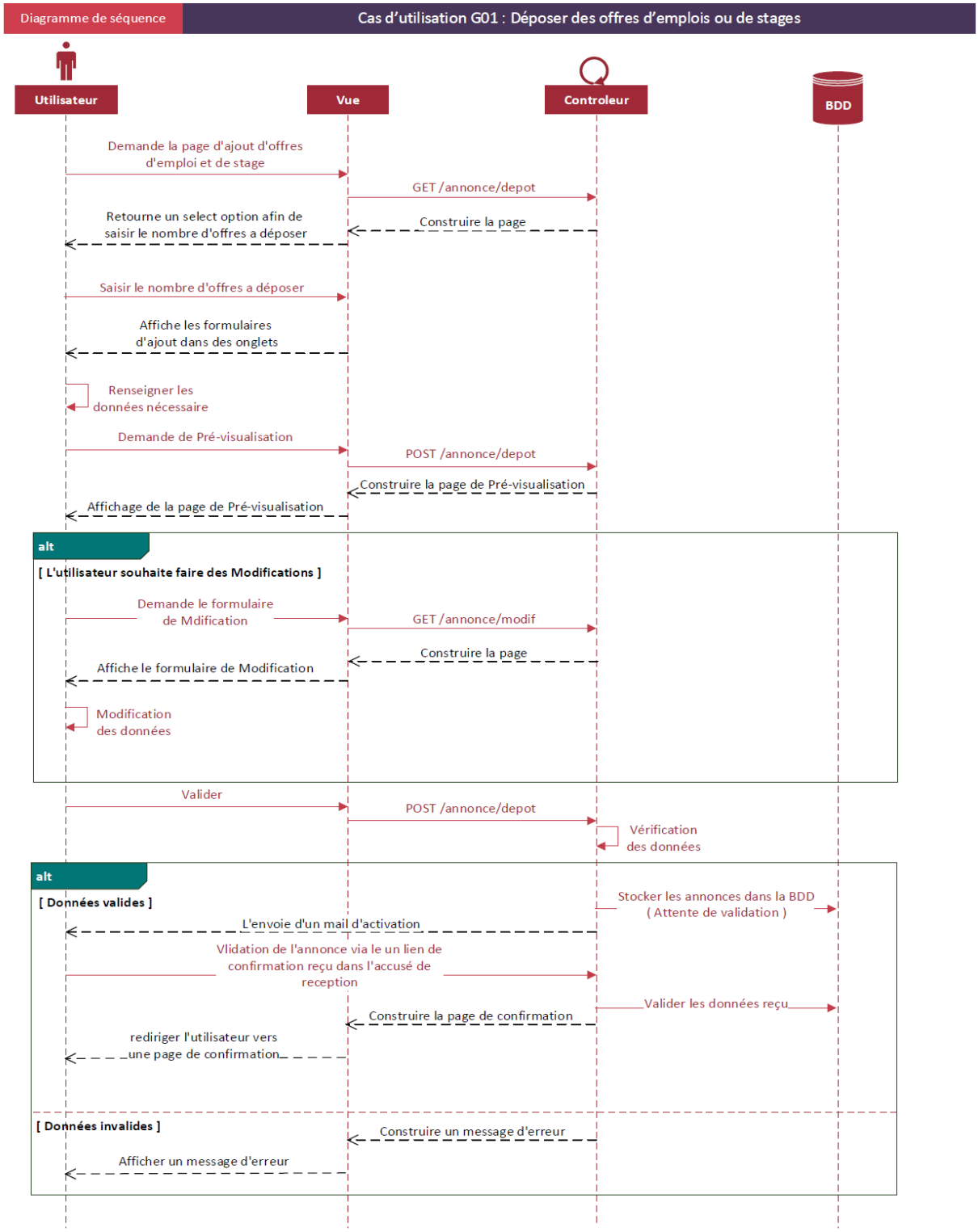

Figure 6 – Diagramme de séquence G01

Le diagramme ci-dessus va nous donner une vision de l’échange qui va être effectué à l'issu d’une demande d’ajout d’une ou de plusieurs offres à la fois. Cette demande peut être effectuée par tous les Acteurs pouvant accéder à l'application.

L’utilisateur commence par demander la page d’ajout d’offres d’emploi et de stage, le système va lui retourner une page qui va comporter un select option afin de saisir le nombre d'offres à déposer dès que l’utilisateur saisit le nombre d’offres a déposé le système va lui afficher le même nombre de formulaires demander dans des onglets sur la même page, l’utilisateur aura la possibilité de visualiser ses offres avant de les valider ainsi que de les modifier, dès qu’il soumet les formulaires, le système va tester la validité de ces données et va les stocker et envoi un mail de validation à l’utilisateur et dès que ce dernier valide le dépôt le système le redirige vers une page de confirmation, si les données qui ont été saisies ne sont pas valides alors le système affichera un message d’erreur.

7.2 Cas d'utilisation P02 : Accéder au paramètre profil

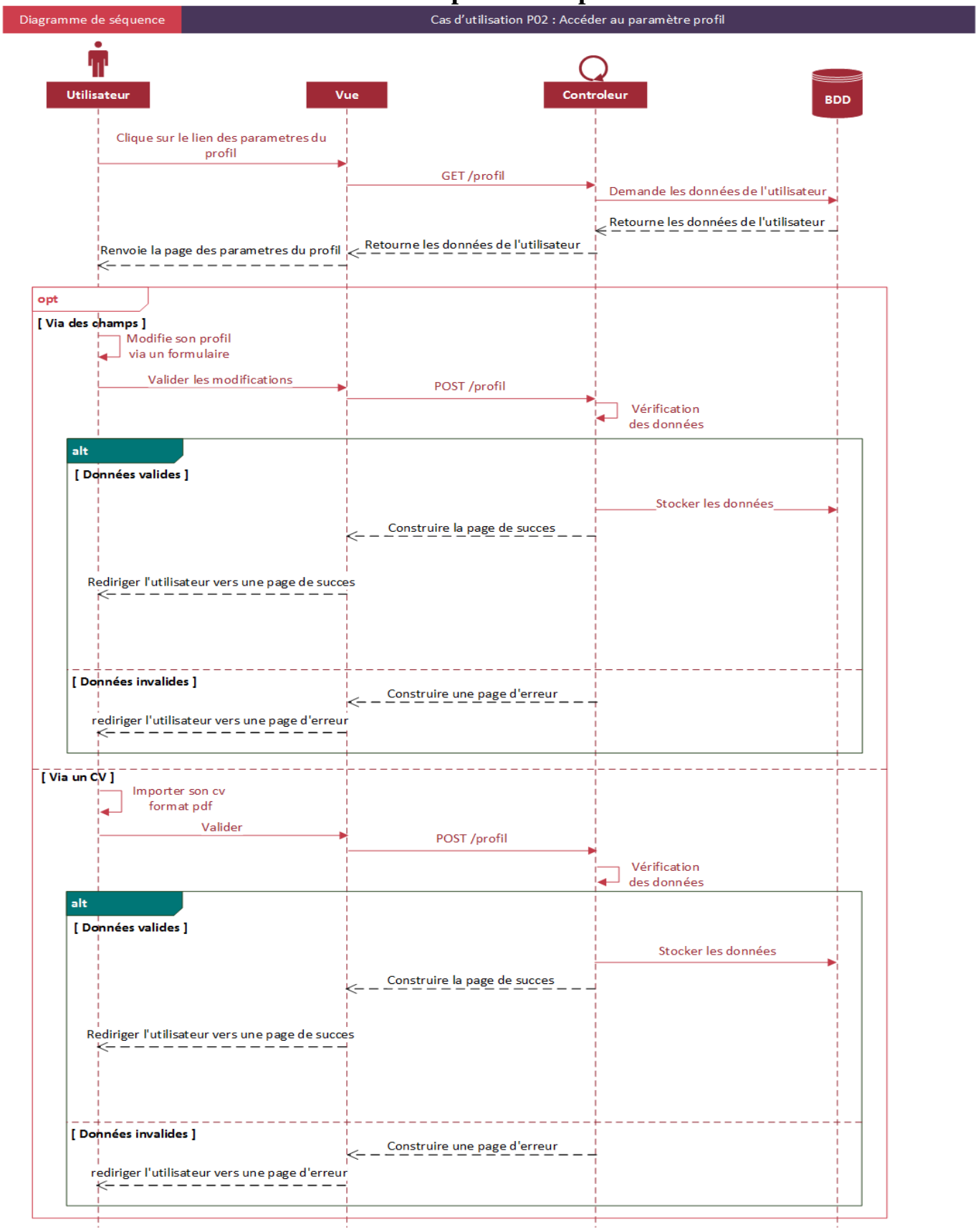

Figure 7 – Diagramme de séquence P02

Le diagramme ci-dessus va nous donner une vision de l’échange qui va être effectué à l'issu d’une demande d’accès au paramètre du profil. Cette demande peut être effectuée par tous les Acteurs pouvant accéder à l'application sauf le Visiteur.

L'utilisateur commence par demander l'accès au paramètre de son profil, le système va lui retourner une page qui va comporter toutes les données qui ont été enregistrées auparavant sur son profil. L'utilisateur aura la possibilité de modifier son profil via un formulaire ou en important un CV format PDF.

7.3 Cas d'utilisation G03 : Consulter le profil réduit

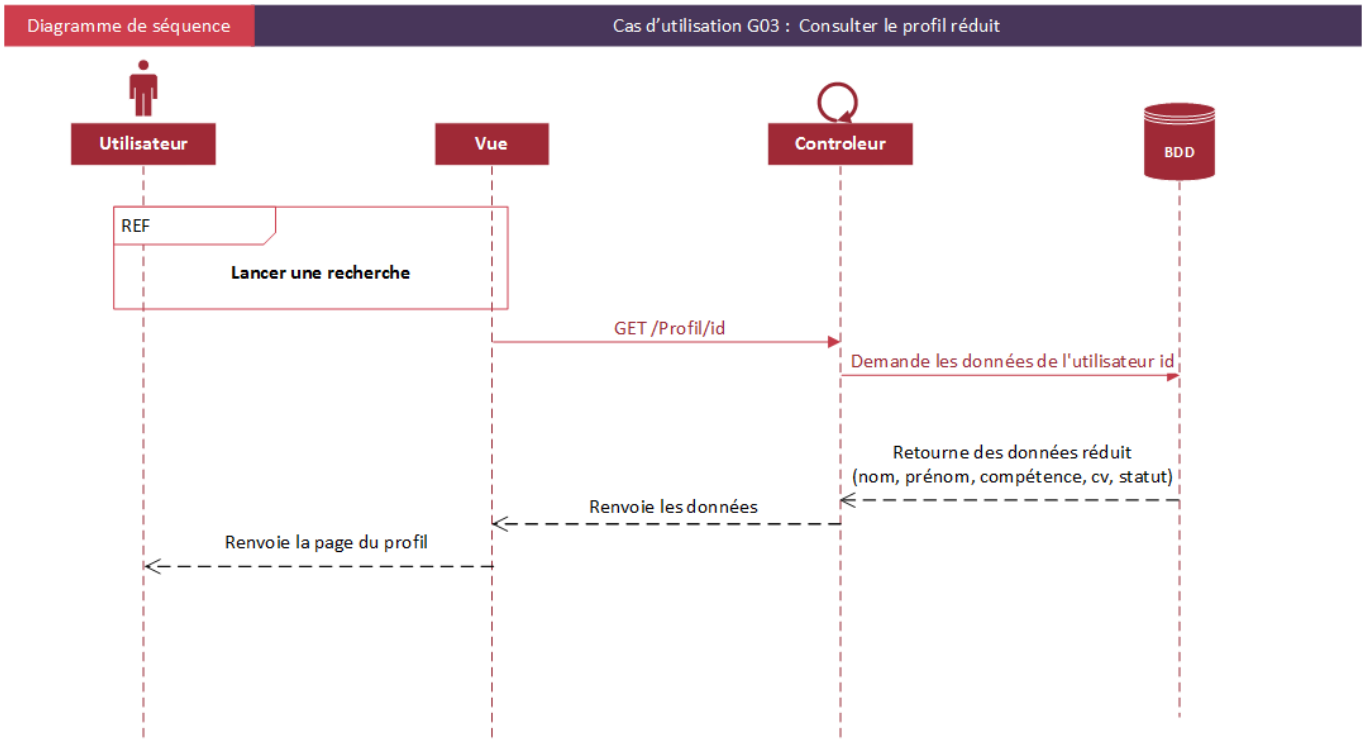

Figure 8 – Diagramme de séquence G03

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'une demande de consultation d'un profil d'un utilisateur donné. Cette action concerne le Visiteur.

L'utilisateur lance une recherche afin de trouver un membre précis par exemple après, il aura juste à cliquer sur ce profil afin d'avoir une vision réduite sur ce dernier.

7.4 Cas d'utilisation G04 : Effectuer une recherche

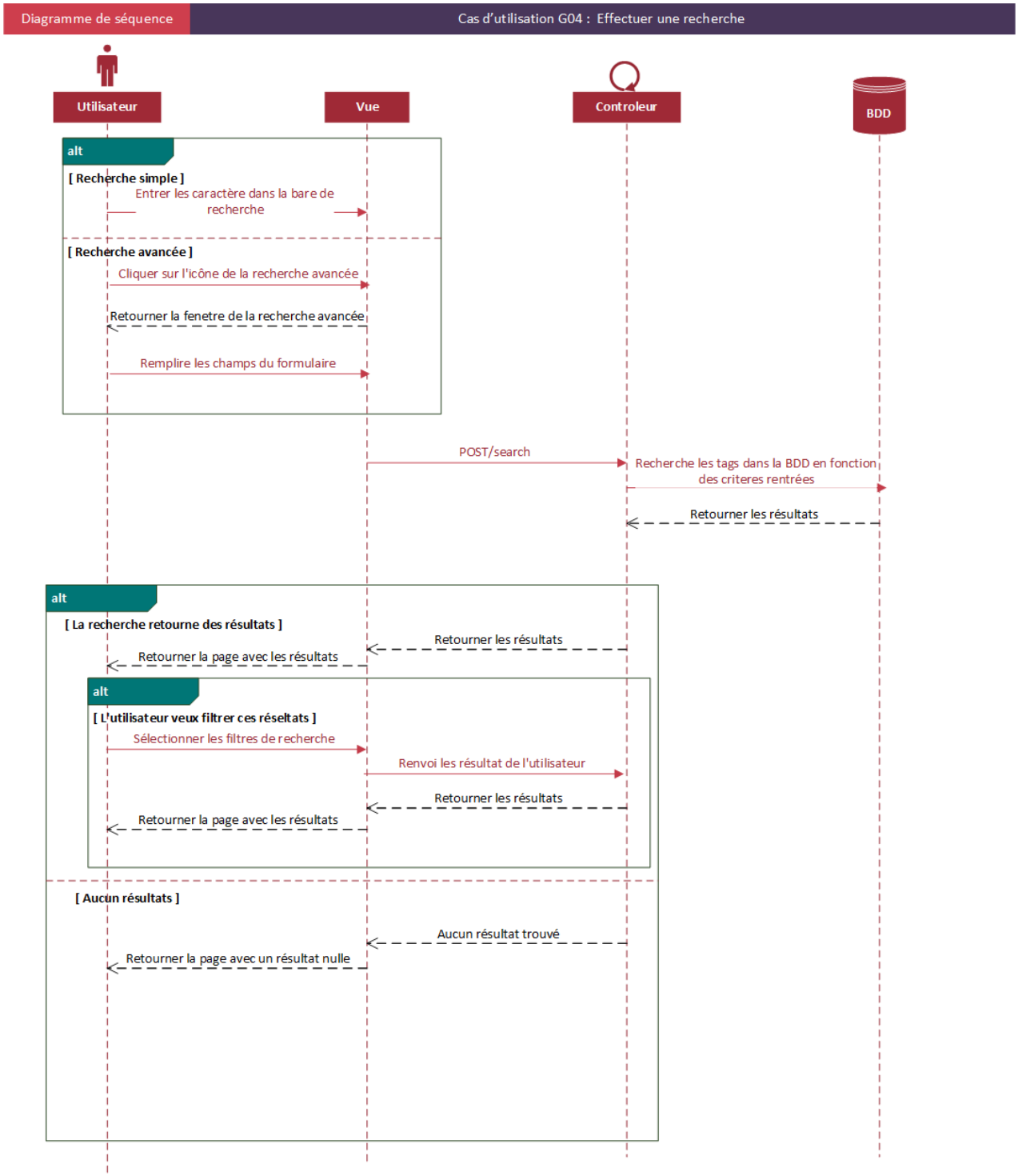

Figure 9 – Diagramme de séquence G04

Le diagramme ci-dessus va nous donner une vision de l’échange qui va être effectué à l'issue d’une recherche. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application.

L'utilisateur lance une recherche via un champ texte, le système retourne les résultats en temps réel dès que l'utilisateur envoi le texte à rechercher le système va rediriger ce dernier vers une page contenant tous les résultats trouvant sur la Base De Donnée ainsi que la possibilité de filtrer ces résultats.

7.5 Cas d'utilisation G05 : Laisser une carte de visite

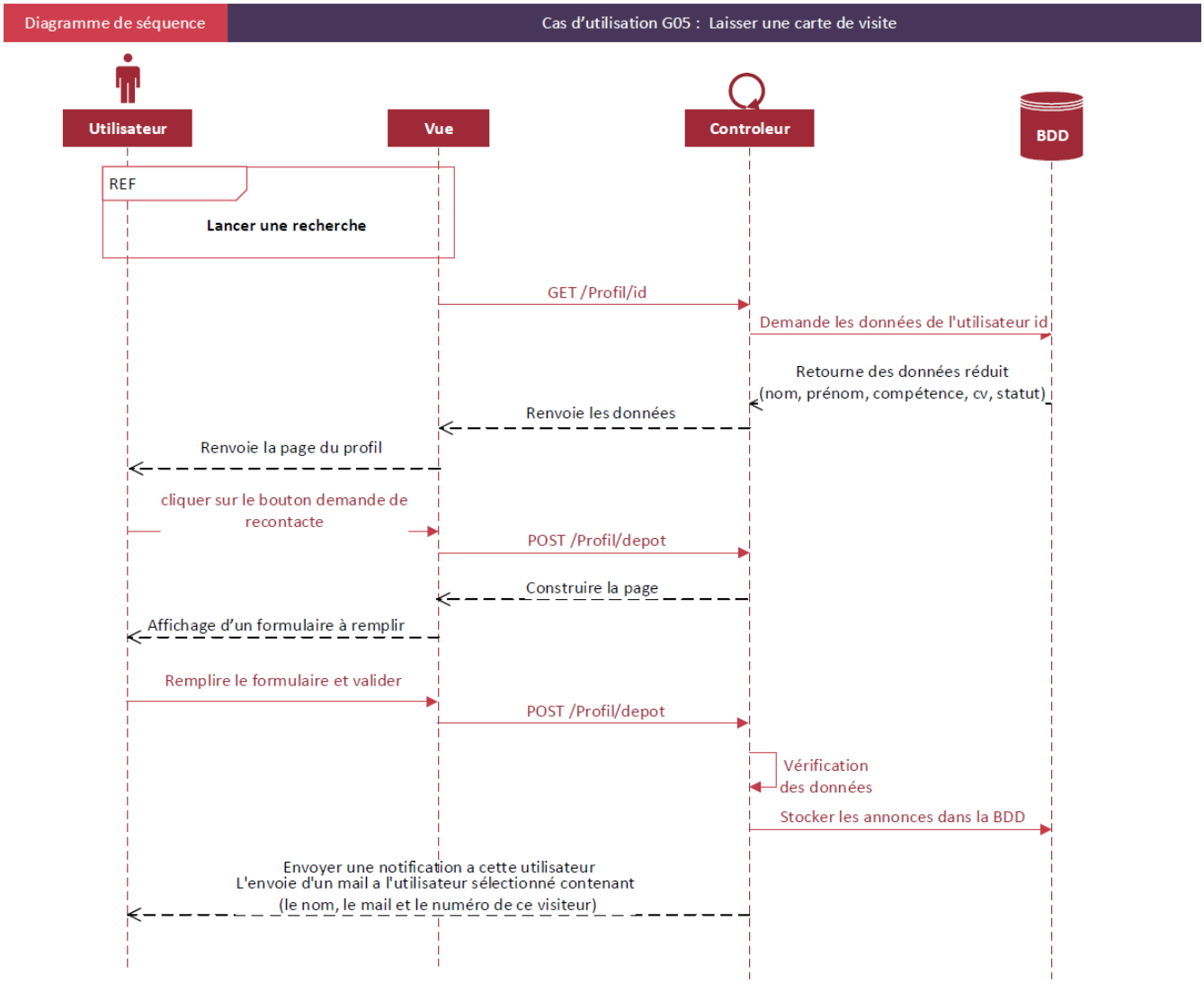

Figure 10 – Diagramme de séquence G05

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'un dépôt de carte de visite. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application.

L'utilisateur lance une recherche afin de trouver un membre précis par exemple après, il aura juste à cliquer sur demande de recontacte pour que le système le redirige vers un formulaire de saisie afin qu'il rentre ces données, dès que ce dernier clique sur valider le système va procéder à un test de validité de ces données les stocke et envoi un mail à l'utilisateur contacté avec les données saisîtes par l'acteur.

7.6 Cas d'utilisation P01 : Gérer un espace de publication

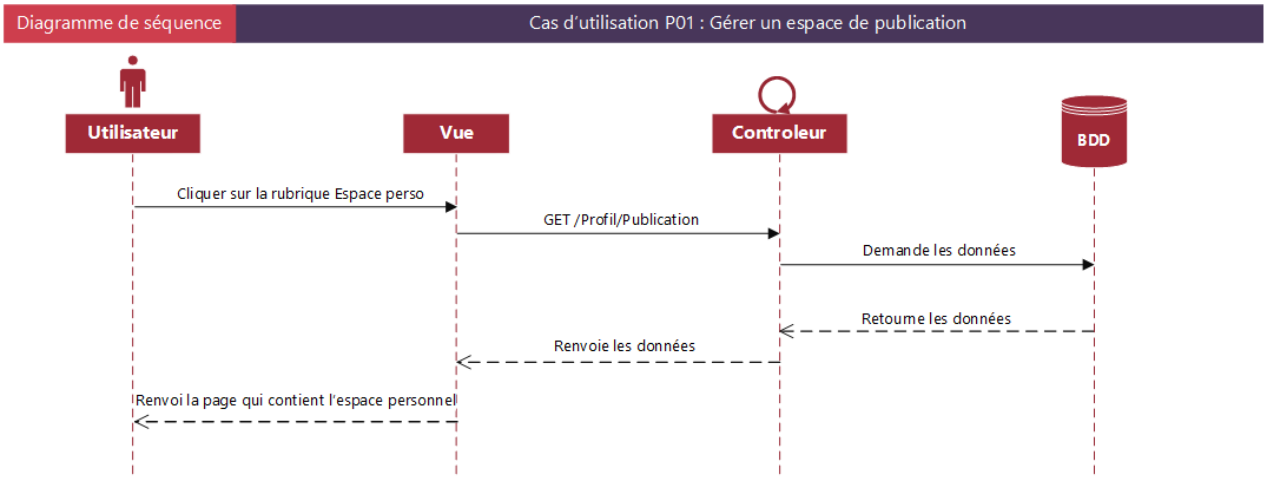

Figure 11 – Diagramme de séquence P01

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'une demande de gestion de l'espace de publication. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application sauf le Visiteur.

L'utilisateur aura juste à cliquer sur Rubrique Espace Perso pour que le système lui donne accès à la gestion de son espace de publication personnel.

7.7 Cas d'utilisation G02 : Consulter les sujets 'public' du forum

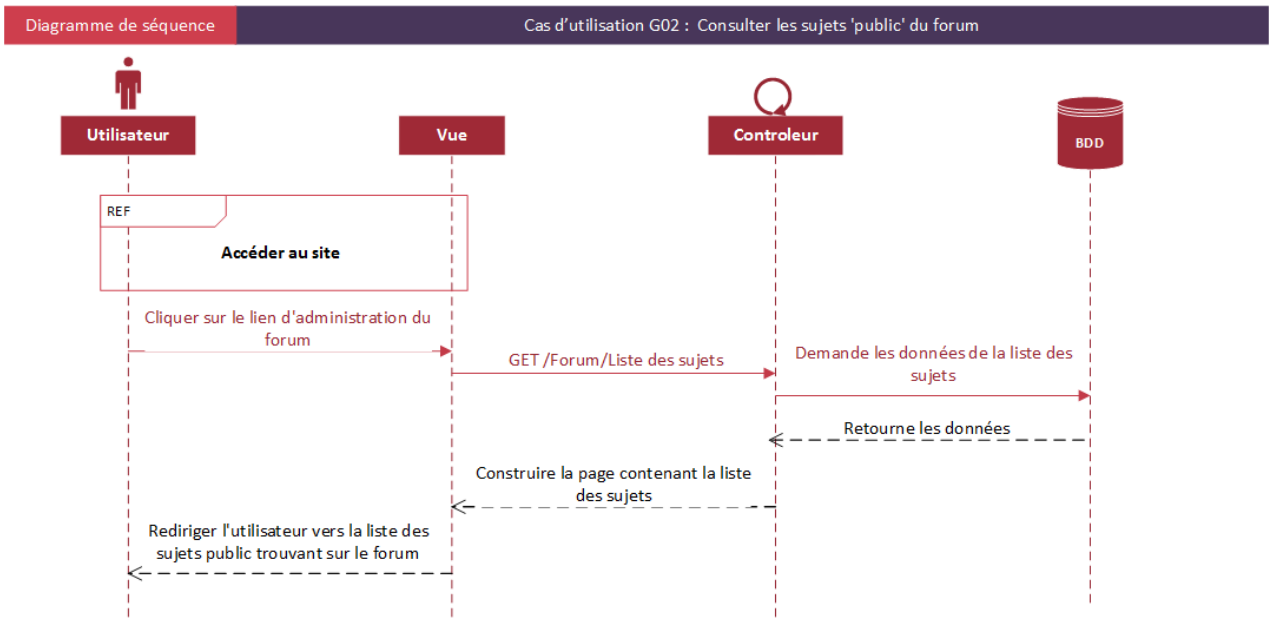

Figure 12 – Diagramme de séquence G02

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'une demande de consultation des sujets publics trouvant sur le forum. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application.

L'utilisateur aura juste à cliquer sur le lien d'administration du forum pour que le système lui donne l'accès à l'ensemble des sujets publics trouvant sur le forum où il pourra les commenter ou laisser des j'aime ou juste les consulté.

7.8 Cas d'utilisation P03 : S'abonner/suivre un membre

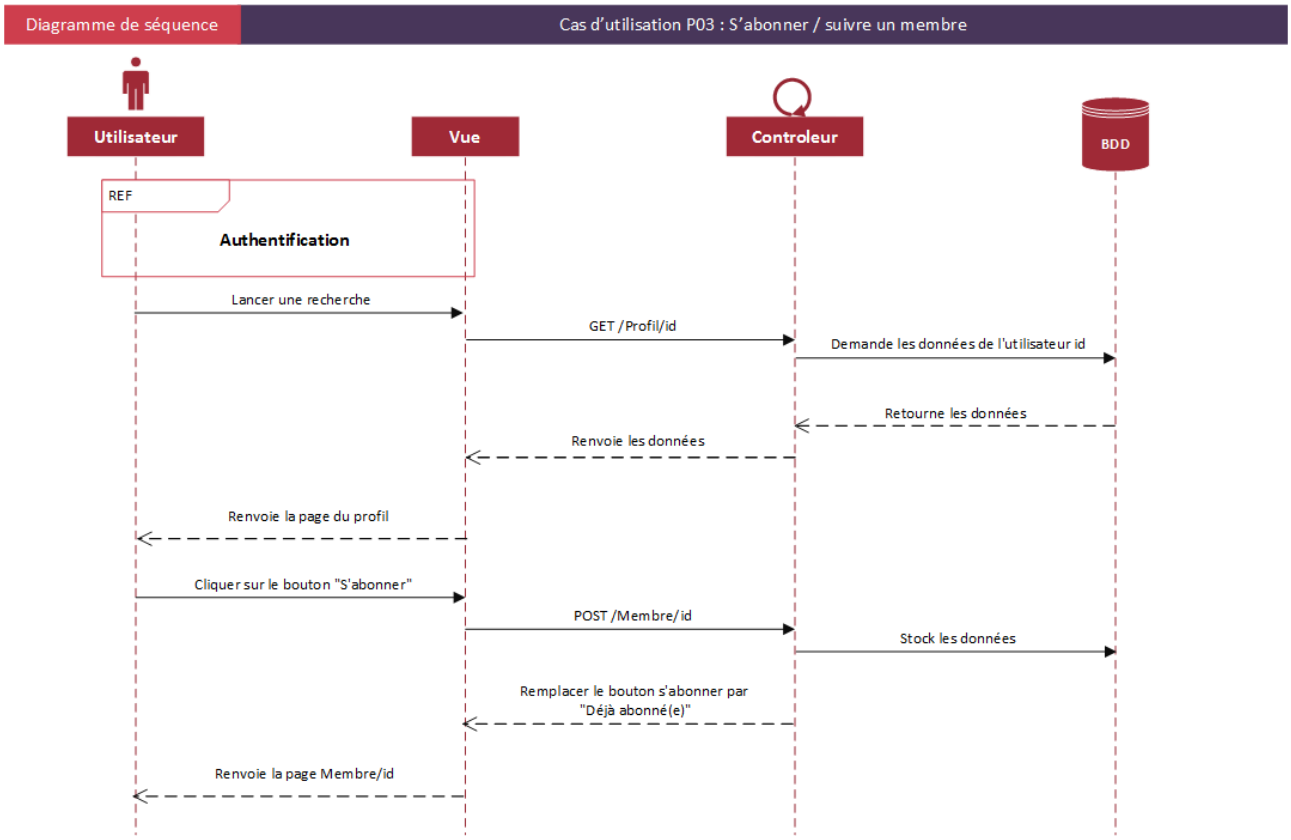

Figure 13 – Diagramme de séquence P03

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'une demande d'abonnement à un utilisateur donnés. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application sauf le Visiteur.

L'utilisateur aura juste à rentrer dans le profil de l'utilisateur voulu et cliquer sûr s'abonner pour suivre les actualisées de cet utilisateur.

7.9 Cas d'utilisation F01 : Créer un sujet sur le forum

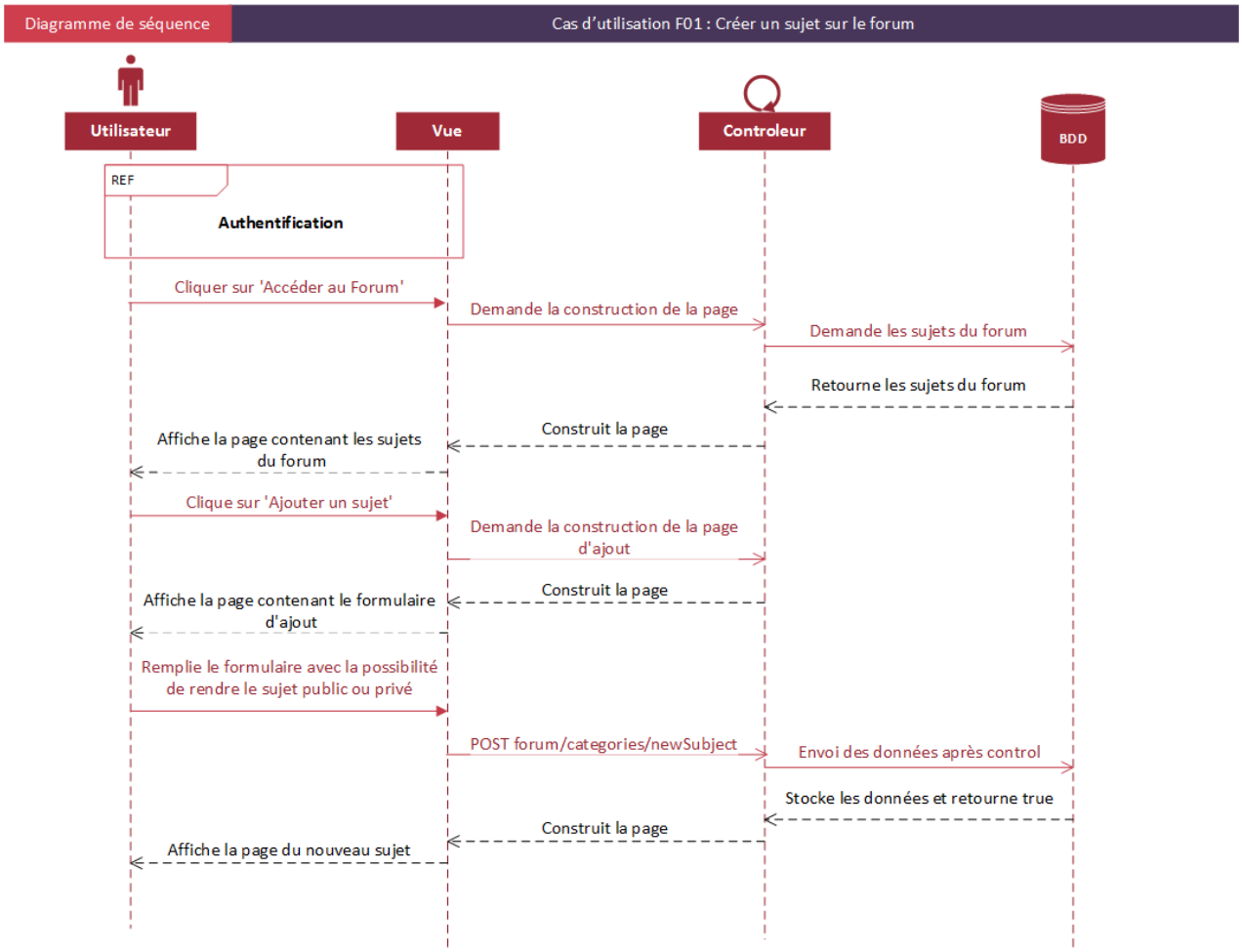

Figure 14 – Diagramme de séquence F01

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'une demande de création d'un sujet donnés. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application sauf le Visiteur.

L'utilisateur aura juste à accéder au Forum et de cliquer sur Ajouter un Sujet pour que le système le redirige vers un formulaire de saisie afin qu'il rentre les donnée nécessaire à la création de ce sujet, dès que ce dernier clique sûr valider, le système va procéder à un test de validité de ces données les stocke et redirige l'utilisateur vers la page du nouveau sujet.

7.10 Cas d'utilisation F02 : Signaler un message / sujet de forum

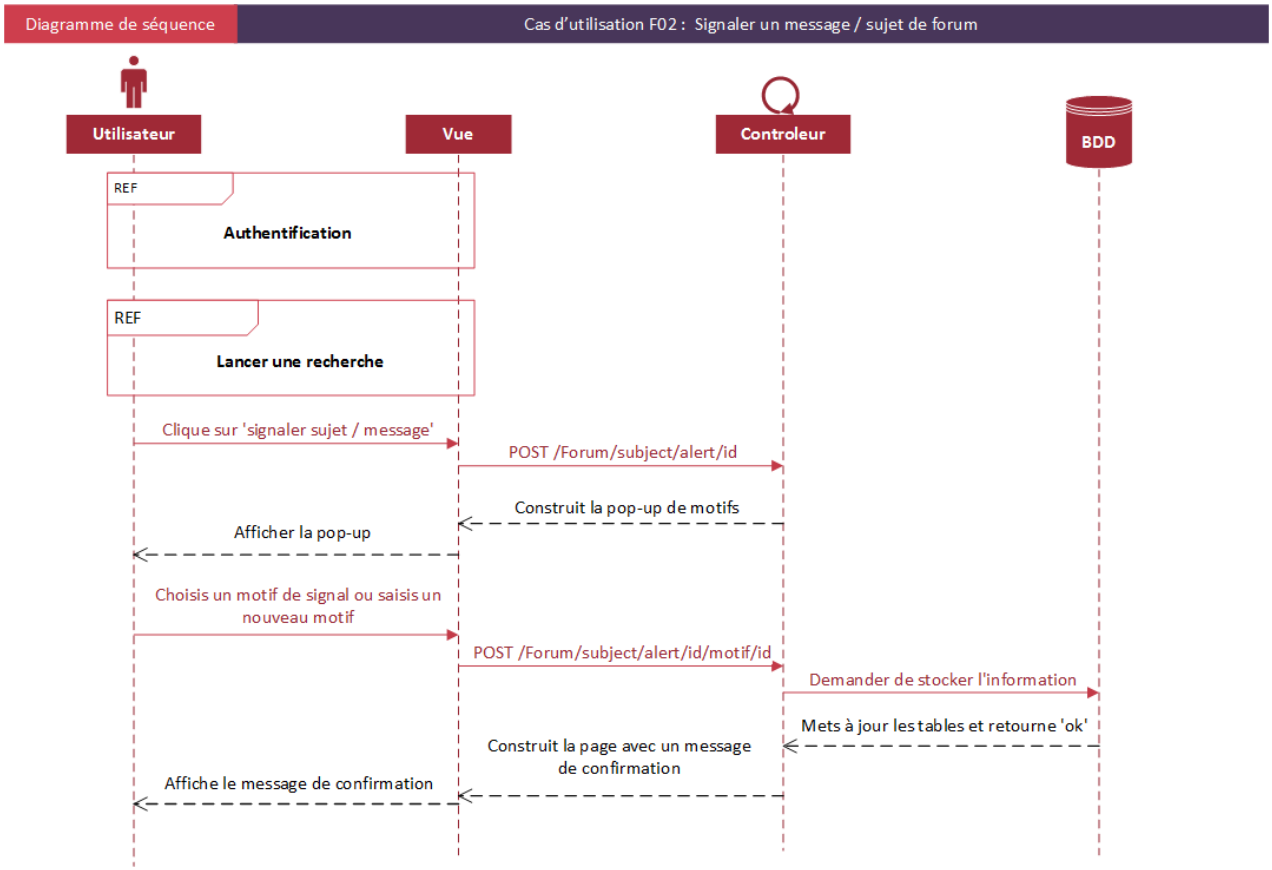

Figure 15 – Diagramme de séquence F02

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'un Signalement d'un message. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application sauf le Visiteur.

L'utilisateur aura juste à survoler le message qui veut signaler et cliquer sûr signaler pour que le système le redirige vers une boite de dialogue qui va lui permettre de renseigner le motif de ce signalement dès que l'utilisateur renseigne son motif le system envoi une notification au Modérateur et affiche un message de confirmation.

7.11 Casd'utilisationML01:Abonnement à un fil d'information

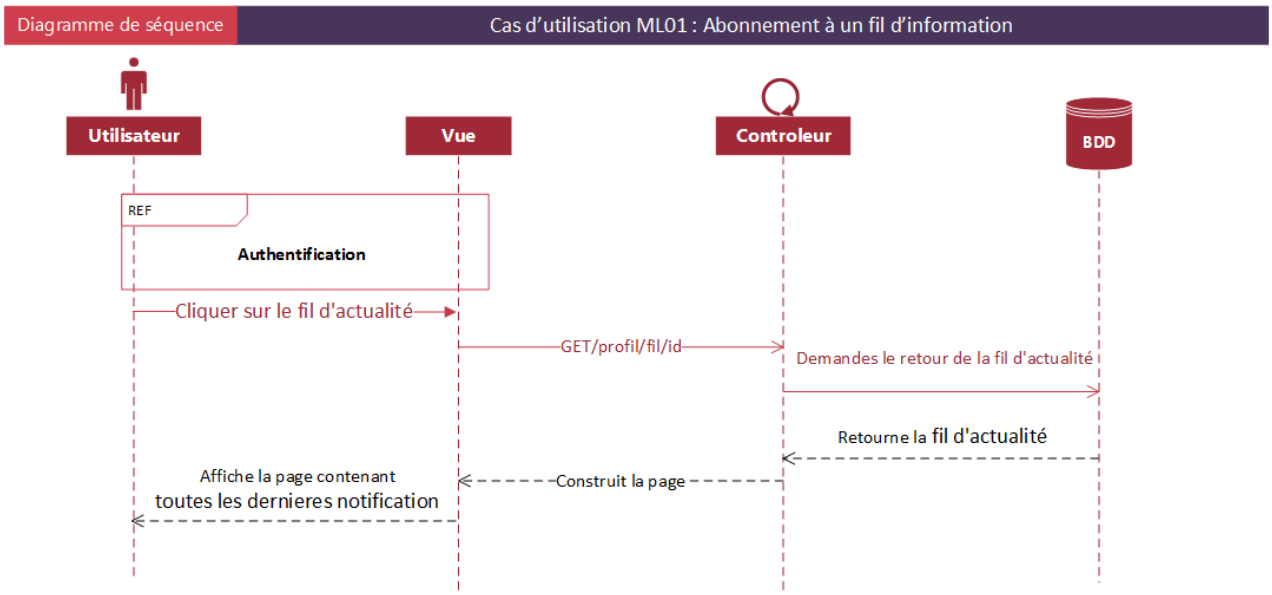

Figure 16 – Diagramme de séquence ML01

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'un Abonnement à un fil d'information. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application sauf le Visiteur.

L'utilisateur aura accès au fil d'actualité contenant toutes les dernières notifications des utilisateurs suivis plus les informations trouvant sur la mailing liste.

7.12 Cas d'utilisation F03 : « Aimer / Ne pas aimer » un message d'un forum

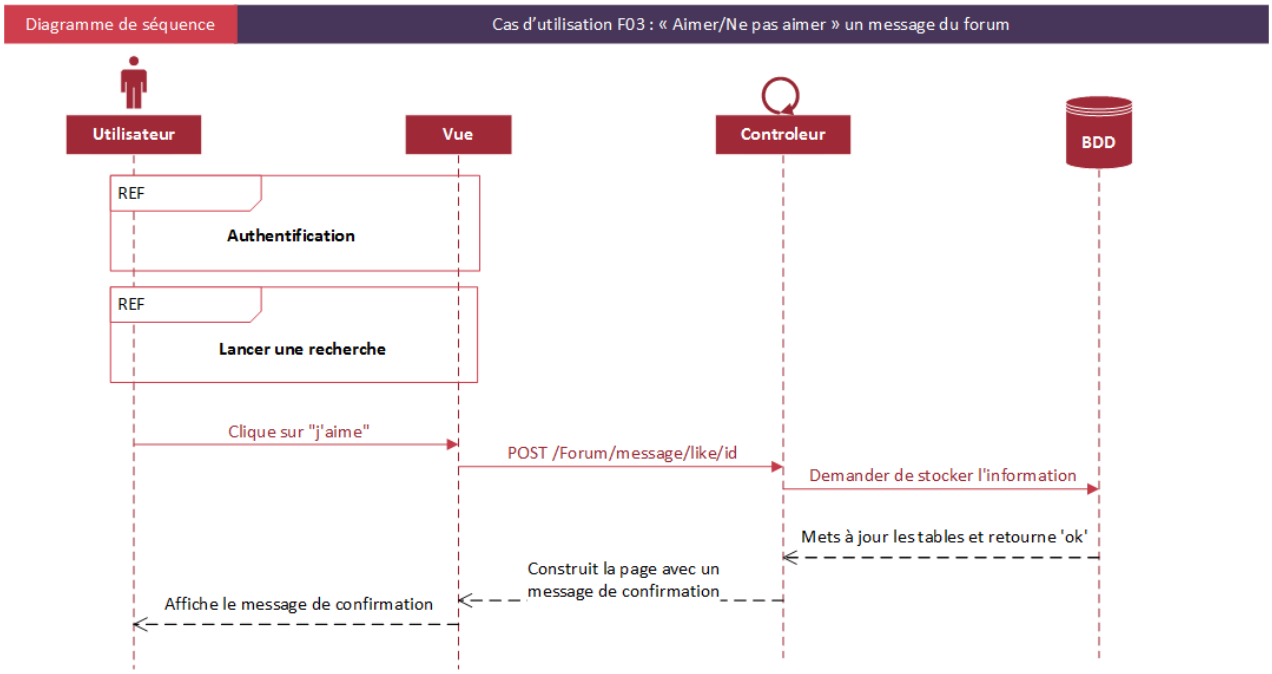

Figure 17 – Diagramme de séquence F03

Le diagramme ci-dessus va nous donner une vision de l'échange qui va être effectué à l'issue d'une expression d'un avis sur un message donné. Cette action peut être effectuée par tous les Acteurs pouvant accéder à l'application sauf le Visiteur.

L'utilisateur aura juste à survoler le message et cliquer sûr J'aime pour que le système enregistre cette information affiche un message de confirmation.