Compte rendu d’activité

Projet : Gestion des vacataires

Groupe 1

Bryce Fuertes

Groupe : BUT3 3B

Table des matières

Introduction

De nombreux vacataires sont présents au sein de l'IUT, leur nombre rend l'organisation compliquée pour l'enseignant responsable. A l'heure actuelle il n'existe pas de solutions spécifiques permettant de répondre à ce problème. Dans ce contexte une équipe d'étudiants de troisième année en formations initiale s'est lancée dans le projet de réaliser une application web de gestion des vacataires. J'interviens avec d'autres alternant en tant qu'équipe de renforcement dans ce projet de développement.

Lors de mon arrivée, l'application était fonctionnelle, la gestion des vacataires et des modules était développée mais l'application ne possédait pas de page d'authentification, de filtre pour les modules et les vacataires, ni de protection de l'API. Dans GitHub les issues relatives à ces taches n'étaient pas créées, la formalisation et la répartition de ces taches s'est faite à l'oral lors de notre arrivée. La documentation était assez réduite, ce qui a rendu la lecture du projet complexe. De plus le développement était réalisé dans un environnement de production et non un environnement de développement.

1. Mes taches

1.1. Création d'une page d'authentification

La tâche générale qui m'a été donnée était de mettre en place un système d'authentification, pour ce faire j'ai créé une branche « dev-auth » et j'ai ensuite créé une page d'authentification avec l'aide de Alex Jolas. Nous avons eu un dilemme quant au choix de l'authentification, soit utiliser un mot de passe hashé que nous stockons en base, soit passer par le CAS. Par manque de documentation nous avons décidé d'opter pour la première méthode qui est le mot de passe hashé en stocké en base.

Nous avons utilisé du Bootstrap qui était déjà utilisé dans le projet pour créer le formulaire d'authentification. Les connaissances d'Alex en Bootstrap nous ont permis d'avancer rapidement sur cette tâche. Nous avons ensuite dû stocker un mot de passe hashé dans notre base de données afin de vérifier le mot de passe lors de l'authentification. Pour ce faire nous avons créé une table « logins » puis nous avons ajouté un jeu de valeur contenant l'identifiant « admin » et le mot de passe hashé. Le hashage du mot de passe s'est fait avec l'outil bcrypt que nous avons d'ailleurs utilisé dans le code afin de comparer le mot de passe hashé au mot de passe en clair rentré par l'utilisateur.

Nous avons rencontré des problèmes du côté de l'API pour la vérification du mot de passe, celle-ci était comme je l'ai expliqué dans un environnement de production et ne possédait pas de branches de développement, nous étions donc obligés de commit et push nos modifications sur la branche master au risque de bloquer l'API.

Figure 1 : Formulaire d'authentification

Afin de faciliter la gestion de mon avancée j'ai créé des issues sur GitHub, celle correspondant à ces deux tâches est la suivante :

Figure 2: Issue de la création du formulaire d'authentification

1.2. Sécurisation de l'API

Afin de ne permettre qu'à un utilisateur identifié de faire des requêtes en utilisant certaines options, j'ai sécurisé l'accès en obligeant l'utilisateur à se connecter, pour se faire j'ai utilisé un token, que je stock en Stockage local et que je vérifie pour toutes les actions de l'utilisateur tel que la création ou la suppression d'un module ou d'un vacataire. Ce token a donc une durée de vie et si celle-ci est dépassée, lors de l'exécution d'une action l'utilisateur sera directement renvoyé vers la page d'authentification.

Afin de bloquer l'utilisateur ne possédant plus de token valide sur une action j'ai utilisé un « Interceptor ».

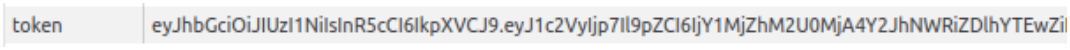

Figure 3 : Sotckage local du token

L'issue correspondante à cette tâche de sécurisation de l'API est la suivante :

Figure 4 : Issue de la sécurisation de l'API

1.3. Sécurisation du site web

Comme expliqué précédemment j'ai dû sécuriser l'API mais j'ai aussi dû sécuriser l'accès au site web, en effet les actions étaient bloquées mais les routes « vacataires » et « modules » étaient elles

toujours ouvertes. J'ai donc utilisé le même principe que pour les actions, un token et un « Interceptor » permettant de bloquer un utilisateur non identifié directement à l'accès du site web. Si celui-ci n'est pas identifié il sera donc directement renvoyé vers la page d'authentification.

De plus, afin de rendre plus simple l'authentification nous avons stocké la dernière page demandée par l'utilisateur afin de pouvoir le rediriger automatiquement vers la page demandée une fois son authentification réussie. La route de cette dernière page est stockée en stockage local comme le token. Pour cette partie j'ai pu être aidé par Michele Florio qui a pu me conseiller et me diriger vers une marche à suivre.

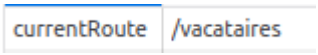

Figure 5 : Stockage en local de la dernière route sélectionnée

Conclusion

En conclusion, malgré les difficultés rencontrées lors de la prise en main et de la mise en place de ce projet, j'ai pu avancer rapidement et finir la tâche qui m'avait été confiée. Nous n'avons en effet pas pu utiliser la connexion CAS, cependant nous avons pu sécuriser correctement l'authentification. De plus, le site web et ses fonctionnalités sont aussi protégés par cette dernière.

Ma collaboration avec Alex Jolas et Michele Florio a été d'une grande aide car ils ont pu me faire part de leurs avis et me donner des idées, des pistes et des solutions pour certains de mes problèmes.

Malgré un état d'avancement satisfaisant pour le projet des améliorations futur peuvent être imaginées, tel qu'une gestion de profil utilisateur, permettant de changer le mot de passe sans être obligé de le faire directement dans la base de donnée.