DOCUMENTATION TECHNIQUE

1. Langages utilisés

Pour le projet Gamesoft, le choix des différents langages à utiliser dépendait, dans mon cas, de deux facteurs essentiels :

- Mes connaissances et affinités sur les dits-langages
- La pertinence des langages en fonction de la partie projet à effectuer.

En ce qui concerne mes connaissances, je suis assez familier des langages HTML, CSS et JavaScript, que j'ai non seulement étudié à la fois dans le cadre de ma formation chez STUDI, mais également auparavant, à travers quelques expériences sur une autre plateforme de formation ou des projets personnels. Dans le cadre de ces derniers, j'ai également exploré le langage python, mais aussi quelques autres langages comme le Basic, ou encore le C#, pour la programmation de microcontrôleurs comme l'Arduino. En revanche, ce n'est que très récemment que j'ai commencé à apprendre les framework : Bootstrap, node.js et Vue.js. C'est dans le cadre d'un projet personnel ayant vocation à devenir un projet professionnel (création d'entreprise), que j'ai pu découvrir le principe de composants avec Vue.js.

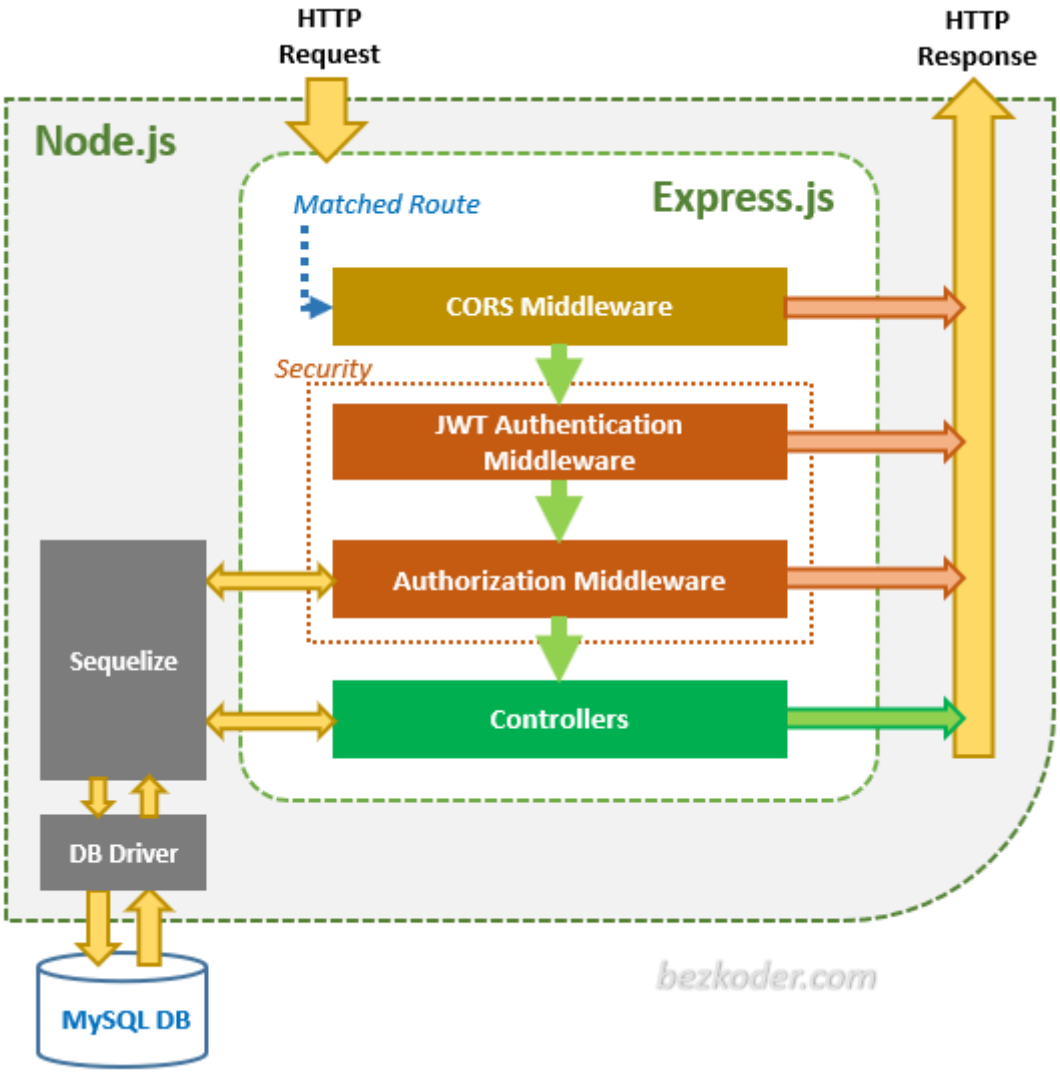
Vue.js est un framework qui m'a paru pertinent pour son système de composants dynamiques, sa simplicité et sa rapidité d'apprentissage, et sa modularité dans le cadre de la réutilisation des composants sur plusieurs parties du site. N'ayant pas encore suffisamment de connaissance en REACT.js, C'est donc naturellement sur Vue.js que mon choix s'est porté pour la création de la partie front-end du site. D'autre part, et d'après les informations que j'ai pu trouver, Vue.js possède nativement certaines protections contre des failles connues, notamment concernant les injections.

Pour gagner du temps dans la gestion du CSS, j'ai opté pour l'implémentation de Bootstrap. Ce framework m'a été très utile pour la construction des différentes pages, les choix esthétiques et la facilité d'intégration de pages et menus responsives, tout comme les systèmes d'accordions ou de carrousels.

Pour la partie back-end, ayant déjà été quelque peu familiarisé avec Node.js, ce framework, notamment avec Express.js et la gestion des CORS, m'est apparu pertinent. La mise en place d'un serveur, bien séparé du front, et la gestion des appels à la base de données via un middleware m'a permis de bien séparer la partie back du front, et ainsi gagner en lisibilité et en flexibilité.

Enfin, en ce qui concerne la base de données, j'ai décidé d'utiliser mySQL, avec lequel je suis également plus habitué, et dont j'ai pu optimiser et simplifier la gestion via un ORM : Sequelize.

Voici un schéma représentant le fonctionnement du back-end (emprunté sur le site www.bezkoder.com) :



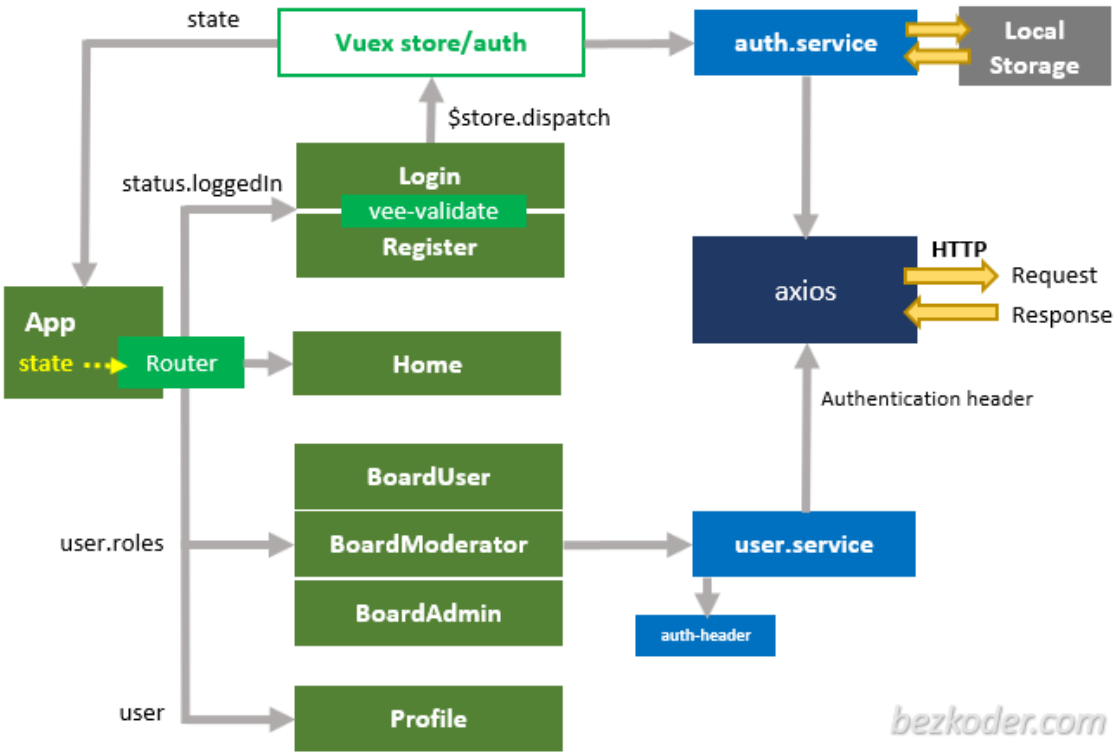
Tous ces langages m'ont permis d'adopter une structure de code modulaire et bien délimitée. En effet, avoir un front séparé du back, utiliser également un ORM pour la gestion de la base de données, permet de mon point de vue, de pouvoir en cas de besoin, changer chacune des parties sans avoir à tout recoder : Le front envoie des requêtes API au middleware dans un format spécifique qui n'a pas ou peu besoin d'être changé, dès lors que la refonte du middleware respecte les règles établies des appels API. De même que si l'on souhaite changer complètement le front, nous n'avons pas besoin de recoder le middleware : il suffit, même en changeant de framework pour le front, de respecter la même structure d'appels. D'autre parts, si la base de données mySQL n'est plus adaptée au projet, il est très facile d'en changer en gardant le même ORM Sequelize qui est compatible avec plusieurs autres architectures de base de données.

La recherche de la flexibilité a été un point important dans ma réflexion sur le développement du site.

Pour la sécurité du site, j'ai décidé de mettre en place un système de cryptage de mot de passe, en utilisant bcrypt. Il était important selon de ne jamais avoir de mots de passe en clair figurants dans la base de données, car si celle-ci est attaquée, ce sont tous les mots de passe associés à des comptes e-mail qui sont susceptibles d'être dérobés.

J'ai également mis en place dans le middleware un système d'API pour recevoir les requêtes du front avec Rôles utilisateurs afin de contrôler l'accès aux pages en fonction des utilisateurs du site (Administrateur, Producteur, Community Manager et Utilisateur standard). Pour cela, une table « roles » a été crée dans la base de données de manière à pouvoir attribuer un rôle à chaque utilisateur enregistré. Ce même rôle est alors utilisé en front via le local storage pour vérifier que l'utilisateur ne puisse pas se retrouver sur une page dans laquelle il n'est pas autorisé à y accéder.

Ci-dessous, un schéma du front, également emprunté au site www.bezkoder.com :



À noter, une petite différence dans l'utilisation du modèle de validation de données : J'ai remplacé vee-validate par yup, que j'ai mieux réussi à appréhender. En effet, du côté front, lors de l'envoie de formulaire, les données sont préalablement validé par yup, et seulement si ces données sont valides, alors elles sont envoyées au back-end pour traitement.

2. Environnement de travail

L'IDE sur lequel je me suis familiarisé avec le temps, est VSCode. Cependant, pour le projet Gamesoft, j'ai décidé d'expérimenter VSCodium, qui est une distribution open source libre de VSCode, mais qui ne possède pas de système de télémétrie Microsoft. L'utilisation est exactement la même, ainsi que les fonctionnalités proposées.

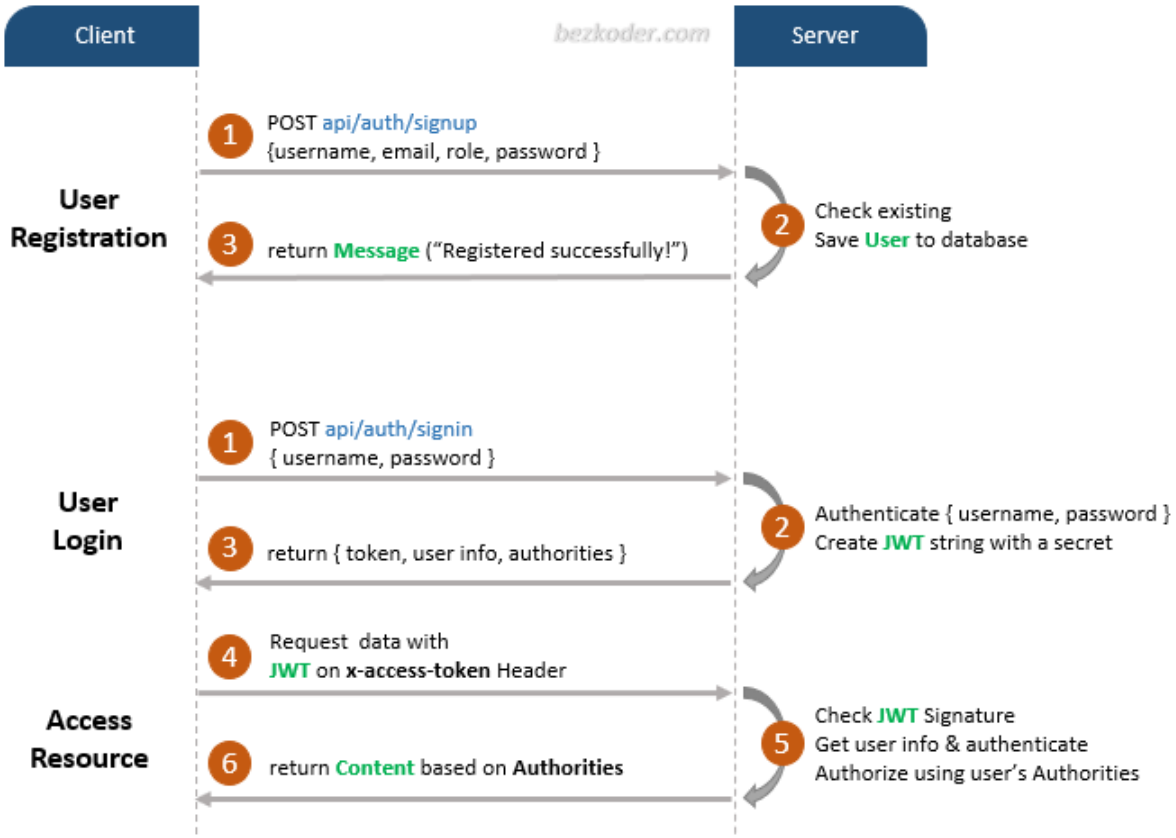
L'ordinateur à partir duquel je code est un PC sous système d'exploitation Windows 10, qui est plus difficile à paramétrer que Linux, mais n'ayant pas de machine sous Linux à disposition, j'ai cherché comment le configurer pour le code (installation de powershell… etc).

Pour les tests, essentiellement du back-end, j'ai utilisé Postman, de manière à vérifier que les schémas des requêtes étaient correctes, et que les réponses du back-end étaient valides. Du côté front, l'utilisation de « Vue ui » m'a été d'une grande d'aide pour vérifier le bon fonctionnement des pages et des composants.

En revanche, je me suis heurté à de grosse difficultés pour le déploiement sur le serveur distant Heroku : Je ne suis pas parvenu à faire fonctionner le site, à cause de la structure choisie. Je n'ai pas encore trouvé comment faire en sorte que le serveur d'Heroku affiche bien le front. De plus, en souhaitant utiliser une base de données SQL sur le serveur, j'ai également des difficultés à configurer mon back pour pouvoir communiquer avec celui-ci. J'ai tenté de modifier le fichier package.json de mon application afin de configurer le serveur, mais sans succès. J'espère encore pouvoir y parvenir prochainement, mes recherches continuent dans ce domaine…

3. Diagrammes

Voici un diagramme décrivant la séquence d'inscription et de connexion d'un utilisateur sur le site :



Voici un diagramme d'utilisation :