Projet d'application web
Site du club Japan7

AUTEURS

Ragot Cyrian
Klein Timothée
Rottier Nino
El Omari Marwa
Arrix-Pouget Baptiste

2025-06-09

Table des matières

1 Introduction

Pour ce projet d'application web, nous avons choisi de refaire le site du club de l'ENSEEIHT Japan7. [1] Nous avons globalement repris les mêmes fonctionnalités de l'ancien site et nous avons ajouté quelques autres. Le projet peut être trouvé à l'adresse donné par [2].

2 Spécifications

Les posts

Un post (entitée Post) possède un titre, une date de publication, une description, un auteur (user) et une liste d'évènements liés. Les admins peuvent créer (bouton + dans la barre de navigation), modifier et supprimer les évènements (en cliquant sur le post). Les posts sont affichés sur la page principale et l'utilisateur peut cliquer dessus pour afficher le post en entier.

Les évènements

Un évènement (entitée Event) possède un nom, une date, une description, un type et un nombre de likes. Les admins peuvent créer (bouton + dans la barre de navigation), modifier et supprimer les évènements (en cliquant sur l'évènement dans l'agenda). Les évènements sont aussi affichés sur les pages correspondantes à leur type (un évènement cuisine s'affiche sur la page cuisine).

Authentification et rôles

L'authentification est gérée par des JWT (Json Web Token) et les autorisations sont faites avec des rôles (entitée Role). Le rôle EXTERN est donné par défaut à un utilisateur qui créée un compte, il ne peut alors pas accéder aux ressources de notre application. Si l'utilisateur est donné le rôle MEMBRE, il peut alors accéder à toutes les informations du site. Il y a aussi des rôles ADMIN différents qui peuvent modifier certaines pages, par exemple le rôle LESSON_ADMIN peut modifier les leçons de japonais de la page "cours". Lorsqu'un admin modifie le rôle d'un utilisateur, une websocket permet d'effectuer le changement directement du côté de l'utilisateur pour qu'il n'ait pas à se déconnecter pour voir le changement.

Barre de navigation

L'application possède une barre de navigation qui est fixée en haut du navigateur et contient un logo, des liens vers les principales pages et un bouton déroulant un sous-menu des pages liées à la gestion du compte. Si l'utilisateur n'est pas connecté, le sous-menu contient des liens pour s'inscrire ou se connecter, sinon il contient des liens vers les pages pour consulter son profile ou se déconnecter (et un lien vers la page d'administration du site s'il est administrateur). Si l'utilisateur est un administrateur et est connecté, sa barre de navigation contient aussi un autre sous-menu lui permettant de rapidement créer un nouveau post ou un nouvel évènement.

Pied de page
Le pied de page contient des liens vers le serveur Discord et le compte Instagram de Japan7. Il contient aussi des liens vers les mentions légales, des informations sur les potentiels cookies utilisés et sur une page de contacts.

Page "Accueil"
La page d'accueil affiche une description du club avec une image illustrative. Les derniers posts ajoutés sont affichés sur cette page si l'utilisateur est connecté et a le rôle MEMBRE.

Page "Cuisine"
Le club peut organiser des ateliers cuisine, cette page affiche seulement les prochains évènements cuisine. L'utilisateur doit être MEMBRE pour accéder à cette page.

Page "Karaoke"
Le club organise des karaokes, cette page affiche donc les prochains évènements karaoke mais aussi les musiques utilisées dans les karaokes avec leur titre et un lien pour retrouver la musique avec les lyrics (entité Song). L'utilisateur doit être MEMBRE pour accéder à cette page. Seulement les utilisateurs avec le rôle KARAOKE_ADMIN peuvent ajouter/supprimer/modifier une musique sur cette page.

Page "Cours"
Le club organise des sessions de cours de japonais, cette page affiche donc les prochains évènements de cours de japonais. L'utilisateur doit être MEMBRE pour accéder à cette page. Aussi, les différents cours sont affichés, un cours est constitué d'un titre, d'un lien vers le cours écrit, un lien vers une fiche de vocabulaire, un lien vers les exercices et un lien vers un point culture (entité Lesson). Seulement les utilisateurs avec le rôle LESSON_ADMIN peuvent ajouter/supprimer/modifier un cours sur cette page.

Page "Projection"
Le club peut organiser des projections d'animes (entitée Anime), cette page affiche les prochains évènements de type projection. Aussi, elle affiche les animes en cours de visionnage et animes déjà visionnés par le club avec des flèches permettant d'incrémenter le nombre d'épisodes vus. Pour obtenir toutes les informations relatives à un anime (illustration, nombre d'épisodes de la série, score, ...) nous nous connectons à Jikan, un API Rest ouverte [3].

Page "Photos"
Cette page affiche les photos souvenirs du club. L'utilisateur doit être MEMBRE pour accéder à cette page. Seulement les utilisateurs avec le rôle ADMIN peuvent ajouter/supprimer une photo sur cette page. Une photo est enregistrée comme entité File, qui contient un nom, un type (jpeg, txt,...) et la donnée en bytes correspondante.

Page ”Agenda”
Cette page affiche un calendrier de tous les évènements présents et passés avec un horaire et une couleur différente selon le type de l'évènement, en cliquant sur un évènement du calendrier, l'utilisateur est redirigé vers la page de l'évènement pour plus d'informations. L'utilisateur doit être MEMBRE pour accéder à cette page.

Page d'administration
Cette page affiche tous les utilisateurs avec leur id, pseudo, rôles et adresse mail. L'administrateur peut modifier les rôles de chacun des utilisateurs et peut supprimer le compte d'un utilisateur. L'utilisateur doit être ADMIN pour accéder à cette page.

3 Technologies utilisées

Dans notre projet, l'API se situe dans le sous projet spring-boot-api et le frontend dans le sous projet vue-app. [2]

3.1 Base de donnée

Nous avons choisi d'utiliser une base de donnée PostgreSQL qui s'intègre très bien avec Spring Boot et qui permet d'avoir un environnement plus proche d'une base de donnée de production et plus documentée, car HSQLDB, bien que léger, est plus rarement utilisé en production. De plus, PostgreSQL intègre des fonctionnalités plus avancées si besoin.

Nous utilisons docker compose afin de lancer une image PostgreSQL pour l'environnement de production ou de développement.

3.2 API Rest

L'API écrit en Java est compatible avec Java 17+ et utilise Spring Boot 3. La gestion des librairies est faite avec Gradle Wrapper.

Plusieurs librairies externes sont utilisées dans ce projet, voir le tableau 1 des dépendances et leur utilité.

Dépendance	Utilisation
`spring-boot-starter-data-jpa`	Accès aux bases de données via JPA (Hibernate)
`spring-boot-starter-web`	Création d'API REST et gestion des requêtes HTTP
`spring-boot-starter-validation`	Validation des entrées utilisateur avec Bean Validation
`spring-boot-starter-security`	Sécurité de l'application (authentification, autorisation)
`spring-boot-starter-websocket`	Support de la communication WebSocket pour les connexions en temps réel
`jjwt-api`, `jjwt-impl`, `jjwt-jackson`	Création, parsing et gestion des JSON Web Tokens (JWT)
`spring-boot-devtools`	Outils pour améliorer le développement (rechargement automatique, etc.)
`postgresql`	Pilote JDBC pour la base de données PostgreSQL
`spring-boot-starter-test`	Bibliothèque de test intégrée (JUnit, Mockito, etc.)
`junit-platform-launcher`	Lancement des tests JUnit
`datafaker`	Génération de données factices pour les tests ou le développement

TABLE 1 – Dépendances utilisées dans le projet Spring Boot et leur utilisation

3.3 Application frontend

Pour le frontend de notre application web, nous avons utilisé le framework Vue 3 qui, comme d'autres frameworks tels que Angular ou React, permet de créer des pages web dynamiques en HTML, CSS et Javascript. L'intérêt de Vue.js est sa courbe d'apprentissage courte et sa documentation/communauté conséquente comparé aux autres solutions.

La gestion des librairies est faite avec le gestionnaire de packages Javascript npm. De plus, nous utilisons Vite, un outil de développement rapide qui intègre un serveur NodeJS, il permet entre autre de recharger automatiquement et rapidement les pages lors du développement.

Plusieurs packages ont été utilisés dans l'application Vue, le tableau tableau 2 donne une liste exhaustive des packages utilisés et leur utilité.

Package	**Utilisation**
`fullcalendar/bootstrap5`, `fullcalendar/core`, `fullcalendar/daygrid`, `fullcalendar/interaction`, `fullcalendar/vue3`	Intégration de FullCalendar, un composant pour avoir un calendrier avec des évènements.
`stomp/stompjs`	Communication en temps réel avec WebSocket via le protocole STOMP
`axios`	Requêtes HTTP asynchrones vers l'API
`bootstrap`	Framework CSS pour la mise en page et le design réactif
`bootstrap-icons`	Icônes vectorielles compatibles avec Bootstrap
`jquery`	Bibliothèque JavaScript pour manipuler le DOM
`jwt-decode`	Décodage des JSON Web Tokens pour extraire les infos de l'utilisateur
`popper.js`	Positionnement d'éléments (tooltips, dropdowns), requis par Bootstrap
`sockjs-client`	Fallback WebSocket compatible navigateurs (utilisé avec STOMP)
`vee-validate`	Validation des formulaires dans Vue.js
`vue-router`	Système de routage pour Vue.js
`vuex`	Gestion centralisée de l'état de l'application dans des stores
`yup`	Schéma de validation pour les formulaires, utilisé avec VeeValidate

TABLE 2 – Packages utilisés dans l'application Vue.js et leur usage

4 Structure du projet

La figure 1 résume l'architecture du projet.

Le projet est séparé en deux dossiers qui correspondent chacun au front (vue-app) et au back (spring-boot-api).

En dehors de ces dossiers, nous avons un Makefile servant à lancer l'ensemble du projet, et au même niveau des fichiers de configuration.

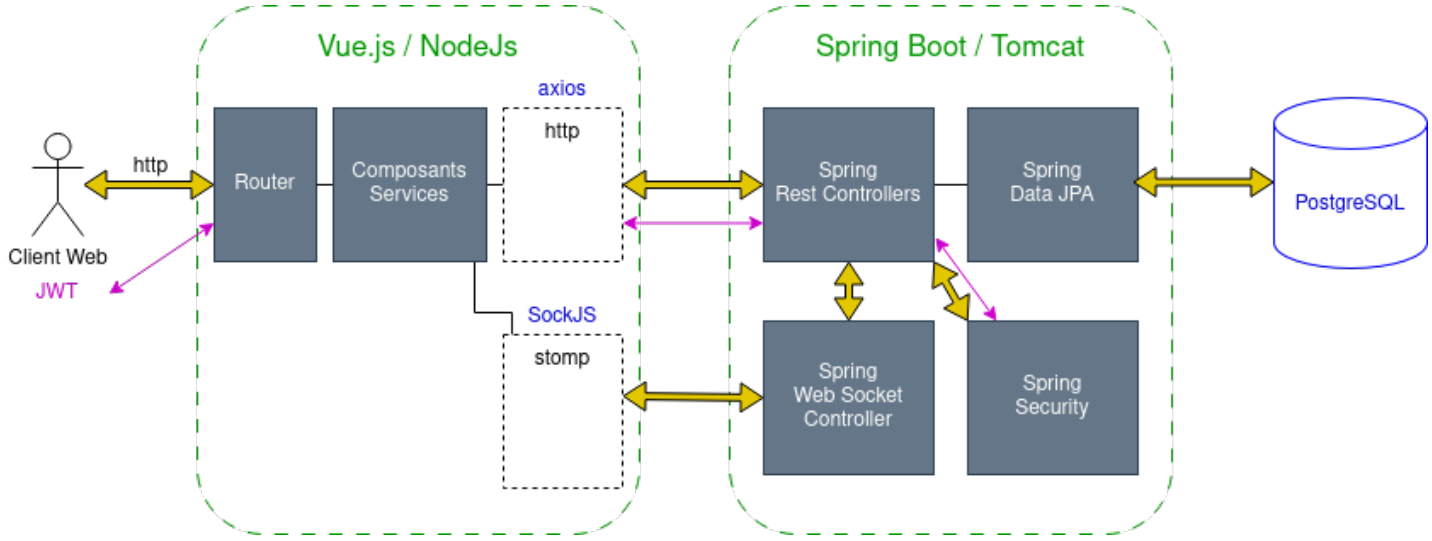

FIGURE 1 – Architecture du projet

4.1 Front-end

Au sein de ce projet, nous avons le fichier html principal index.html qui sert d'entrée : Vue insère les composants Vue compilés au sein de ce fichier html.

Dans le dossier src, nous avons :

- le composant Vue principal, `App.vue`,
- des fichiers javascript qui servent à configurer Vue, Vite, l'authentification/lien avec le back (http-common.js) et le routage des composants (router.js),
- un fichier css commun à toutes les pages du site,
- le dossier `components` qui contient en vrac tous les composants (secondaires) Vue,
- le dossier `assets` qui contient les images, svg, etc.,
- le dossier `services` qui contient des fichiers Javascript servant à simplifier les requêtes au back,
- le dossier `store` s'occupe des stores vuex (données dynamiques inter-composants).

4.2 Back-end

Dans ce projet, nous avons des fichiers de configuration de Gradle et Spring Boot, et le dossier src. Dans le package `fr.n7.spring_boot_api`, nous avons un dossier par type de fichier java :

— le dossier `controllers` contient un fichier java par Controller (nous en avons un pour chaque type de requête à notre serveur : EventController gère les requêtes concernant les évènements, FileController gère les fichiers...),

— le dossier `datasource` s'occupe du seeding de la base de données en dev ou en prod,

— le dossier `model` contient toutes les entités,

— le dossier `payload` contient des classes intermédiaires permettant d'alléger par exemple des entités pour ne transmettre que les données utiles,

— le dossier `repository` contient les repository des entités,

— le dossier `security` s'occupe de l'authentification

— le dossier `service` contient des classes auxiliaires

— le dossier `websocket` contient les fichiers Java s'occupant des WebSocket

5 Gestion de projet

Pour la gestion de projet, nous avons utilisé un backlog avec des issues GitHub, nous pouvions facilement nous repérer avec des listes "Todo", "In Progress" et "Done" pour savoir quelle étaient les tâches à faire et qui travaillait sur quelle tâche. Nous avions aussi un serveur discord avec différents salons pour communiquer en appels ou en messages.

Comme expliqué dans la partie sur les technologies, nous avons utilisé Gradle et npm pour gérer les dépendances des deux projets. Aussi, pour faciliter le développpement, nous utilisons un makefile qui contient plusieurs commandes telles que `make dev` qui lance les serveurs de développement en arrière-plan, `make stop` qui stoppe les serveurs de développement, `make clean` qui supprime les fichiers de build et les dépendances téléchargées. Un fichier nommé `CONTRIBUTING.md` indique comment développer sur le projet et comment utiliser ce makfile plus en détails.

Nous utilisons aussi des fichiers `.env.example` et `.env` pour régler des variables d'environnement qui doivent être différentes lors du déploiement en production ou d'un simple environnement de développement.

6 Conclusion

Beaucoup d'améliorations sont envisageables, le site est quasiment déployable en l'état mais nécessiterait un peu plus de temps pour le peaufiner. Cependant, nous avons réussi à mettre en place toutes les fonctionnalités prévues, parfois même plus.

Les pistes d'amélioration seraient d'utiliser un vrai système de stockage pour les fichiers/images et non pas les enregistrer telles quel dans la base de données en format byte. Aussi, certains endpoints de l'api ne sont certainement pas très optimisés et renvoient trop de données d'un coup, ou alors ne sont pas bien scalable avec beaucoup d'utilisateurs.

Table des figures

Liste des tableaux

Références

[1] Japan7 Club, “Japan7 – club de culture japonaise de l’ENSEEIHT.” https://japan7.bde.enseeiht.fr/, 2025. Consulté le 30 mai 2025.

[2] C. R. . Co., “App-web-n7 – projet d’application web de l’enseeiht.” https://github.com/cyrianR/app-web-n7, 2025. GitHub repository, consulté le 30 mai 2025.

[3] J. API, “Jikan - unofficial myanimelist api.” https://jikan.moe/, 2025. Accessed : 2025-06-04.