



TPI – Canap-Gest

« CANAP-GEST » APPLICATION WEB DE GESTION DE
CANDIDATURES D'APPRENTISSAGE EPFL

EPFL | NICOLAS CRAUSAZ

CHEF DE PROJET | NICOLAS DUBOIS
EXPERTS | B. FOLOMIETOW, F. ANDOLFATTO

Sommaire

SOMMAIRE	1
1 ANALYSE PRÉLIMINAIRE	2
1.1 INTRODUCTION.....	2
1.2 SITUATION ACTUELLE	2
1.3 OBJECTIF	4
1.4 TRAVAIL PRÉALABLE.....	4
2 ANALYSE DU PROJET	6
2.1 POINTS IMPORTANTS DU CAHIER DES CHARGES	6
2.2 FAISABILITÉ DU PROJET	9
2.3 CHOIX DES TECHNOLOGIES.....	10
2.4 PLANIFICATION INITIALE	10
2.5 MATÉRIEL ET RESSOURCES À DISPOSITION	11
2.6 VERSIONNING ET SAUVEGARDES	15
3 RÉALISATION	17
3.1 MODÈLE DB	17
3.1.1 MCD	17
3.1.2 MODÈLE DB - MPD	18
3.2 STRUCTURE DE L'APPLICATION.....	19
3.3 AUTHENTIFICATION	20
3.3.1 TEQUILA	20
3.3.2 TOKEN	20
3.3.3 PROCESSUS	23
3.4 REQUÊTES API	24
3.5 INTERFACE	28
3.5.1 STRUCTURE	28
3.5.2 FONCTIONNALITÉS	28
3.5.3 TECHNOLOGIES	35
3.5.4 DESIGN	37
3.5.5 PRATIQUES DE CODE	38
3.6 GESTION DES ERREURS.....	42
3.6.1 BACKEND	42
3.6.2 FRONTEND	44
3.7 TESTS.....	45
4 CONCLUSION	47
4.1 ATTEINTE DES OBJECTIFS	47
4.2 PLANIFICATION	47
4.3 AMÉLIORATIONS FUTURES	48
4.4 PROBLÈMES RENCONTRÉS	49
4.5 RESENTI PERSONNEL	52
5 GLOSSAIRE	53
6 ANNEXES	56
6.1 SOURCES	56
6.2 COMPTE-RENDU DES DISCUSSIONS.....	57
6.3 CODE SOURCE	58
6.4 AUTRES.....	58

1 Analyse préliminaire

1.1 Introduction

Il y a deux ans, j'ai développé un formulaire permettant la postulation en ligne des potentiels apprentis, pour la formation apprentis EPFL. Ceci visait à remplacer le formulaire papier envoyé par poste qui était jusqu'à présent utilisé.

Ce formulaire permet de saisir diverses informations (personnelles, scolaires, professionnelles) et de joindre des fichiers annexes (cv, lettre de motivation etc.).

Le système est stable et fonctionne bien, cependant son utilisation comporte encore quelques contraintes. L'idée du projet **Canap-Gest** naît donc du principe de l'amélioration de ce système.

1.2 Situation actuelle

La personne désirant postuler pour un apprentissage à l'EPFL se rend sur le formulaire, crée un compte au travers du service de compte invités de l'EPFL, puis remplit ses informations et joint ses fichiers annexes.

Les résultats des postulations sont stockés sur un partage réseau réservé à cet effet, dans un dossier « nouvelles postulations ».

Le partage est structuré en 4 dossiers :

- Nouvelles : Résultats des postulations
- Valides : Dossiers valides
- Exclues : Dossiers non valides, incomplets, etc.
- Engagées : Dossiers retenus et prochainement engagés

Les responsables de la formation apprentis EPFL s'occupent de procéder à la sélection des dossiers et de transmettre les dossiers (en les déplaçant entre les dossiers parents) aux formateurs qui choisiront leurs futurs apprentis.

Les résultats des postulations suivent une structure de dossier spécifique, triée premièrement par métier, le dossier du postulant contient ensuite les fichiers annexes et les informations, celles-ci sont stockées dans un fichier **JSON**.

Le diagramme suivant récapitule la stratégie actuelle :

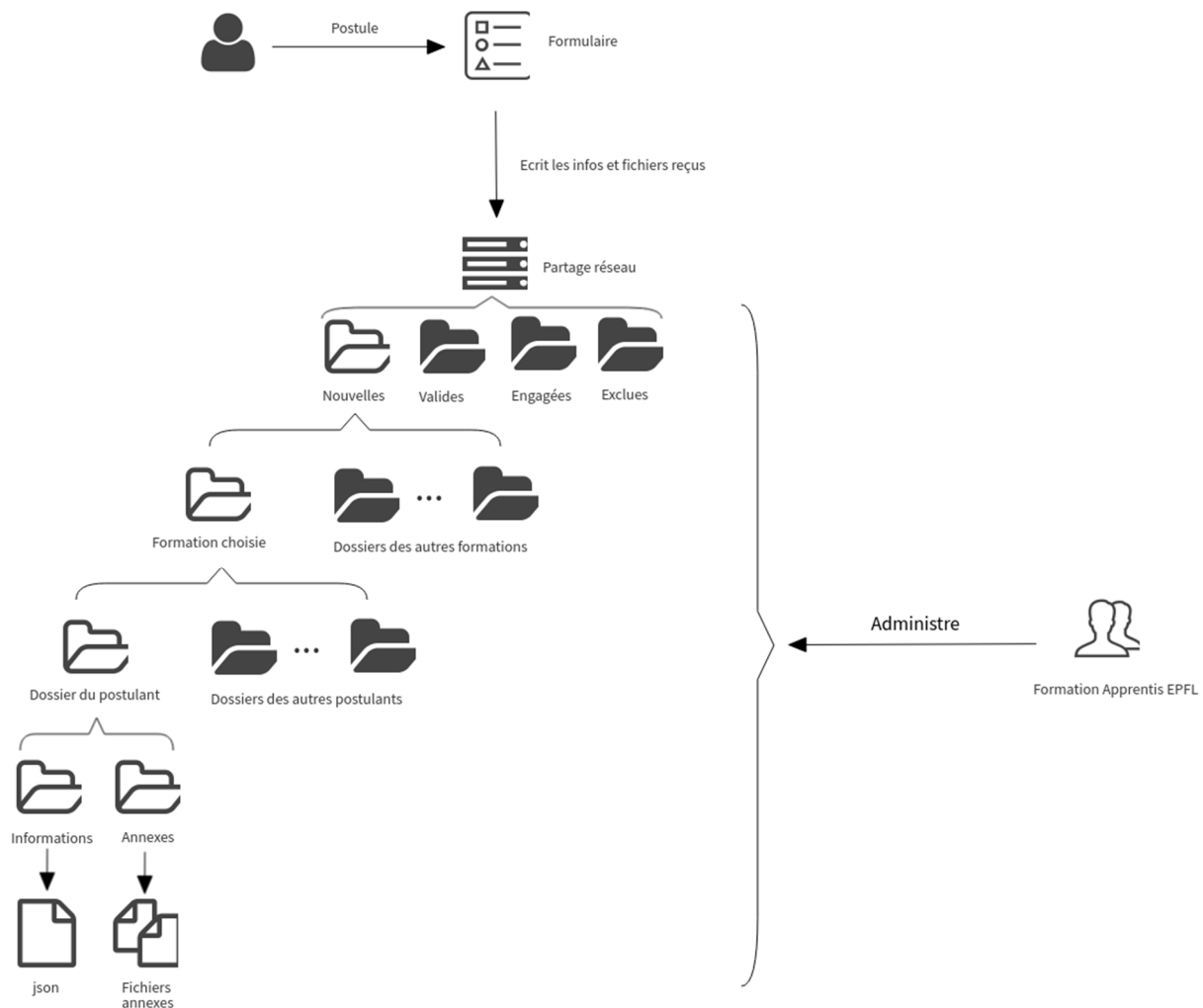


Figure 1: Fonctionnement actuel du système

Quelles sont donc les contraintes actuelles ?

- Les informations personnelles du postulant sont stockées dans un fichier JSON, il n'est donc pas facilement lisible sans utiliser quelconque outil pour faciliter son accès.
- Les responsables modifient directement les dossiers sur le partage, en cas de mauvaise manipulation (par exemple glisser-déposer au lieu de copier-coller) cela peut entraîner des problèmes (fichiers corrompus, permissions modifiées etc.).
- Nous ne disposons pas d'une vue globale sur les postulations.
- Nous ne disposons pas d'une manière de trier et comparer efficacement les postulations entre elles selon certains critères.

Des améliorations au niveau de la protection des données doivent également être appliquées, par exemple, dans l'état actuel, un candidat ne peut pas voir, modifier ou supprimer sa ou ses postulations effectuées, ce qui entraîne des conflits avec la loi **RGPD**.

1.3 Objectif

Le but principal serait donc de trouver des compromis pour éliminer les contraintes actuelles, il faut donc appliquer une solution permettant principalement de :

- Éviter la manipulation des fichiers de postulation directement sur le partage réseau.
- Améliorer l'accès et la facilité de traitement des données des candidats.
- Améliorer et faciliter le travail de sélection des candidatures valides par les responsables Formation Apprentis EPFL.
- Faciliter le travail de sélection et recrutement des postulants par les formateurs EPFL.

L'idée qui a été retenue pour obtenir ces résultats est de développer une interface de gestion centralisée, permettant aux responsables de la Formation Apprentis EPFL et des formateurs d'effectuer tout leur travail de tri/recrutement.

Pour ce faire, il a fallu avant tout réaliser les points suivants :

- Enregistrer les informations des candidatures (sans les fichiers) dans une base de données pour en faciliter l'accès ultérieur.
- Développer une interface web permettant l'administration des candidatures.
- Développer une API permettant l'accès à la base de données, faisant le lien entre l'interface et la base de données.

1.4 Travail préalable

Une partie du travail a été réalisée au préalable ; Premièrement j'ai créé et mis en production une base de données (**DB**), qui est hébergée sur un serveur MySQL de l'EPFL. Ensuite, le formulaire à évidemment dû être adapté pour pouvoir enregistrer les données en DB, et que le postulant puisse voir, éditer et supprimer ses postulations.

L'API pour l'accès aux données a également été développée en partie pour préparer l'implémentation des fonctionnalités de la future interface.

L'ajout de la base de données a déjà permis de combler la problématique du traitement des fichiers d'informations directement sur le partage réseau et d'en simplifier sa structure,

puisque que celui-ci ne sera utilisé que pour les fichiers annexes des candidatures et que l'accès direct au serveur ne sera plus utile.

Plusieurs discussions avec les responsables de la formation apprentis EPFL ont été entreprises, également un questionnaire a été envoyé aux formateurs EPFL afin de leur proposer des fonctionnalités et connaître leur avis sur l'idée du projet.

Le chef de projet s'est notamment basé sur les réponses de ce questionnaire pour établir les critères du cahier des charges du projet.

Le diagramme suivant résume la situation :

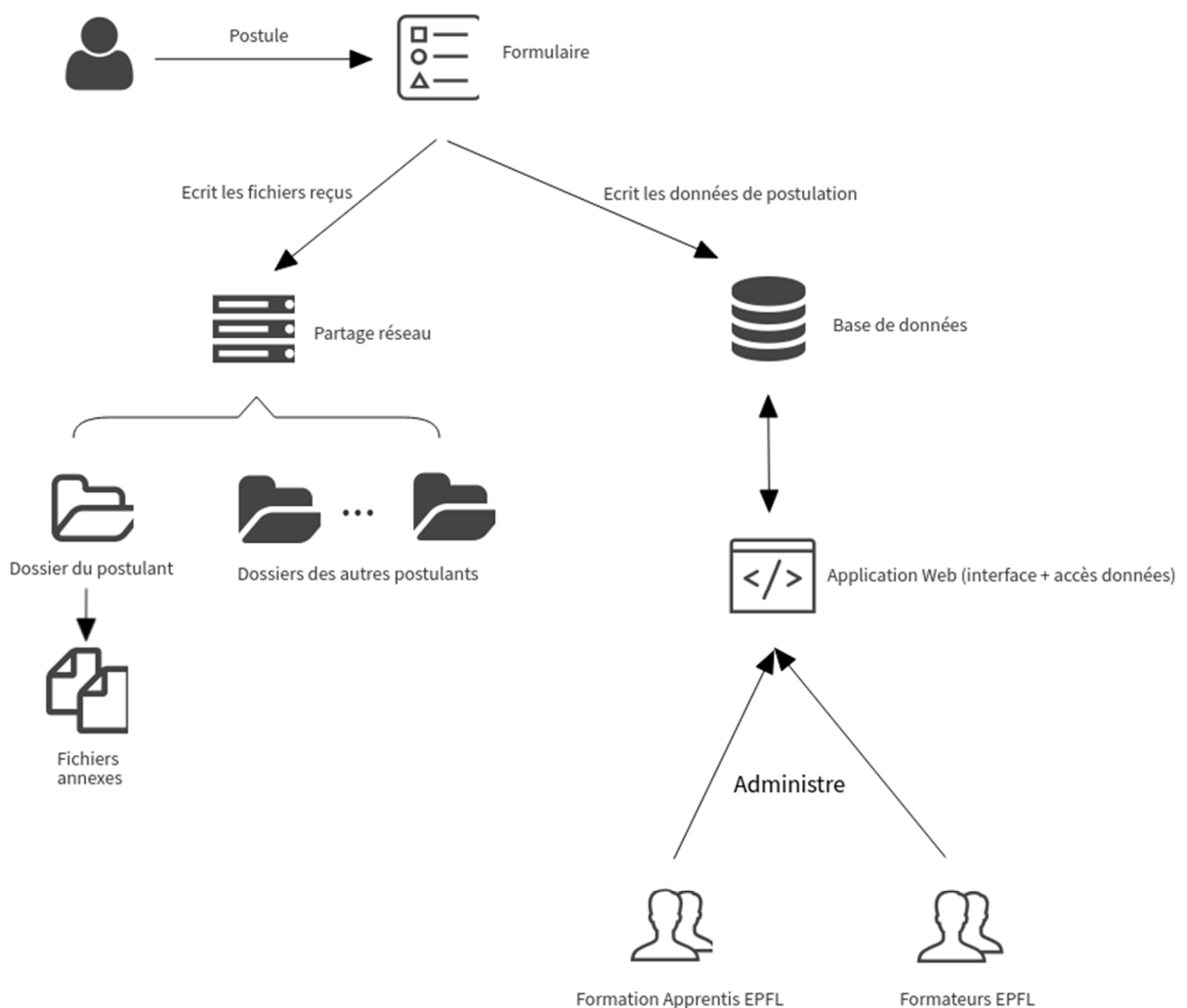


Figure 2: Fonctionnement du système attendu

2 Analyse du projet

2.1 Points importants du cahier des charges

Comme décrit auparavant, le projet consiste principalement à développer une application web offrant les caractéristiques listées ci-dessous, les fonctionnalités sont séparées entre les formateurs et les responsables.

Fonctionnalités pour les responsables

- **Gérer la liste des sites EPFL (lieux de formation)**

Il faudra ajouter une table « lieu » au modèle de la base de données actuel et d'ajouter des requêtes sur l'API pour l'ajout et l'édition des lieux.
L'interface permettra l'édition des lieux par les responsables.

- **Gérer la liste des professions et des places ouvertes**

Il faudra ici également ajouter plusieurs tables au modèle de la base de données actuel et d'ajouter des requêtes sur l'API pour l'ajout et l'édition des lieux.
Ces fonctionnalités sont communes à l'interface réalisée pour le point précédent.

- **Voir les candidatures, les valider, refuser ou les marquer comme incomplètes**

Au niveau de l'interface, il faudra ajouter une vue pour lister les postulations de manière globale et également de manière individuelle. Les vues devront avoir les fonctionnalités pour modifier le statut de la candidature.

Au niveau de l'API, les requêtes liées aux fonctionnalités ont déjà été implémentées, cependant un changement de la structure de la base de données, notamment l'ajout d'une table « statut » est envisageable, il faudra donc mettre à jour les fonctions de l'API.

- **Refuser des « vagues » de candidatures et supprimer leurs données**

La vue globale des candidatures devra offrir la possibilité de sélectionner plusieurs candidatures et de les supprimer en effectuant une seule action.

Lors d'un refus, le responsable contactera « manuellement » la personne concernée, puis ses données devront être totalement supprimées.

Fonctionnalités pour les formateurs

- **Lister les candidatures, globales et individuellement, voir les informations complètes et les fichiers annexes**

Les vues globales et individuelles devront s'adapter en fonction du rôle de l'utilisateur (responsable ou formateur) et afficher ou non des fonctionnalités. La vue globale contiendra les informations personnelles les plus importantes d'une candidature. La vue individuelle regroupera toutes les informations liées (responsables, parcours scolaire, fichiers annexes etc.).

- **Identifier les nouvelles candidatures**

Il faudra rendre identifiables les nouvelles candidatures, par exemple en affichant le nombre de nouvelles candidatures depuis la dernière connexion de l'utilisateur. L'ajout d'un marqueur dans la vue globale peut également être ajouté pour améliorer la visibilité de ces candidatures.

- **Attribuer des notes aux candidats**

Un formateur pourra attribuer des notes aux candidatures afin qu'il puisse organiser ses candidats favoris. Un système de notation, par exemple de 1 à 5, pourra être mis en place.

Pour cela il faudra ajouter un table « note » à la base de données et ajouter les fonctionnalités nécessaires à l'API (ajout, modification et suppression).

- **Ajouter des commentaires aux candidats, privé ou partagés avec les autres formateurs**

Un formateur pourra commenter une candidature et choisir de partager ce commentaire avec les autres formateurs disposant de l'accès à cette même candidature, ou alors de restreindre la visibilité du commentaire à lui-même. Ces commentaires s'afficheront sur la vue individuelle d'une candidature.

Des points techniques globaux seront également évalués et devront donc être mis en place :

- **Gestion de exceptions**

Les exceptions et erreurs devront être gérées à tous les niveaux de l'application, c'est-à-dire, au niveau des interactions avec la base de données, des erreurs de l'API (renvoi des messages d'erreur à l'interface) et également de l'interface utilisateur.

- **Connexion à la base de données au travers d'un compte de service n'ayant que le minimum de droits nécessaires**

Le compte de service (utilisateur MySQL) utilisé par API pour se connecter à la base de données devra uniquement disposer des droits nécessaires aux interactions effectuées. Le script de création de ce compte sera compris dans le script de création de la base de données.

- **Respect des directives web EPFL basiques**

Le choix du design général de l'application est libre, mais devra respecter des directives EPFL de base, c'est-à-dire d'avoir le logo EPFL en haut à gauche de la page et un pied de page comprenant le texte « © EPFL, tous droits réservés. »

- **L'application fonctionne sur les navigateurs modernes**

Les technologies utilisées devront être compatibles avec les navigateurs modernes Firefox et Google Chrome. L'application devra fonctionner et s'afficher correctement sur ces deux navigateurs.

- **L'application n'est pas sensible aux attaques XSS et aux injections SQL**

Une attaque **XSS** ou Cross-Site Scripting est l'exploitation d'une faille dans une application permettant l'injection de code dans des variables non validées. Ces variables risquent ensuite d'être stocké en DB et de causer des dysfonctionnements ou failles de manière globales à l'application.

Une **injection SQL** est l'exploitation de faille dans les requêtes SQL vers une DB, il s'agit d'injecter un morceau de requêtes dans la requête qui va être exécuté afin d'en modifier le comportement et passer au travers de potentielles vérifications.

Pour désensibiliser l'application à ces deux types d'attaques, il faut impérativement inclure un processus de validation pour toutes données envoyées depuis le client vers l'API. Ne jamais faire confiance aux utilisateurs est le seul moyen de se prévenir de ce type d'attaques.

- **Le code respecte les conventions Vue.js**

Les convention Vue.js de catégorie A¹ doivent être appliquées.
En cas d'exceptions, elles devront être documentées.

Afin de démontrer ces conventions dans un cas plus pratique, elles seront détaillées plus tard dans le rapport².

Afin d'avoir plus de précisions sur certains points du cahier des charges, quelques questions ont été posées au chef de projet :

Q : Est-ce que les responsables peuvent également attribuer des notes et commentaires sur les candidatures ?

R : Non, ces fonctionnalités sont réservées aux formateurs, cependant les commentaires publics sont également visibles par les responsables.

Q : Est-ce qu'il faut également mettre à jour les fonctionnalités du formulaire de candidatures suite aux modifications de la base de données

R : Non, ces modifications ne sont pas comprises dans le projet et se feront ultérieurement.

Q : Est-ce que les formateurs peuvent voir toutes les candidatures qu'importe leur statut ?

R : Non, les formateurs ne peuvent voir uniquement les candidatures notées comme valides.

2.2 Faisabilité du projet

Le projet dans son ensemble semble tout à fait réalisable, les prérequis techniques demandés (Vue.js, PHP, MySQL) sont acquis et ne devraient donc pas poser de problèmes ou bloquer la bonne progression du projet dans son ensemble.

Les points du cahier des charges sont assez précis, mais laissent tous de même une entière liberté au niveau de leurs réalisations, les quelques questions posées au chef de projet ont permis de répondre aux dernières incertitudes sur ces points. Les fonctionnalités demandées semblent réalisables au niveau de mes connaissances et compétences techniques. Certaines de ces fonctionnalités sont d'ailleurs déjà partiellement implémentées, cependant beaucoup de modifications devront y être apportées.

¹ <https://fr.vuejs.org/v2/style-guide/index.html>

² Voir point 3.6.7

Le temps à disposition est fixé à 88 heures sur 11 jours. J'estime que la charge de travail relativement correcte par rapport au temps à disposition, ce qui me laissera un minimum de marge d'erreur sur la planification du projet.

Je prévois d'anticiper au maximum le travail en cas d'avance sur la planification et en cas de problèmes, de ne pas rester bloquer trop longtemps sur la tâche qui pose un problème. En cas de manque de temps, je privilégierai la rédaction des documentations quitte à laisser une fonctionnalité de l'application partiellement ou non terminée.

2.3 Choix des technologies

Les technologies principales de ce projet ont été choisies au préalable suite à des discussions avec le chef de projet. Finalement le choix s'est porté vers PHP pour le backend (API), MySQL (DB) et Vue.js pour le frontend (**UI / UX**).

L'utilisation d'un framework PHP pour la réalisation de l'API a également été mise en avant et j'ai donc porté mon choix sur le micro-Framework Lumen³. Le fait que Lumen fournisse uniquement les méthodes permettant la réalisation d'API est un argument qu'y m'a conforté dans le choix de ce Framework

Au niveau du frontend, le choix de Vue.js était très important car je dispose d'une expérience conséquente avec ce Framework, notamment grâce à la réalisation de divers projets d'envergures que j'ai eu l'occasion de réaliser avec cette technologie.

Quant au design de l'application, étant donné qu'aucune technologie n'est imposée, j'ai choisi d'utiliser un Framework nommé Vuetify⁴. Malgré le fait que je ne l'ai jamais utilisé auparavant, j'estime au vu de sa documentation qu'il répondra tout à fait au besoin de l'application, notamment grâce à sa simplicité d'implémentation du fait qu'il soit développé en Vue.js, de son large choix de composants et de sa documentation très complète.

2.4 Planification initiale

La planification initiale se trouve en annexe.

Afin de garder une trace des éventuels problèmes ou bugs dans mon application et également d'organiser mon travail, j'utiliserai la plateforme Trello qui me permet de créer des tableaux de tâches et d'y ajouter des remarques.

³ Voir point 2.5

⁴ Voir point 2.5

2.5 Matériel et ressources à disposition

Ma machine de travail sera un PC sous Windows 10 Education doté d'un IDE (éditeur de code Visual Studio Code), d'un client SQL (MySQL Workbench) et des dépendances suivantes :

Git

Git est un logiciel de gestion de version (VCS) ainsi qu'un repository sur la plateforme C4science (<https://c4science.ch/source/canapgest/>), il est documenté en détail au point 2.6.

WampServer

WampServer est une plateforme de développement web conçue pour Windows, elle permet d'héberger un serveur web tournant sur Apache, accompagné de MySQL et PHP.

La plateforme apporte également quelques raccourcis vers des actions de configuration du serveur, par exemple **vHosts**, configuration PHP, etc.

Dans le cadre de ce projet, j'utiliserai PHP en version 7.2, téléchargeable sur <http://www.wampserver.com/>

Composer

Composer est un gestionnaire de paquet pour PHP, dans le cadre du projet il est utilisé pour installer le Framework Lumen et ses dépendances.

- Télécharger l'installateur sur : <https://getcomposer.org/doc/00-intro.md#installation-windows>
- Ajouter le chemin vers PHP à la variable d'environnement PATH

Pour valider l'installation :

```
C:\> composer
```

Lumen

Lumen est un micro-Framework basé sur le Framework **Laravel**, il réutilise en partie le fonctionnement de ce dernier, en retirant les éléments servant à la partie Vue. Lumen est donc parfaitement adapté pour réaliser une API, il est également très facile à prendre en main du fait de la simplicité de son code et de sa documentation et de sa grande communauté (principalement celle de Laravel). Il embarque par exemple des méthodes pour l'authentification, la validation, etc.

Pour créer un projet Lumen avec une structure de base :

```
C:\.> composer create-project --prefer-dist laravel/lumen <nom_projet>
```

Puis démarrer l'application :

```
C:\..\projet> php -S localhost:8000 -t public
```

Pour la configuration, il est nécessaire d'éditer le fichier `.env` se trouvant à la racine du projet avec les informations demandées : `APP_KEY` (Il suffit de générer une chaîne de caractères pour cette valeur) et info de connexion à la DB.

firebase/php-jwt

Librairie pour l'utilisation de tokens **JWT** avec PHP, qui permettent de transmettre des informations de manière sécurisée et de faire office d'authentification du client lors des requêtes à l'API.

Pour importer cette librairie, ouvrir un terminal dans le dossier du projet Lumen, puis :

```
C:\..\projet> composer require firebase/php-jwt
```

Dans le fichier «.env», ajouter un champ `JWT_SECRET`, la valeur de ce champ sera la clé privée qui servira à chiffrer/déchiffrer les tokens JWT. Il suffit de générer une chaîne de caractères pour cette valeur

Node.js :

Node.js est un environnement JavaScript permettant d'interpréter du JavaScript côté serveur.

Dans le cadre du projet, **Node.js** servira uniquement au développement, notamment pour installer les dépendances grâce à **NPM**, son gestionnaire de paquets inclus nativement.

Téléchargement et installation Node.js (v.11 actuellement) sur : <https://nodejs.org/en/>

Vue.js

Vue.js est un Framework JavaScript conçu pour développer des interfaces d'application web **SPA**, c'est-à-dire à page unique. Il est orienté sur l'utilisation de composants leur interactions. Au point 3.6.2, l'utilisation de ce Framework est décrite en détails.

La documentation officielle est disponible sur : <https://vuejs.org/>

@vue/cli

Outil en ligne de commande (**CLI**) servant à générer des projets Vue.js avec une structure de base, en choisissant les dépendances souhaitées.

Il faut installer le CLI de manière globale :

```
C:\> npm install @vue/cli -g
```

L'argument **-g** permet d'installer le module de manière globale au système.

Une fois installé on peut donc l'utiliser depuis une invite de commande pour créer un nouveau projet :

```
C:\> vue create <nom_projet>

? Please pick a preset: Manually select features

? Check the features needed for your project:
  (*) Babel
  ( ) TypeScript
  ( ) Progressive Web App (PWA) Support
  (*) Router
  (*) Vuex
  ( ) CSS Pre-processors
  (*) Linter / Formatter
  ( ) Unit Testing
  ( ) E2E Testing

? Use history mode for router ? N

? Pick a linter / formatter config : ESLint with error prevention only

? Pick additional lint features: Lint on save

? Where do you prefer placing config for Babel etc. ? In dedicated config files
```

Le « history mode » cité plus haut, permet d'enlever le "#" dans l'url de l'application qui est affiché par défaut par vue-router. Si l'on choisit oui, il faudra avoir un `.htaccess` (ou autre) qui redirige toutes les requêtes sur `index.html`, car l'utilité du "#" est justement de faire comprendre à Vue Router qu'il s'agit d'une URL de l'application.

ESLint est un **linter** JavaScript permet de valider la manière dont le code est écrit et d'assurer les bonnes pratiques en manière de développement (indentation, code superflu etc.) en se basant sur les normes **ECMAScript**. Ici, j'utilise les standards officiels JavaScript⁵ comme configuration.

Les frameworks JavaScript actuels utilisent la norme ECMAScript Version 6, appelée **ES6**, alors que les navigateurs ne supportent pas encore totalement cette norme. Pour cette raison, il est nécessaire d'utiliser un **transcompilateur** (en JavaScript, paquet Babel installé précédemment) qui permet de convertir notre code vers la norme **ES5**, supportée par les navigateurs.

Pour commencer le développement Vue.js, il faut installer les dépendances puis démarrer le serveur de développement :

```
C:\..\> cd projet
C:\..\projet> npm install
C:\..\projet> npm run serve
```

Cette commande va exécuter la fonction « serve » se trouvant dans le fichier `package.json`, situé à la racine du projet, qui va donc démarrer le serveur.

Lors du déploiement en production de l'application, il suffira d'exécuter la commande :

```
C:\..\projet> npm run build
```

Cette commande va optimiser le code source en générant un site statique, avec un unique point d'entrée `index.html`, dans le dossier `dist`.

Vue Router

Comme son nom l'indique, Vue Router permet de disposer d'un routeur dans une application Vue.js. Vu que l'application est considérée comme une SPA, Vue Router va interpréter l'URL pour finalement générer les composants selon les routes et paramètre fournis.

Sa documentation est disponible ici : <https://router.vuejs.org/>

Son utilisation est détaillée au point 3.6.2

⁵ <https://standardjs.com/>

Vuex

Vuex est un module pour Vue.js qui permet de disposer d'un état centralisé (**store**) pour l'application, ce qui a pour but de simplifier la structure et la dépendance des composants Vue.js entre eux, principalement leurs interactions.

Sa documentation est disponible ici : <https://vuex.vuejs.org/fr/> , son utilisation est détaillée au point 3.6.2.

Vuetify

Vuetify est un Framework de composants CSS écrit en Vue.js. Il sera utilisé pour réaliser l'interface de l'application web.

Pour l'ajouter à un projet Vue.js à l'aide de @vue/cli :

```
C:\..\projet> vue add vuetify
```

Son utilisation est détaillée au point 3.6.3

Axios

Axios est une librairie JavaScript / Node.js servant à effectuer des requêtes AJAX (XMLHttpRequest - **XHR**)⁶ , une requête XHR est un objet permettant l'interaction client-serveur sans rafraichissement de page. Dans ce projet, il sera utilisé pour effectuer des requêtes vers l'API.

2.6 Versionning et sauvegardes

J'utiliserai dans ce projet, le logiciel de gestion de version (**VCS**), très populaire, nommé Git. Git permet de créer des versions, appelés **commit**, d'un ensemble de fichiers. Git permet également de créer des **branches**, permettant de travailler sur plusieurs états du projet en parallèle sans affecter la version stable, la branche par défaut est nommée **master**.

Un commit se compose d'un **hash** (signature) unique, d'un auteur et d'un message (optionnel mais évidemment très recommandé). Le hash permet d'identifier le commit et d'y effectuer des actions, par exemple d'effectuer un retour en arrière dans les versions.

Git permet également de publier l'état de l'arbre des versions vers un serveur dédié **remote/repository**, dans le cadre de ce projet, il s'agit de c4science.ch, plateforme git dédiées aux universités suisses et recommandé par l'EPFL.

La plateforme c4science permettra de disposer de suivre l'avancement du travail à travers les différents commits et de comparer les modifications effectuées entre deux versions.

⁶ <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>

Exemple d'utilisation :

On initialise le repository :

```
C:\..> mkdir test-repo  
C:\..> cd test-repo  
C:\..\test-repo> git init
```

On ajoute un nouveau document :

```
C:\..\test-repo> echo test >> readme.txt
```

On affiche l'état actuel des modifications :

```
C:\..\test-repo> git status
```

On « stage » les modifications :

```
C:\..\test-repo> git add *
```

On « commit » :

```
C:\..\test-repo> git commit -m "votre message"
```

On peut également ajouter un repository distant et y publier les modifications :

```
C:\..\test-repo> git remote add origin "lien du repository"  
C:\..\test-repo> git push
```

J'effectuerai des commits / push de manière régulière, lors de modifications majeures et d'un état majoritairement stable de l'application et des documents.

Je garderai également une copie du projet à chaque publication vers c4science son mon espace personnel EPFL qui dispose d'une sauvegarde par heure.

En cas de problème avec ma machine de travail ou si j'effectue des modifications qui rendraient le projet inutilisable, je pourrais récupérer l'avancée de mon travail avec un minimum de pertes.

3 Réalisation

3.1 Modèle DB

3.1.1 MCD

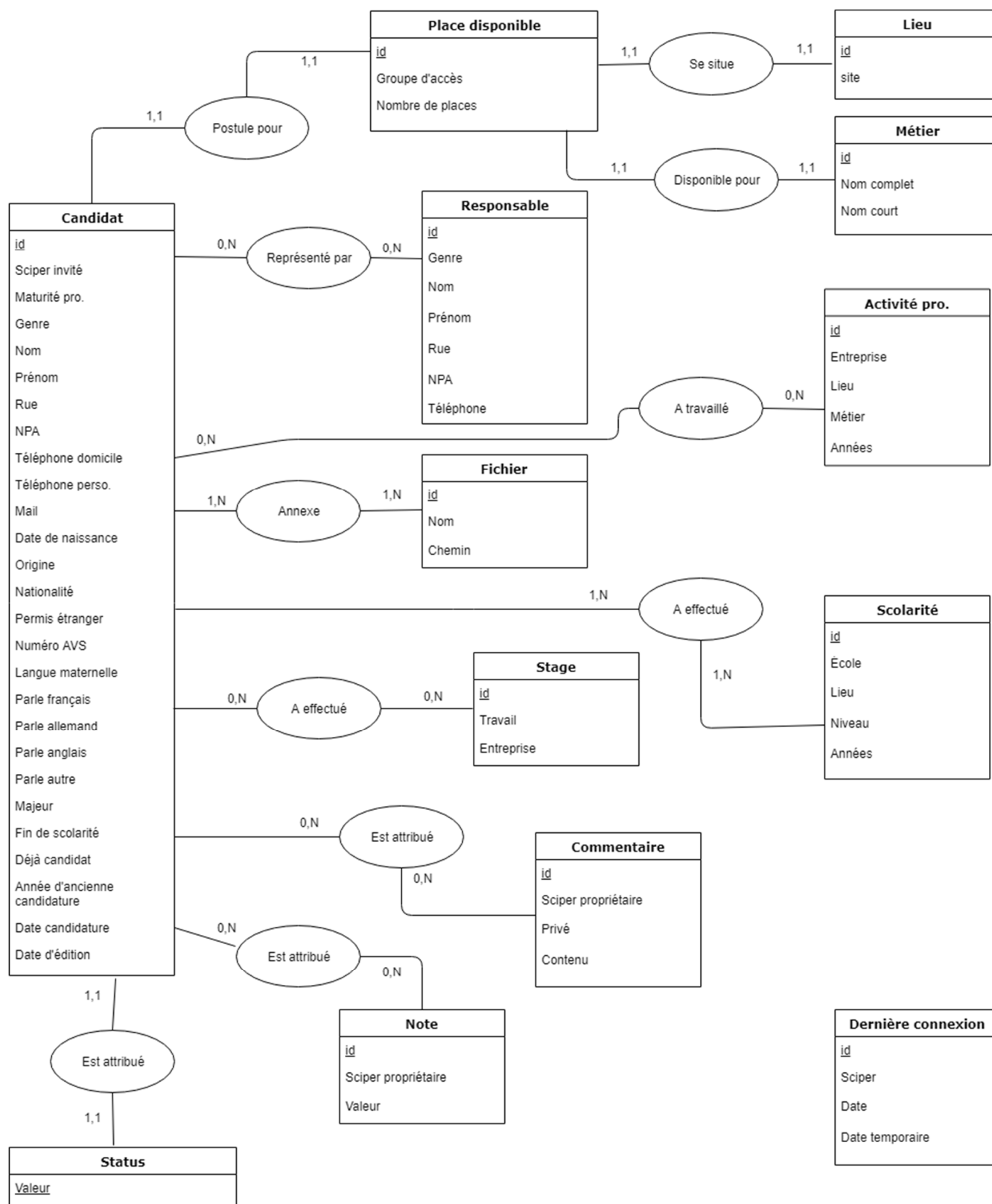


Figure 3: MCD

3.1.2 Modèle DB - MPD

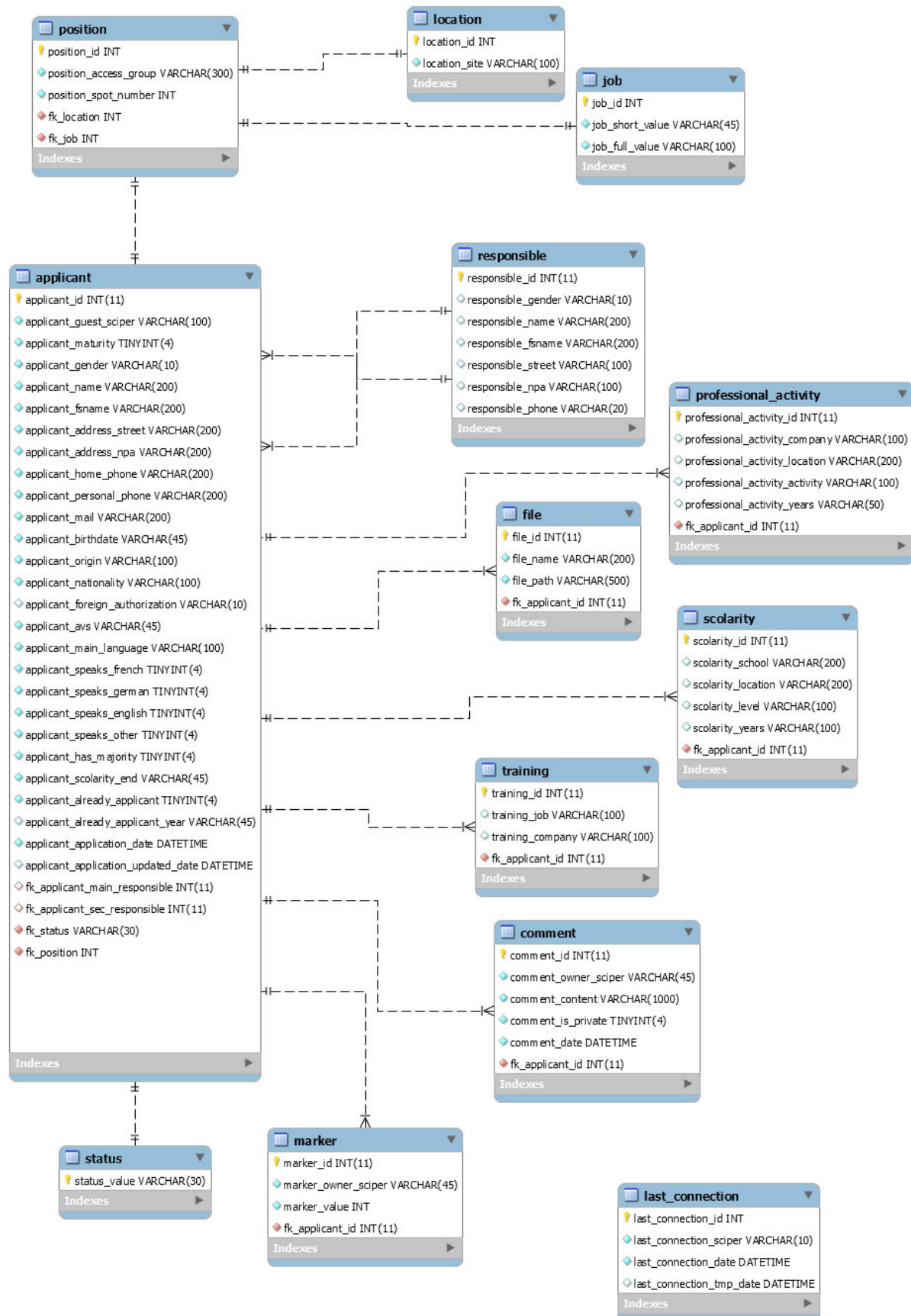


Figure 4: MPD

3.2 Structure de l'application

Comme expliqué précédemment, l'application web est séparée en plusieurs éléments ; Le Frontend, qui comprend l'interface web coté client (Vue.js), et le Backend qui englobe l'API et la DB ainsi que tout autre service, par exemple le service d'authentification **Tequila**.

Ce type d'architecture logicielle est très utilisée dans les services web, celle-ci est appelée **REST** ou RESTful.

Les principes d'une architecture REST sont d'offrir des fonctions à tous ordinateurs ou clients web en leur proposant des ressources uniquement textuelles, par exemple au format JSON, XML etc. Un service REST est dit **stateless** (sans état), c'est-à-dire qu'aucune connexion permanente n'est établie entre le client et le serveur et qu'aucune session n'est conservée du côté serveur, chaque requête est traitée indépendamment.

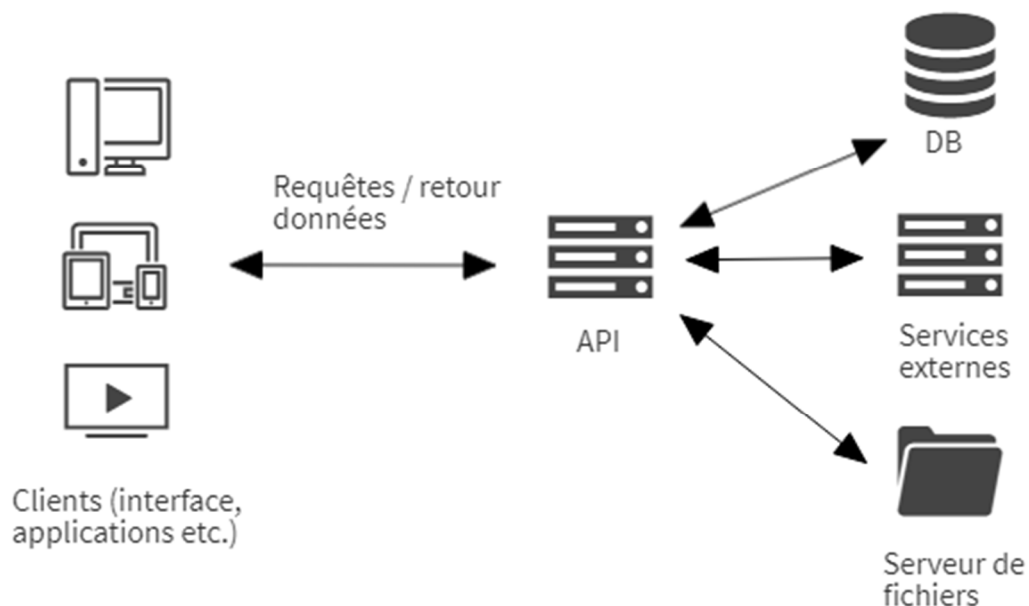


Figure 5: Architecture REST

Les avantages principaux de l'utilisation d'architectures REST sont donc nombreux :

- Tout d'abord le fait que le client ne soit pas lié au reste de l'application sépare le code et permet donc la maintenance distincte sans conséquences sur le fonctionnement global, de ce fait il serait possible par exemple d'avoir plusieurs versions de l'API fonctionnant en même temps, sans incidence sur le fonctionnement des clients. Du fait de la séparation de l'application, plusieurs clients, par exemple une interface web et une application mobile, pourront utiliser la même API REST et donc réduire considérablement le temps de développement du service.

- La charge du serveur est inférieure à celle d’une architecture plus « standard », principalement due à l’absence de la gestion des états (sessions). Le service est donc plus simple et plus performant lors de grand nombre de requêtes simultanées. En cas de service à forte demande, une architecture REST est plus facilement adaptable pour une répartition sur plusieurs serveurs (scalabilité), vu qu’il n’y a pas de session à conserver sur un serveur spécifique ou à répliquer sur des serveurs multiples.
- REST fonctionne avec l’utilisation du protocole HTTP, ce qui permet d’en exploiter ces fonctionnalités, par exemple ses entêtes et méthodes (**verbs**, types de requêtes). On utilise principalement les méthodes GET, POST, PUT, PATCH, DELETE, HEAD, CONNECT, OPTIONS et TRACE.

L’architecture REST est donc un choix judicieux et en vogue depuis bien des années, c’est d’ailleurs l’architecture dominante en termes de service web. D’un point de vue développement, ce genre d’architecture est bien plus agréable, principalement du fait de la séparation frontend et backend.

3.3 Authentification

3.3.1 Tequila

Tequila est le service d’authentification web de l’EPFL. Il fait office d’Identity Provider (**IdP**) et de Single sign-on (**SSO**), c’est-à-dire qu’il permet la connexion à de multiples services web avec une seule identité. Il fonctionne ainsi au travers de **LDAP**, permettant donc d’authentifier l’accès à des applications web et de récupérer des attributs de l’utilisateur.

Le serveur Tequila est centralisé pour authentifier une application il faut donc utiliser un client Tequila⁷. Ce client dispose de plusieurs versions écrites en PHP, Perl, JAVA et Ruby. Etant donné qu’il est **opensource**, la communauté à développer d’autres versions des clients, notamment pour son utilisation en Python (Django⁸) et JavaScript (Node.js).

Dans le cadre de ce projet, il s’agit évidemment du client PHP qui est utilisé, l’authentification de l’API au travers de Tequila est expliquée au chapitre suivant.

3.3.2 Token

Afin de sécuriser les données et l’utilisation de l’API et interface, une authentification a été mise en place. Une API REST n’utilise pas de session, il n’y pas de moyen d’établir une connexion « permanente » entre l’API et le client, c’est pour cela qu’il faut authentifier chaque requête faite depuis les clients vers l’API.

⁷ Disponible ici : <https://tequila.epfl.ch/download.html>

⁸ Voir <https://www.djangoproject.com/>

J'ai donc choisi d'implémenter l'utilisation des JSON Web Token (JWT) (voir 3.4.2).

Lorsque qu'un client se rend sur l'interface, il envoie une requête GET sur une route spécifique pour la connexion à l'API, celle-ci va effectuer la connexion via le service EPFL Tequila, qui en cas d'authentification valide, renverra les données EPFL de l'utilisateur à l'API.

L'API va ensuite générer le token JWT contenant ces données, qu'elle enverra en tant que réponse au client. Le client stockera ce token et l'utilisera pour utiliser les requêtes nécessitant une authentification.

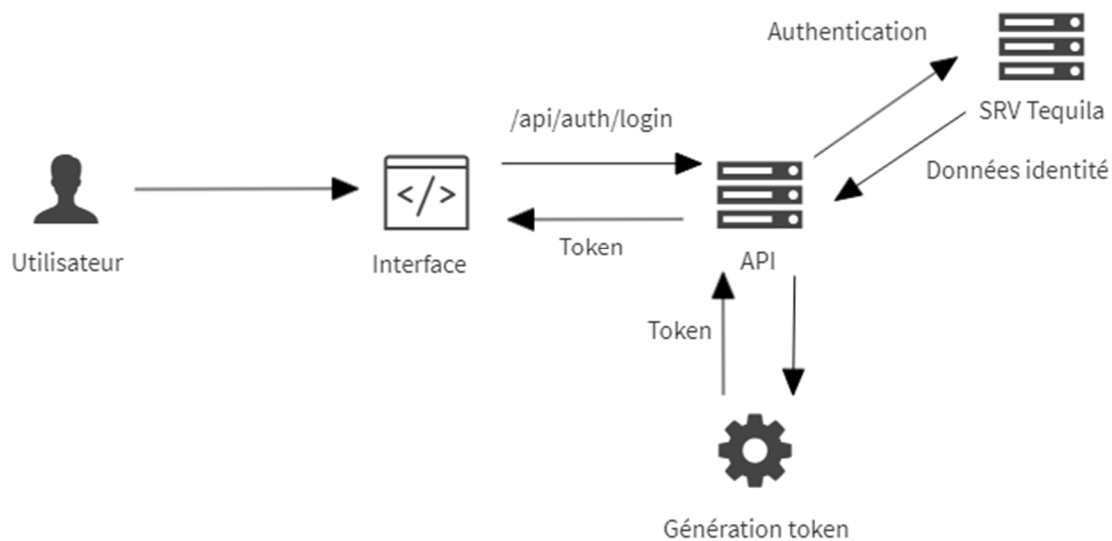


Figure 6: Requête d'authentification

Un token JWT est une chaîne de caractères encodés qui se compose de trois parties distinctes. Prenons l'exemple suivant (dont la deuxième partie a été raccourcie) :

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJsZyY3ODk3Nn0.upmPwSUNZdJ-FYhtjHch3FTz_rOnR9g-eEViFLVt5k

Ces trois parties sont séparées par un « . », la première est appelée « header », elle contient le type de token et l'algorithme de chiffrement utilisé pour le générer (ici, **HS256**).

Déchiffrée, cette chaîne correspond à la structure suivante :

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

La deuxième est appelée « payload », elle contient les données, dans le cas de ce projet, la structure est la suivante :

```
{
  "iss": "canap-gest",
  "sub": "262544",
  "tequila_data": {
    "firstname": "Nicolas Benjamin",
    "name": "Crausaz",
    "group": "...",
    "user": "ncrausaz",
    "sciper": "262544"
  },
  "permissions": [...],
  "role": "responsable",
  "iat": 1553592576,
  "exp": 1553678976
}
```

Les valeurs « iat » et « exp » sont obligatoires, elles représentent le **timestamp** de génération et d'expiration. Lors de la validation du token, une vérification du timestamp d'expiration s'effectue, si la date est passée, l'authentification sera donc révoquée.

Les autres valeurs ont été ajoutées dans le code au niveau serveur, le token est donc utilisé afin de pouvoir récupérer les informations retournées par Tequila, ici le nom, le sciper, username, les métiers autorisés et les groupes auxquels l'utilisateur appartient. Ces groupes proviennent du service de gestion de groupes LDAP de l'EPFL⁹.

La dernière partie est nommée « signature ». En effet elle contient un hashage **HCSHA256** des données des parties précédentes (en **base64**) et de la clé de chiffrement du token (qui est définie du côté serveur) :

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  <encryp_key>
)
```

⁹ <https://groups.epfl.ch>

3.3.3 Processus

Lors que l'authentification a été réussie, le client reçoit et conserve le token, il le transmettra avec chaque requête qui nécessite une authentification. Pour ce faire, le token est envoyé dans le header HTTP « Authorization », l'API est donc configurée pour récupérer le token et effectuer la vérification nécessaire, valider le droit d'accès à la ressource puis de traiter la demande du client.

On configure Axios pour l'utilisation du token de la manière suivante :

```
import axios from 'axios'

const instance = axios.create({
  headers: { 'Authorization': "Bearer " + token},
  baseURL: 'http://localhost:8000/api'
})
```

Ainsi les requêtes du client à l'API sont désormais authentifiées.

Le diagramme suivant représente le processus d'une requête authentifiée :

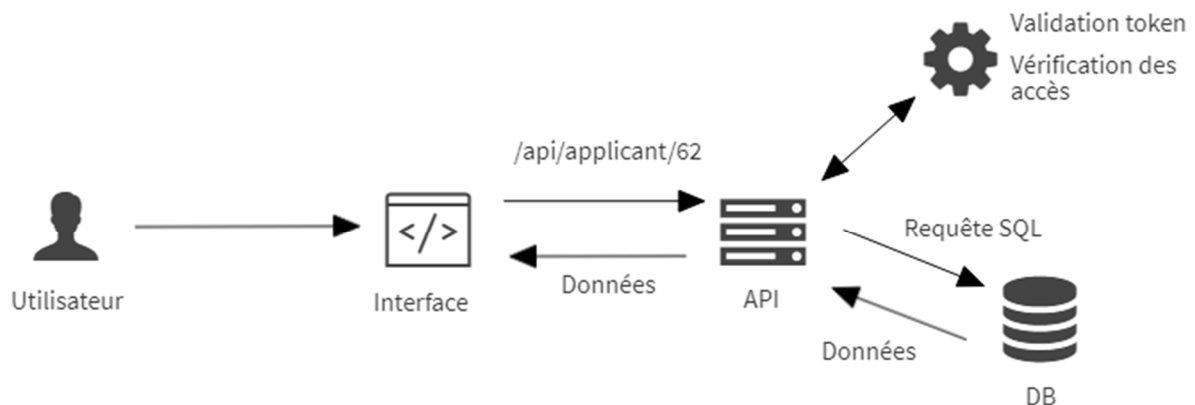


Figure 7: Requête authentifiée

3.4 Requêtes API

Ci-dessous se trouve une liste exhaustive des requêtes disponibles dans l'API, celles-ci groupées par « controller ».

Les données retournées sont au format JSON. Le choix de ce format est justifié du fait que :

- Ce format est et facile à lire. Aucun apprentissage n'est requis puisque la syntaxe n'utilise que quelques marques de ponctuations.
- Ne dépend d'aucun langage (format d'échange de données ouvert).
- Il est pris en charge par de nombreux langages : JavaScript, PHP, Python, Ruby etc.
- Sa structure de tableau et d'objets et sa syntaxe simple lui permet de rester très "léger" et facilement manipulable.
- Pour Vue.js, c'est l'idéal de recevoir les données en JSON car cela permet de les utiliser sans aucune « retouche ».
- Pour PHP / Lumen, l'encodage de tableaux vers JSON est très facile et rapide.

Les routes marquées en **VERT** ont été ajoutées lors du projet, les autres routes ont pour la plupart été modifiées partiellement.

Base de l'URL : <http://localhost:8000>
Auth : Token obligatoire
Rôle : limitation de la route à certains rôles (resp = Responsable)
Body : Contenu (données) obligatoires pour l'action de la requête

Authentification - AuthController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/auth/login	Non	tous	-	Login via Tequila, retour token
GET	/api/auth/logout	Non	tous	-	Déconnexion de Tequila

Utilisateur - UsersController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/user	Oui	tous	-	Retourne les infos de l'utilisateur connecté
GET	/api/user/permittedjobs	Oui	tous	-	Retourne les métiers visibles par l'user
GET	/api/user/hascommentedormarked	Oui	tous	-	Retourne les postulations commentées ou marquées par l'user
GET	/api/user/data/{sciper}	Oui	tous	-	Retourne les infos LDAP d'un utilisateur selon son sciper
GET	/api/user/connection	Oui	tous	-	Retourne la date de dernière connexion
PATCH	/api/user/connection	Oui	tous	-	Modifie la date de dernière connexion

Postulations – ApplicantsController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/applicants	Oui	tous	-	Retourne les postulations des tous les métiers accessibles (et non vides)
GET	/api/applicants/job/{job}	Oui	tous	-	Retourne les postulations d'un métier
GET	/applicant/{id}	Oui	tous	-	Retourne toutes les infos d'une postulation
GET	/applicant/{id}/export	Oui	tous	-	Exporte et télécharge les données (JSON) d'une postulation
DELETE	/applicant/{id}	Oui	resp	-	Supprime une postulation

Commentaires - CommentsController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/applicant/{id}/comments	Oui	tous	-	Retourne les commentaires d'une postulation, privés et publics
PUT	/api/comment	Oui	tous	content, is_private, applicant_id	Ajoute un commentaire
PATCH	/api/comment/{id}	Oui	tous	content, is_private	Edite un commentaire
DELETE	/api/comment/{id}	Oui	tous	-	Supprime un commentaire

Notes – MarkersController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/applicant/{id}/marker	Oui	tous	-	Récupère la note de l'utilisateur sur une candidature
PUT	/api/marker	Oui	tous	value, applicant_id	Ajoute une note
PATCH	/api/marker/{id}	Oui	tous	value	Édite une note
DELETE	/api/marker/{id}	Oui	tous	-	Supprime une note

Statut - StatusController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/status	Oui	tous	-	Récupère et télécharge un fichier
PATCH	/api/status/applicant/{id}	Oui	resp	status	Modifie le statut d'une postulation

Fichiers - FilesController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/files/{id}	Oui	tous	-	Récupère et télécharge un fichier

Offres (places d'apprentissage) – PositionController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/positions	Oui	resp	-	Récupère les places
GET	/api/locations	Oui	resp	-	Récupère les lieux
GET	/api/jobs	Oui	resp	-	Récupère les métiers
PUT	/api/position	Oui	resp	position_access_group, position_spot_number, location_id, job_id	Ajoute une place
PUT	/api/location	Oui	resp	location_site	Ajoute un lieu
PUT	/api/job	Oui	resp	job_full_value	Ajoute un métier
PATCH	/api/position/{id}	Oui	resp	position_access_group, position_spot_number, location_id, job_id	Modifie une place
DELETE	/api/position/{id}	Oui	resp	-	Supprime une place

Statistiques – StatsController

Type	Route	Auth (token) ?	Rôle	Body	Description
GET	/api/stats/total	Oui	tous	-	Récupère le nombre de postulation par métier

3.5 Interface

3.5.1 Structure

Les « pages » du site sont représentées par des routes interprétées par Vue Router, le routeur côté client.

Les routes disponibles sont les suivantes :

URL	Accès	Description
/	Tous	Page d'accueil
/applications	Tous	Page affichant la liste des candidatures
/application/{id}	Tous	Page affichant une candidature sélectionnée
/openjobs	Responsables	Page pour l'administration des places et métiers
/error	Tous	Page d'erreur
/*	Tous	Aucune page trouvée, redirection vers /error

3.5.2 Fonctionnalités

Certains éléments seront affichés en fonction du rôle de l'utilisateur connecté (formateur ou responsable). Ci-dessous, le rôle de chaque page sera expliqué et ses fonctionnalités détaillées.

Accueil - (/)

Une fois l'utilisateur connecté, il est redirigé sur la page d'accueil principale, cette page affiche le nombre de nouvelles candidatures disponibles depuis sa dernière connexion à l'application. Ce nombre est également affiché dans le menu de gauche.

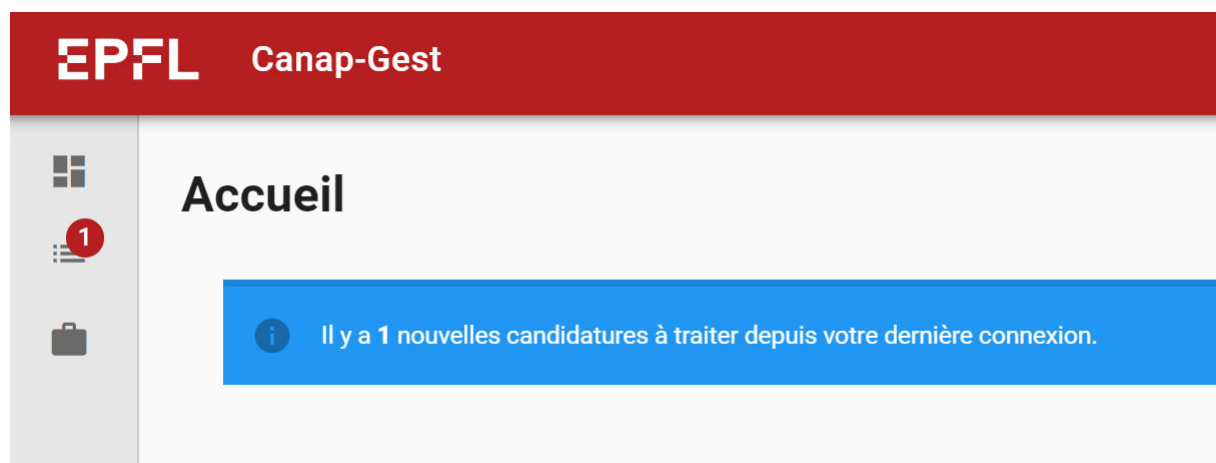


Figure 8: Page d'accueil

Candidatures – (/applications)

Cette page affiche un tableau contenant les candidatures. Ce tableau est triable et filtrable, par métier, par statut et est également équipé d'une pagination.

Depuis cette page, les responsables peuvent effectuer les tris dans les postulations en modifiant les statuts des candidatures et ont également la possibilité de les supprimer.

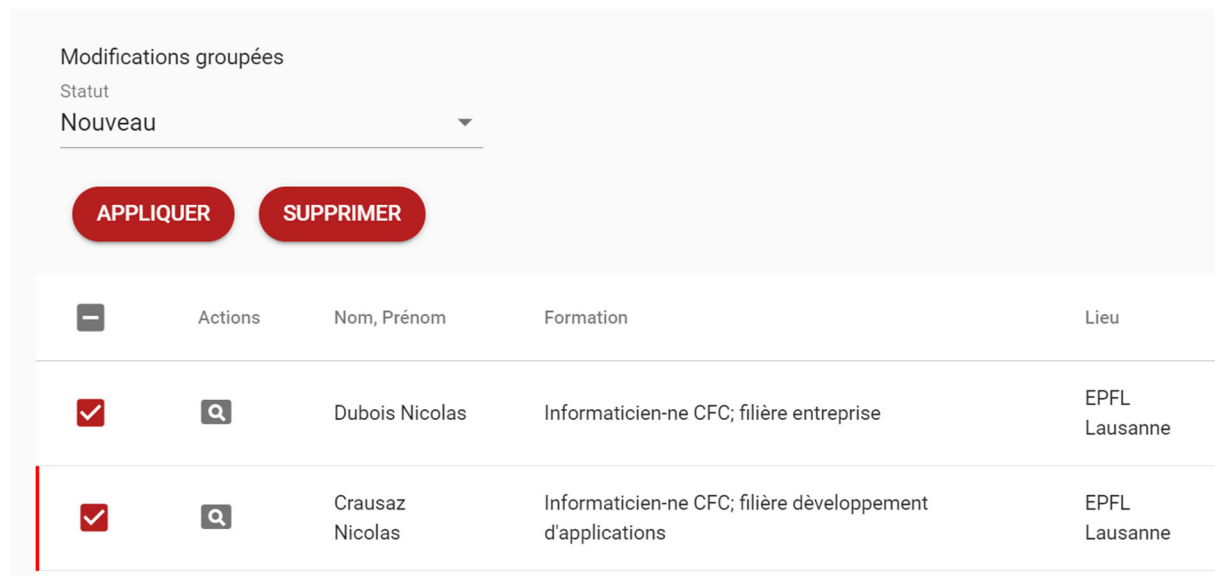


Figure 9: Edition multiple des candidatures

Les responsables peuvent sélectionner de manière multiple des candidatures afin de changer leurs statuts d'une seule action ou de les supprimer. Sinon ils peuvent modifier le statut d'une seule candidature grâce au sélecteur :

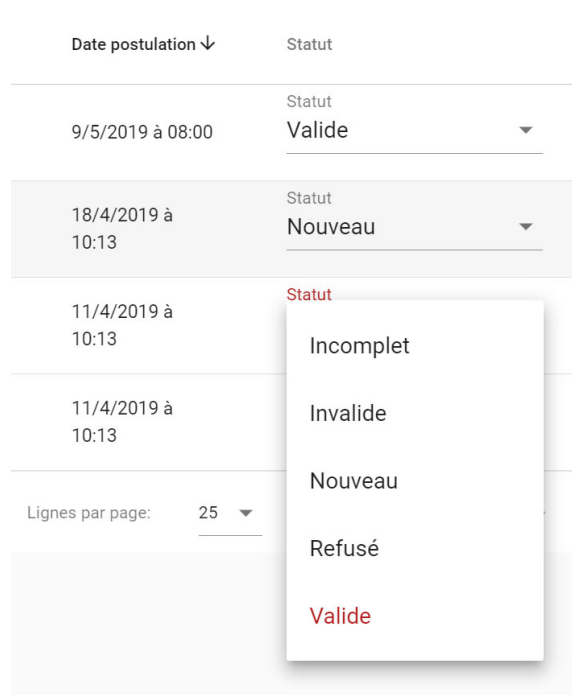


Figure 10: Edition du statut d'une candidature

Les formateurs ne peuvent pas effectuer ces actions mais peuvent visionner les candidatures ayant été validés auparavant par un responsable (statut valide).
Les nouvelles candidatures notifiées sont marquées d'un indicateur rouge afin de les retrouver plus facilement.

Postulations





Métier (tous) ▼					
Actions	Nom, Prénom	Formation	Lieu	Adresse	Email
	Dubois Nicolas	Informaticien-ne CFC; filière entreprise	EPFL Lausanne	test 1134 Vufflens	nicolas.
	Crausaz Nicolas	Electronicien-ne CFC	EPFL Lausanne	Village 5 1134 Vufflens	n.craus
	Crausaz Nicolas	Informaticien-ne CFC; filière technique des systèmes	EPFL Lausanne	Village 5 1134 Vufflens	n.craus
	Crausaz Nicolas	Informaticien-ne CFC; filière technique des systèmes	EPFL Lausanne	Village 5 1134 Vufflens	n.craus


Figure 11: Nouvelles candidatures

Candidature – (/application/{id})



Cette page regroupe toutes les informations fournies par un candidat et propose également aux les formateurs l'ajout de commentaires et l'attribution d'une note.



Les informations sont représentées par des « cartes » pour chaque catégorie de données (personnelles, représentants, scolarité, etc.). Les fichiers annexes d'une candidature sont également visibles et téléchargeables.



Données personnelles



Nom, prénom	Crausaz Nicolas
Formation, lieu	Informaticien-ne CFC; filière technique des systèmes, EPFL Lausanne
Maturité	✓
Adresse	Village 5, 1134 Vufflens
Email	n.crausaz99@gmail.com
Téléphones (privé, domicile)	123512351, 12351235
Date naissance, majeur	03/08/1999
Majeur	✓
Numéro AVS	21324.1235.123
Nationalité	Suisse
Origine	Chavannes
Langues	
Fin de scolarité	2015
Déjà candidat ? (Année)	X



Annexes



 [photo-passeport.pdf](#) 



 [carte-identite.pdf](#) 

 [curriculum-vitae.pdf](#) 

 [lettre-motivation.pdf](#) 

 [annexe1.pdf](#) 

 [annexe2.pdf](#) 

 [annexe3.pdf](#) 

TOUT TÉLÉCHARGER

Figure 13: Fichier annexes

Figure 12: Informations personnelles

Une barre d'outils se trouve en haut de la page et propose l'attribution d'une note, de démarrer une impression de la page, de contacter la candidat (mail) et d'exporter les données de la candidature vers un fichier JSON (téléchargement du fichier).

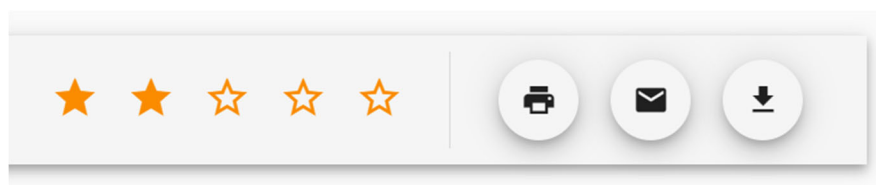


Figure 14: Barre d'outils, version formateur

L'attribution de commentaires s'effectue grâce au composant prévu à cet effet.


Commentaires

Ajouter un commentaire...

☐ Commentaire privé


AJOUTER


Commentaires publics



Test de commentaire
Nicolas Crausaz


9/5/2019 à 12:52





Un super commentaire !
Nicolas Dubois


9/3/2019 à 20:52



Un autre super commentaire !
Sylvain Tauxe

1/3/2019 à 10:52

Vos commentaires privés



Test de commentaire privé
Nicolas Crausaz

9/5/2019 à 06:58




Figure 15: Visualisation, ajout et modification de commentaires

Un formateur peut ajouter, voir les commentaires publics et ses commentaires privés et les éditer.

Pour les responsables, l'ajout de note ou de commentaire n'est pas disponible mais ils ont la possibilité de visionner les commentaires publics ayant été ajoutés par les formateurs. Un bouton de suppression de la candidature est également disponible dans la barre d'outils.

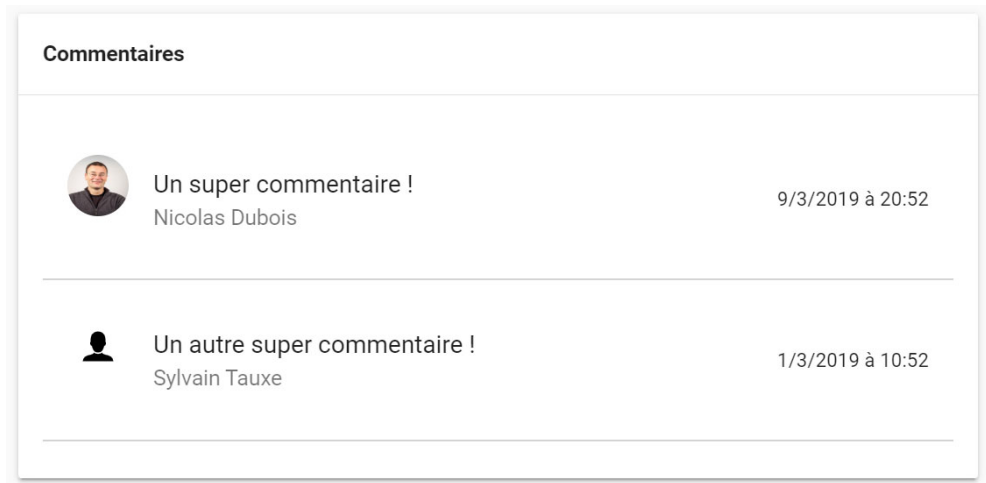


Figure 16: Visualisation des commentaires pour responsables

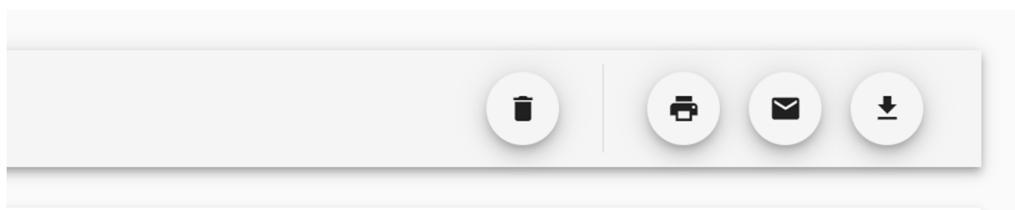


Figure 17: Barre d'outils, version responsable

Le bouton de suppression remplace donc la fonctionnalité d'attribution de note.

Places disponibles – (/openjobs)

Cette page réservée aux responsables affiche un tableau contenant la liste des places d'apprentissages disponibles. Elle permet aux responsables de créer des offres et de définir le nombre de places disponibles. Les offres sont éditables et supprimables à condition qu'aucune candidature y soit liée, dans ce cas il faudra finir de traiter / supprimer les candidatures liées. La possibilité d'ajouter des métiers et des lieux est également présente dans cette page.

Nouvelle place

Métier ▼ + Lieu ▼ +

Groupe d'accès

Nombre de places
0

ANNULER CONFIRMER

EPFL Lausanne formateurs-planificateurs-electriciens

EPFL Lausanne formateurs-laborantins-en-biologie

Figure 18: Ajout d'une offre d'apprentissage

Edition

Métier ▼ + Lieu ▼ +

Polymécanicien-ne CFC

EPFL Lausanne

Groupe d'accès

formateurs-polymecaniciens

Nombre de places
1

ANNULER CONFIRMER

EPFL Lausanne formateurs-planificateurs-electriciens

EPFL Lausanne formateurs-laborantins-en-biologie

Figure 19: Édition d'une offre d'apprentissage

Erreur – (/error)

En cas d'erreur, l'application est redirigée vers cette page d'erreur qui affiche le statut (code HTTP¹⁰), l'url demandée et un message décrivant l'erreur.



Figure 20: Page d'erreur

¹⁰ Voir https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

3.5.3 Technologies

Vue.js

Vue.js est un Framework JavaScript permettant de réaliser des interfaces d'application web monopage (**SPA**), c'est-à-dire sans rechargement à chaque action demandée. Le principe d'une SPA est de fluidifier l'expérience utilisateur, en se basant sur l'utilisation de composants réactifs.

Le principe de Vue.js est de ne plus interagir directement avec le **DOM** (document éditable par des script, contenu du navigateur). Pour se faire, on va créer des composants basés sur un « état » qui, lorsqu'il sera modifié, mettra à jour le composant. Pour mieux comprendre, voici un exemple de composant assez basique :

```
<template>
  <ul id="exemple">
    <li
      v-for="todo in todos"
      :key="todo.id"
      class="test"
      @click="logValue(todo.text)"
    >{{ todo.text }}</li>
  </ul>
</template>
<script>
export default {
  name: 'exemple',
  data() {
    return {
      todos: [
        { id: 1, text: 'Todo 1' },
        { id: 2, text: 'Todo 2' }
      ]
    }
  },
  methods: {
    logValue(value) {
      console.log(value)
    }
  }
}
</script>
<style scoped>
.test {
  border: 1px solid black;
}
</style>
```

On constate qu'un composant est composé de trois parties : le template, le script et le style, malgré que ce dernier ne soit pas obligatoire. Le template sera le rendu de notre composant, il est défini par des balises HTML. Le script contient l'état du composant et sa logique.

Un avantage conséquent de Vue.js est sa facilité de prise en main, en effet si l'on connaît déjà HTML, CSS et JavaScript son apprentissage se fait très facilement. Il est également prévu pour un apprentissage progressif, c'est-à-dire que les premiers projets peuvent très bien être conçus sans y ajouter d'autres modules (par exemple Vuex).

Vuex

Vuex est un gestionnaire d'état qui permet de disposer d'un état centralisé (appelé **store**) pour l'application avec lequel les composants interagissent pour effectuer des actions et interagir avec les données.

En utilisant Vuex, on va effectuer toutes les actions tels les requêtes vers l'API au niveau du store puis modifier l'état du store. Afin d'assurer la cohérence des données et la mise à jour au travers de tous les composants, le store ne peut pas être modifié de manière imprévisible grâce à l'utilisation de **mutations**. Les actions vont donc appliquer des mutations avec les données reçues pour mettre à jour le store.

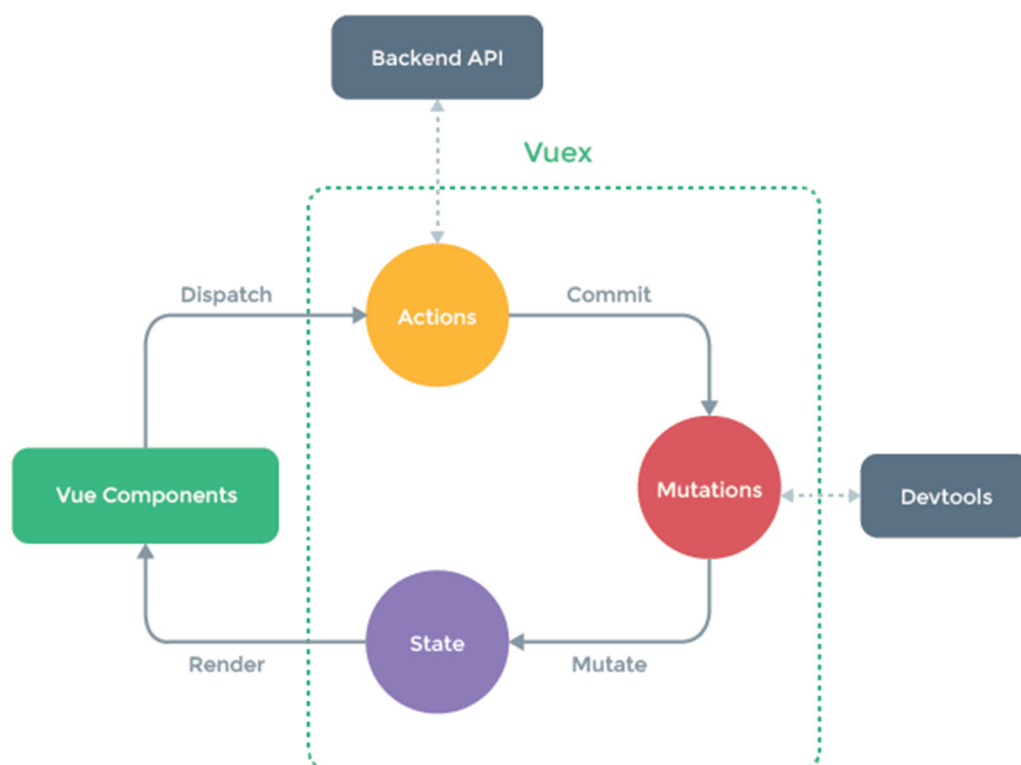


Figure 21: Fonctionnement de Vuex

L'utilisation de Vuex n'est pas obligatoire dans la réalisation d'un projet avec Vue.js. Sans store, l'implémentation changera radicalement car il sera nécessaire de gérer les données au niveau des composants qui devront interagir entre eux à l'aide de passage de propriétés (parent à enfant) ou avec des bus d'événements (emit, watchers etc.) afin d'être réactifs aux modifications de ces données. On se rend donc vite compte lors de projets conséquents que l'utilisation de Vuex, bien que cela nécessite un apprentissage d'utilisation, permet finalement de simplifier le développement, la structure et la propreté du code.

Pour conclure cette explication, je citerai Dan Abramov, auteur d'un module de store centralisé pour un Framework concurrent (React.js) :

« Les librairies Flux (Vuex et autres), c'est comme les lunettes : vous saurez quand vous en aurez besoin »

3.5.4 Design

Comme cité précédemment, j'ai porté mon choix sur le Framework Vuetify pour la réalisation du design de l'interface.

Malgré le fait que je n'ai jamais utilisé ce Framework auparavant, mon expérience avec d'autre frameworks concurrents me servira le pour prendre en main facilement et rapidement.

Vuetify dispose du grand avantage d'être écrit en Vue.js, son implémentation dans l'application est donc très simple, ne pose pas de problème et il n'y a donc pas besoin d'adapter les composants pour l'utilisation avec Vue.js comme on devrait le faire pour un autre Framework CSS quelconque.

Vuetify est également très personnalisable, j'ai donc modifié quelques éléments du thème standard en apportant un design aux couleurs de la charte graphique EPFL¹¹.

¹¹ Charte visible sur : <https://epfl-idevelop.github.io/elements/#/colors>

Vuetify est livré en anglais, j'ai donc inclus et modifié la **locale** pour obtenir une interface en français en incluant le code suivant dans le fichier `Site/src/plugins/vuetify.js`

```
import Vue from 'vue'
import Vuetify from 'vuetify/lib'
import 'vuetify/src/stylus/app.styl'
import fr from 'vuetify/es5/locale/fr'

Vue.use(Vuetify, {
  iconfont: 'md',
  theme: {
    primary: '#B51F1F',
    secondary: '#E6E6E6',
  },
  lang: {
    locales: { fr },
    current: 'fr'
  }
})
```

J'ai également respecté la directive EPFL de disposer le logo en haut à gauche et d'un pied de page mentionnant « © EPFL, tous droits réservés ».



Figure 23: Logo

© 2019 EPFL, tous droits réservés

Figure 22: Pied de page

3.5.5 Pratiques de code

De manière générale, j'essaie de réaliser le code le plus optimisé et de respecter les bonnes pratiques des différents langages. Avec Vue.js, les bonnes pratiques sont décrites en tant que conventions¹², celles de priorité A ont été respectées lors de ce projet :

Les noms de composant devraient toujours être des mots multiples

A l'exception du composant racine `App`, les composants créés respectent un nommage le plus explicite possible et à mot multiples, par exemple : `applicationsFilters.vue`, composant utilisé pour les filtres de candidatures.

¹² Visibles sur : <https://fr.vuejs.org/v2/style-guide/index.html>

La propriété data doit être une fonction

La propriété data, qui représente les données d'un composant, doit impérativement être une fonction, tel que dans l'exemple suivant tiré du composant *applicationsFilters* :

```
data () {  
  return {  
    jobFilter: null,  
    statusFilter: null,  
    search: ''  
  }  
}
```

Les définitions de props devraient être aussi détaillées que possible

Les props d'un composant sont des données qui sont passées d'un composant parent vers un enfant, on les utilise de la manière suivante :

```
<!-- Composant parent -->  
<applicationCommentItem :comment="comment"></applicationCommentItem>
```

Le composant parent inclut le composant enfant *applicationCommentItem* et lui passe la prop *comment*, qui dans ce cas est un objet.

```
<script>  
// Composant enfant  
export default {  
  name: 'application-comment-item',  
  props: {  
    comment: Object  
  }  
}  
// ...  
}
```

Le composant enfant définit les props qu'il attend de son parent, en respectant la bonne structure et les bons types de données.

Toujours utiliser key avec v-for

Un v-for permet d'effectuer une boucle for sur un élément. Afin d'éviter des problèmes de rafraîchissement et d'autres erreurs, chaque élément doit être identifié de manière unique, d'où l'attribut key.

```
<template>
  <ul id="exemple">
    <li v-for="todo in todos" :key="todo.id">{{ todo.text }}</li>
  </ul>
</template>

<script>
export default {
  name: 'exemple',
  data() {
    return {
      todos: [
        {
          id: 1,
          text: 'Todo 1'
        },
        {
          id: 2,
          text: 'Todo 2'
        }
      ]
    }
  }
}
</script>
```

Ici, le composant boucle sur un tableau d'objets *todos* et affiche sa propriété *text*, l'élément unique est la propriété *id* de chaque objet.

N'utilisez jamais v-if sur le même élément que v-for

Vue permet de réaliser des affichages conditionnels grâce à « v-if ».

```
<template>
  <div id="exemple">
    <div v-if="test === 4">Test vaut 4</div>
    <div v-else>Test vaut autre chose</div>
  </div>
</template>
...
```

```
...  
<script>  
export default {  
  name: 'exemple',  
  data() {  
    return {  
      test: 3  
    }  
  }  
}  
</script>
```

v-if et *v-for* ne doivent donc pas être utilisés sur le même élément, le *v-if* peut par exemple être assigné sur l'élément parent à la boucle.

Portée du style dans le composant

Le style défini au niveau App (composant racine) doit être global, mais tous les autres styles définis dans des composants doivent être avec une portée limitée au composant unique.

```
<style scoped>  
.example {  
  background-color: red;  
}  
</style>
```

L'attribut *scoped* permet de limiter la classe *example* à ce composant uniquement.

3.6 Gestion des erreurs

3.6.1 Backend

Lors de la réception de requêtes au niveau de l'API, une série de vérifications vont s'effectuer afin de garantir l'accès sécurisé aux ressources. Le diagramme suivant représente le déroulement de ces vérifications :

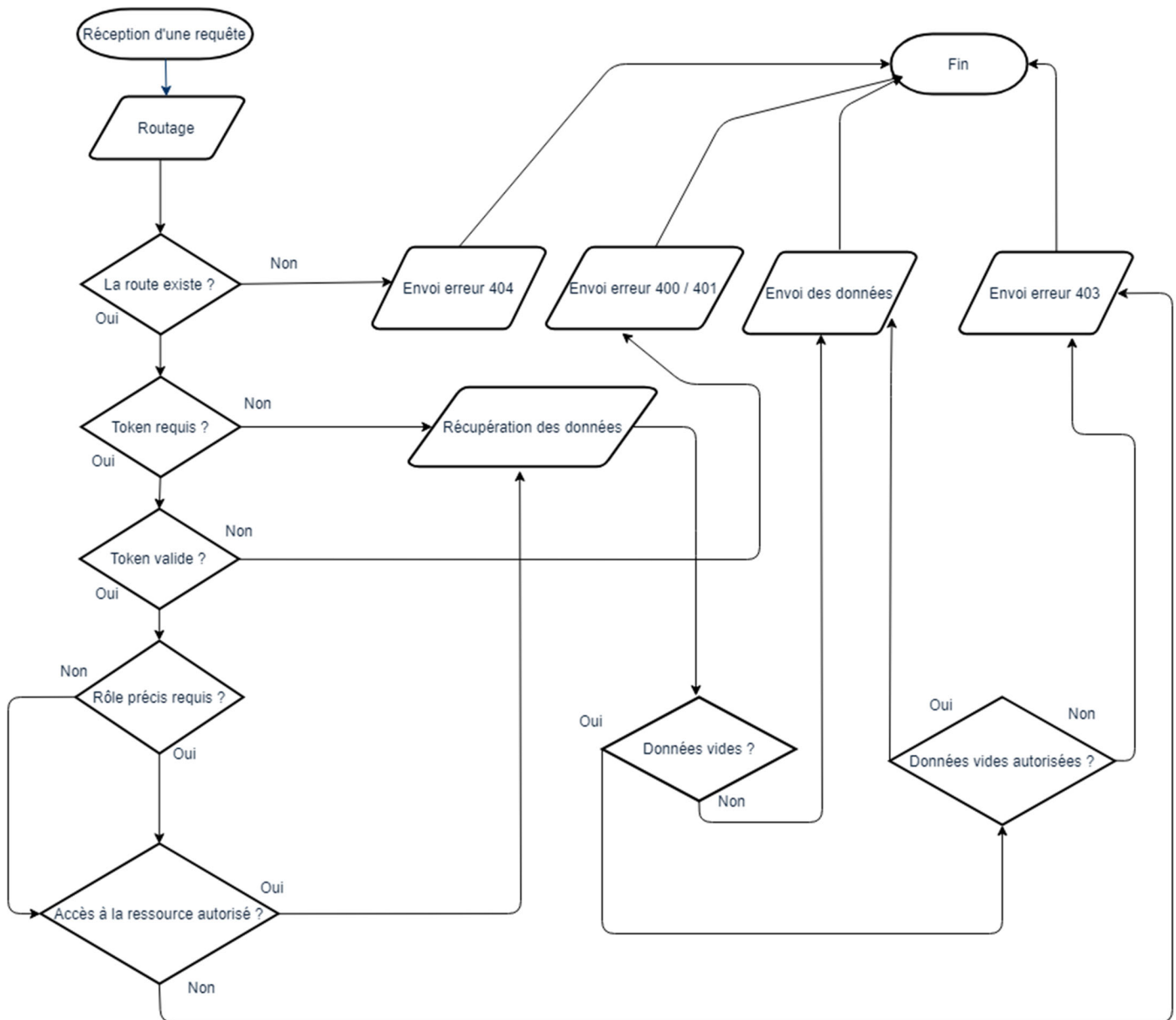


Figure 24 : Processus de validation, backend

Après cette validation, si la requête est un ajout (PUT), une édition (PATCH) ou une suppression (DELETE), les retours en cas d'erreur sont des codes HTTP différents (par exemple 422 en cas de données manquantes). Lors d'un ajout ou d'une modification, les données envoyées depuis l'interface sont validées avant d'être traitées (valeurs requises, type etc.).

Lumen utilise un **query builder** pour construire les requêtes SQL, par exemple pour récupérer tous les status de la table *status*, on procède de la manière suivante :

```
public function getAvailableStatus()
{
    return DB::table('status')->select()->get();
}
```

Lumen certifie que ce query builder protège contre les injections SQL, aucun code n'as donc besoin d'être ajouté pour échapper les variables, sauf dans le cas on l'on exécute une requête *raw* (c'est-à-dire en SQL pur). J'ai utilisé ce genre de requête une seule fois dans l'API :

```
return DB::table('applicant')
    ->select(DB::raw('job_short_value, job_full_value, count(*) as total'))
    ->join('position', 'applicant.fk_position', '=', 'position.position_id')
    ->join('job', 'position.fk_job', '=', 'job.job_id')
    ->groupBy('job_short_value')
    ->get();
```

Cependant, cette requête ne se base pas sur une valeur entrée par l'utilisateur pour s'exécuter, il n'y a donc pas de risque d'injection.

Comme expliqué auparavant, pour éviter les attaques XSS, il faut valider toute données envoyées depuis l'interface afin de ne pas accepter des données qui pourraient être dangereuses. Lumen offre des méthodes de validation qui, en plus de limiter les données non souhaitées, retourne par défaut des messages d'erreurs selon les noms de champs DB. J'ai cependant dû ajouter des fichiers de traductions pour traduire manuellement ces messages en français. Voici un exemple de validation (ajout d'un commentaire) :

```
$this->validate($this->request, [
    'content' => 'required|string',
    'is_private' => 'required|boolean',
    'applicant_id' => 'required|numeric'
], [lang::get('validation.required')]);
```

Les valeurs reçues doivent donc être non nulles et de types précis pour être acceptées. Le message de retour en cas d'erreur est ici traduit.

Lumen offre également d'autres filtres de validation prédéfinis¹³ (email, date, numérique, etc.).

¹³ Disponibles ici : <https://laravel.com/docs/5.1/validation#available-validation-rules>

3.6.2 Frontend

Lorsque l'interface envoie une requête et reçoit une réponse, elle vérifie si la requête à bien été effectuée avec succès, dans le cas contraire elle gère les erreurs selon le diagramme suivant :

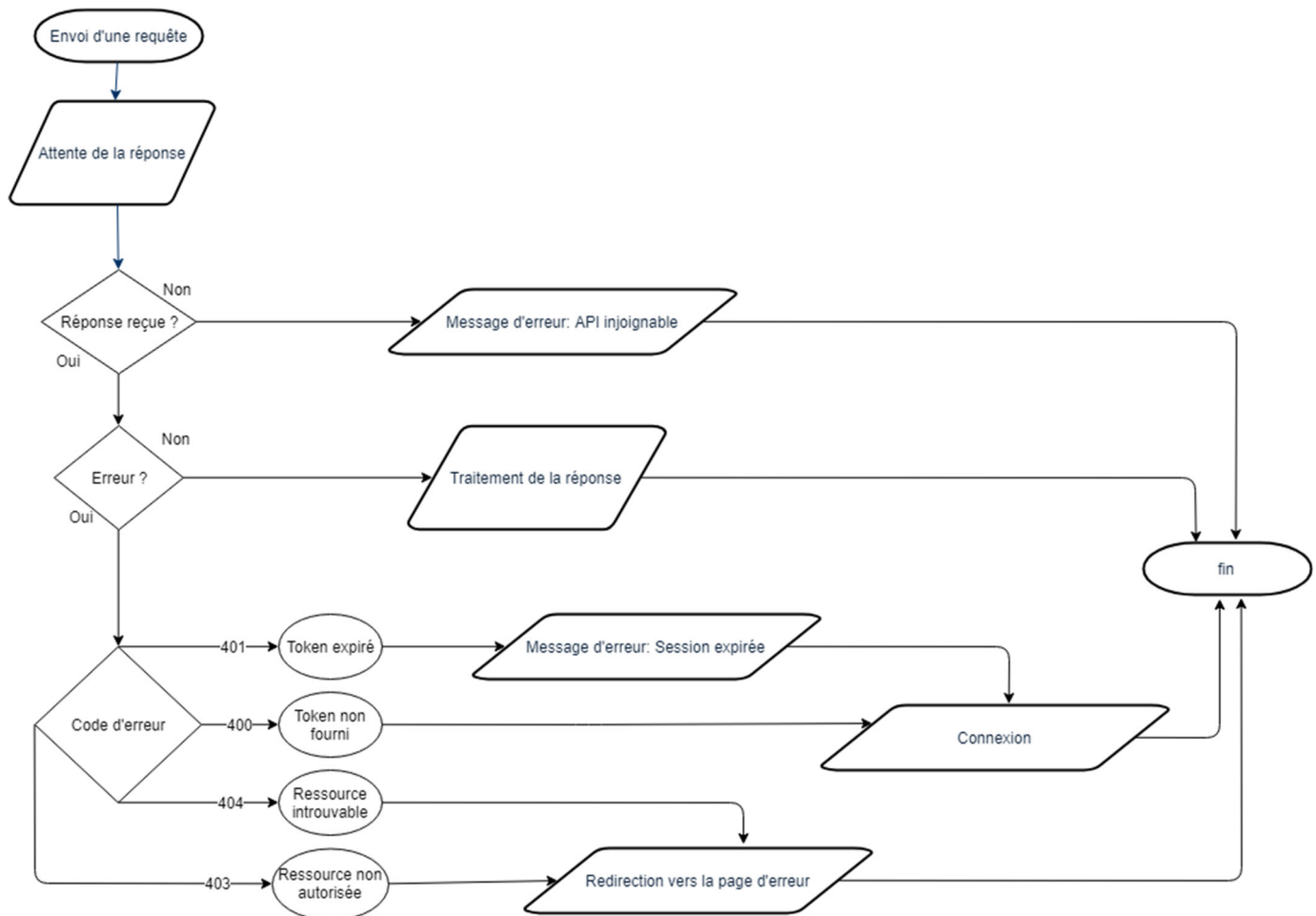


Figure 25 : Gestion des erreurs, frontend

Ce processus de vérification est effectué par Axios à l'aide d'un intercepteur, qui est exécuté après chaque réception de réponse:

```

instance.interceptors.response.use(function (response) {
  // Pas d'erreur, aucune action supplémentaire
  return response
}, function (error) {
  if (!error.response) {
    // API injoignable
  } else if (error.response.status == 400 || error.response.status == 401) {
    // Erreur de token (Expiré ou invalide)
  } else if (error.response.status == 403 || error.response.status == 404) {
    // Erreur 403 ou 404
  } else {
    // Affichage des erreurs (422)
  }
  return Promise.reject(error)
})

```

3.7 Tests

De manière globale, les tests effectués sont :

- Fonctionnement des requêtes API et vérification de leurs données ou valeurs de retour.
- Gestion des erreurs d'accès et d'autres types et envoi des statuts HTTP corrects (API).
- Gestion des erreurs et comportement correct selon les statuts HTTP retournés (interface).
- Fonctionnement correct et réactivité des composants lors des interactions avec l'interface.
- Compatibilité des navigateurs Chrome et Firefox

Au niveau du code, des tests automatisés n'ont pas été implémentés, car cela implique la connaissance des outils (en JavaScript, des modules comme Karma et Mocha permettant de faire des tests unitaires¹⁴) et évidemment du temps supplémentaire pour leur implémentation.

À la suite d'un problème sur la route de connexion (expliqué au point 4.4), je n'ai pas pu effectuer de tests sur le navigateur Firefox. Tous les tests décrits ont été effectués sur le navigateur Chrome. Cependant, les technologies utilisées (Vue.js et Vuetify) assurent la compatibilité de ces navigateurs¹⁵.

Pour tester les requêtes de l'API avant de devoir implémenter leurs utilisations dans l'API, j'utilise l'outil Postman¹⁶ qui permet d'effectuer des requêtes http vers l'API. Je fournis un token valide en configurant le header *Authorization* et choisis les données à envoyer si la requête le demande. La procédure générale de test pour une requête est la suivante :

Requête	Cas	Réponse attendue
GET	L'utilisateur n'a pas l'accès à la ressource	403 : Action non autorisée
GET	Ressource non existante	403 : Action non autorisée
GET	Pas d'erreur	Succès : Données demandées
PUT	Une ou plusieurs valeurs requises manquent à la requête	422 : Entité non utilisable
PUT	Toutes les données sont fournies, mais ne se respectent pas les critères de validations	422 : Entité non utilisable
PUT	Toutes les données sont fournies et validées avec succès	Succès : Élément ajouté

¹⁴ Voir <https://mochajs.org/> et <http://karma-runner.github.io/latest/>

¹⁵ Références <https://vuetifyjs.com/en/getting-started/quick-start> et <https://github.com/vuejs/vue#browser-compatibility>

¹⁶ Voir <https://www.getpostman.com/>

PATCH	Ressource non existante	403 : Action non autorisée
PATCH	L'utilisateur n'a pas l'accès à la ressource	403 : Action non autorisée
PATCH	Une ou plusieurs valeurs requises manque à la requête	422 : Entité non utilisable
PATCH	Toutes les données sont fournies, mais ne se respectent pas les critères de validations	422 : Entité non utilisable
PATCH	Toutes les données sont fournies et validées avec succès	Succès : Élément modifié
DELETE	L'utilisateur n'a pas l'accès à la ressource	403 : Action non autorisée
DELETE	Ressource non existante	403 : Action non autorisée
DELETE	La suppression est acceptée	Succès : Élément supprimé

Chaque type de requête a été testé pour chacun des cas attribués afin de valider son fonctionnement. L'implémentation dans l'interface a ensuite été implémentée en respectant les données demandées et les messages de retour gérés au niveau de l'interface.

Les composants de l'interface n'ont pas été testés en suivant une structure précise, mais à chaque implémentation de requête. Lors d'ajout, de modifications ou de suppressions, les données doivent se mettre à jour, cette mise à jour a également été testée.

4 Conclusion

4.1 Atteinte des objectifs

J'estime avoir pu prendre en compte tous les objectifs du cahier des charges ont été pris en compte et effectués lors du déroulement du projet.

Un seul des points techniques spécifiques n'a pas pu être totalement complété, il s'agit du point 4. *L'application fonctionne correctement sur Firefox et Chrome.* A cause d'un problème sur l'authentification (détaillé au point 4.4), je n'ai pas pu trouver une méthode qui corrige le problème à la fois sur Chrome et Firefox et je n'ai donc pas pu effectuer mes tests comme je le souhaitais.

La fonctionnalité permettant l'administration des places disponibles a été implémentée complètement mais je pense que l'interface pourrait être encore améliorée afin d'en simplifier l'utilisation.

4.2 Planification

En comparant la planification initiale avec le déroulement réel du projet, on constate que le projet a pu être réalisé dans les temps planifiés. Le temps que j'avais prévu pour chaque étape est globalement correct, mais certaines fonctionnalités m'ont demandé moins de temps à réaliser que prévu. Cela m'a permis de prendre de l'avance dans les fonctionnalités restantes et dans la rédaction de la documentation.

Je pense également que le temps que j'avais originellement pour la documentation n'était pas assez important, l'avance obtenue m'as donc permis de compenser ce manque de temps. J'ai également prendre deux heures supplémentaires pour réaliser les divers diagrammes de mon rapport et sa relecture.

Au niveau de l'ordre des étapes de la planification, j'ai respecté l'ordre prévu à l'exception des étapes de modifications du modèle de la base de données que j'ai effectué en une seule étape au lieu de plusieurs. Cela m'a également permis de prendre de l'avance sur le planning.

La planification initiale et la comparaison avec le déroulement réels se trouvent en annexes.

4.3 Améliorations futures

Bien que l'application propose déjà quelques fonctionnalités intéressantes pour faciliter le travail de sélection et de recrutement des apprentis, elle n'est pas encore assez complète pour être mise à disposition pour le moment. L'application doit être en adéquation avec les besoins des utilisateurs, c'est pourquoi les fonctionnalités ci-dessous pourraient être ajoutées à l'application afin d'offrir un outil complet et bien conçu aux responsables de la formation apprentis EPFL et des formateurs :

- Ajout d'une page « favoris » personnelle servant à retrouver facilement les candidatures commentées ou notée par un formateur.
- Ajout d'une fonctionnalité de comparaison de plusieurs candidatures simultanément ou de pouvoir ouvrir plusieurs candidatures en même temps sans devoir retourner dans la liste de candidatures (système d'onglets par exemple).
- Ajout d'envoi de réponse prédéfinies en cas de refus, données incomplètes, etc.
- Ajout d'envoi mail directement dans l'interface pour le contact des candidats et éventuellement garder un historique des échanges.
- Ajout de notifications par mail lors de nouvelles candidatures.
- Le formulaire de candidature devra être adapté à l'utilisation des places d'apprentissage ouvertes ajoutées lors du projet.

De nouveaux éléments ou des modifications peuvent également être apportées aux fonctionnalités actuelles :

- Le téléchargement multiple des annexes pourrait se faire dans un dossier compressé (.zip). L'export complet du dossier (annexes + données) avec une seule opération devrait également être ajouté (actuellement disponible mais séparément). L'ajout de formats d'export supplémentaires (csv, etc.) seraient également à prévoir.
- Un graphique pourrait être ajouté sur la page d'accueil afin de représenter le nombre de candidatures sur un plage de temps données. La requête API pour cette fonctionnalité est déjà disponible.
- Comme expliqué plus tôt, la page d'administration devra être réadaptée pour rendre son utilisation plus facile.

4.4 Problèmes rencontrés

Des erreurs multiples s'affichent du côté client lors du rendu de certains composants Vue.js

Lorsque des données sont nécessaires au rendu d'un composant Vue.js et que ces données doivent être récupérées depuis le retour d'une requête vers l'API, il se peut que la requête ne soit pas encore terminée lors du rendu et donc que les données soient vides, provoquant des erreurs et l'échec du rendu.

Solution : Pour éviter ces erreurs, il faut donc attendre que la requête vers l'API soit terminée avant d'effectuer des actions sur ces données. Dans ce but intervient l'utilisation des **promesses** JavaScript (Promises en anglais). Etant donné que le langage peut fonctionner tant de manière synchrone qu'asynchrone, les promesses servent à effectuer des actions uniquement lorsque l'action précédente est terminée. Voici un exemple de promesse :

```
function getUserData(context) {  
  return new Promise((resolve, reject) => {  
    axios({  
      method: 'get',  
      url: '/user'  
    })  
    .then(response => {  
      resolve(response.data)  
    })  
    .catch(err => {  
      reject(err)  
    })  
  })  
}
```

Ici, on instancie une promesse qui effectue une requête Axios (qui est d'ailleurs elle-même une promesse). Lors que la requête est terminée, elle retourne des données en cas de succès (resolve), sinon une erreur en cas d'échec (reject).

L'appel de cette fonction se fait ensuite de la manière suivante :

```
getUserData().then(data => {  
  // Succès de la Promise  
})  
.catch(error => {  
  // Echec de la Promise  
})
```

En cas de succès, la fonction `then()` est exécutée, sinon on passe dans la fonction `catch()`.

Erreur SQL contrainte lors de la suppression de candidatures

Lors de la suppression de candidatures, certaines requêtes retournent une erreur SQL Constraint. Il s'agit simplement des candidatures auxquelles des commentaires ont été attribués.

Solution : Il suffit de modifier la contrainte `fk_applicant_id` de la table `comment` pour lui attribuer l'action SQL `ON DELETE CASCADE`, ce qui va supprimer les commentaires liés à une candidature lorsque celle-ci est supprimée.

Authentification – CORS

La requête de connexion (`/api/auth/login`) retourne une erreur :

Access to XMLHttpRequest at 'http://localhost:8000/api/auth/login' from origin 'https://canap-gest-dev.local:8080' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

Il s'agit en fait d'une alerte de sécurité : la *CORS policy* citée proviens d'un processus de vérification appelé Cross-origin resource sharing (**CORS**), il s'agit d'une limitation dans laquelle un service web peut interagir avec un élément d'un domaine différent du sien.

Dès qu'une requête HTTP n'est pas de type GET ou POST et qu'elle contient des headers personnalisés (dans mon cas, le header *Authorization* du token) une validation CORS s'effectue :

- Le client envoie une requête de type OPTIONS afin de demander les types de requêtes et de headers autorisés par le serveur.
- Le serveur, s'il est bien configuré, répond avec ses exigences.
- Si les headers et la méthode sont autorisés, la requête peut s'effectuer.

L'ajout d'un **middleware** (élément placé entre la requête reçue par le serveur et la réponse renvoyée) configurant la réponse à cette requête OPTION résout ces erreurs.

Dans mon application ce middleware est écrit de la manière suivante :

```
<?php
namespace App\Http\Middleware;

use Closure;

class CorsMiddleware
{
    public function handle($request, Closure $next)
    {
        $headers = [
            'Access-Control-Allow-Origin'      => '*',
            'Access-Control-Allow-Methods'    => 'GET, POST, PUT, PATCH,
DELETE, OPTIONS',
            'Access-Control-Allow-Credentials' => 'true',
            'Access-Control-Max-Age'          => '86400',
            'Access-Control-Allow-Headers'    => 'Content-Type, Accept,
Authorization, X-Requested-With, Application'
        ];

        if ($request->isMethod('OPTIONS'))
        {
            return response()->json('{"method":"OPTIONS"}', 200, $headers);
        }

        $response = $next($request);
        foreach($headers as $key => $value)
        {
            $response->headers->set($key, $value);
        }

        return $response;
    }
}
```

Ce middleware définit les origines, les méthodes et les headers autorisés. Ici toutes les origines sont autorisées, les méthodes GET, POST, PUT, PATCH, DELETE et OPTIONS ainsi qu'une série de headers sont configurés comme acceptables.

Cependant cette opération corrige toutes les requêtes, sauf celle de connexion car celle-ci effectue une redirection supplémentaire pour la connexion à travers Tequila qui ne gère elle pas CORS au niveau de son serveur. Tequila ne semble pas être totalement compatible avec une utilisation stateless.

Solution : Je n'ai pas trouvé de solution dans le cadre du projet qui fonctionnait sur tous les navigateurs. Afin de ne pas être bloqué dans le travail, j'ai désactivé cette vérification sur Chrome¹⁷. L'application est fonctionnelle si l'on désactive cette vérification, évidemment cela n'est pas une solution envisageable, il faudra donc trouver une autre méthode de fonctionnement pour effectuer l'authentification.

4.5 Ressenti personnel

J'ai le sentiment d'avoir accompli avec succès les éléments de ce projet. J'ai travaillé à un rythme très productif tout au long du déroulement afin de pouvoir livrer un travail que j'estime au niveau de mes compétences dans ce type de projet de développement web. J'ai souhaité réaliser une application web la plus fonctionnelle, intuitive et esthétique que possible en fonction du temps à disposition ainsi qu'une documentation exploitant la totalité des éléments du projet.

Mes compétences professionnelles et mon expérience dans le développement web que j'ai acquises lors de ma formation dans mon entreprise m'ont été très utiles pour pouvoir réaliser ce projet que j'estime très ambitieux et complet. Mon expérience avec Vue.js m'a permis d'avancer vite dans le développement et de pouvoir contourner les problèmes avant même de les avoir rencontrés, notamment en utilisant les bonnes pratiques apprises auparavant.

Ce projet m'a donné l'occasion de réaliser un projet conséquent dans un temps strict, planifié et structuré. Dans le cadre de mon entreprise formatrice, l'aspect de planification n'était pas vraiment présent lors de mes divers projets que j'ai réalisés au cours de ma formation. Cela m'a donc appris à me situer sur une plage de temps définie, à estimer mon temps de travail sur chaque fonctionnalité et à m'adapter aux imprévus. J'ai effectué ce travail avec motivation et plaisir et ressort pleinement satisfait de mon travail.

¹⁷ Voir <http://testingfreak.com/how-to-fix-cross-origin-request-security-cors-error-in-firefox-chrome-and-ie/>

5 Glossaire

A

API

application programming interface, interface permettant de fournir des services à d'autres applications..... 4

B

base64

Algorithme de chiffage 22

C

Canap-Gest

Nom donné au projet, abréviation de "Candidature Apprentissage - Gestion" 2

CLI

Commande Line Interface 13

commit

Nom pour les versions git 15

CORS

Cross-origin resource sharing..... 50

D

DB

Abréviation anglaise de base de données..... 4

DOM

Document Object Model..... 35

E

ECMAScript

Normes JavaScript en matière de code..... 14

ES5

ECMAScript v.5 14

ES6

ECMAScript v.6..... 14

ESLint

Linteur JavaScript 14

G

Git

Logiciel de gestion de version 11

H

hash

Signature unique d'un élément..... 15

HCSHA256

Algorithme de chiffage 22

HS256

Algorithme de chiffage 21

I**IdP**

Identity Provider 20

injection SQLExploitation de failles dans les requêtes SQL 8

J**JSON**

Format de données dérivé de la syntaxe des objets JavaScript 2

JWTJSON Web Token 12

L**LDAP**

Lightweight Directory Access Protocol, interaction avec les services d'annuaire 20

linter

Valdateur de code 14

localeParamètre qui définit la langue et le format des dates de l'utilisateur 38

M**master**

Branche git principale 15

middleware

Élément placé entre la requête reçue par le serveur et la réponse renvoyée 50

mutationsActions qui modifient le store 36

N**Node.js**

Environnement JavaScript côté serveur 12

NPMNode.js Packet Manager 12

O**opensource**Code source ouvert 20

Q**query builder**Classe et méthodes qui permettent d'effectuer des requêtes vers une base de données 43

R**REST**

Structure de fonctionnement sans état 19

RGPDRèglement général sur la protection des données 4

S**SPA**

Single Page App, application dynamique coté client monospace 35

SSO

Single sign-on 20

stateless

Service web sans état 19

store

État centralisé 36

T**Tequila**

Service d'authentification EPFL 19

timestamp

Nombre de secondes écoulées depuis le 1er janvier 1970 à minuit UTC 22

transcompilateur

Converti un code source d'un langage vers un autre 14

U**UI**

Interface utilisateur 10

UX

Expérience utilisateur 10

V**VCS**

Version Control System 11, 15

verbs

Type de requête http 20

X**XHR**

XMLHttpRequest, objet d'interaction client-serveur 15

XSS

Cross-site scripting 8

6 Annexes

6.1 Sources

Sources de documentations utilisées dans le cadre du projet

Source	Auteur	Utilisation
https://vuejs.org/	Evan You et communauté	Documentation Vue.js officielle
https://github.com/axios/axios	axios	Documentation Axios officielle
https://vuetifyjs.com/en/	Vuetify, LLC	Documentation Vuetify officielle
https://jwt.io/	auth0	Documentation du standard JWT
https://github.com/firebase/php-jwt	Firebase / Google	Documentation firebase/php-jwt officielle
https://vuex.vuejs.org/fr/	Evan You et communauté	Documentation Vuex officielle
https://router.vuejs.org/	Evan You et communauté	Documentation Vue Router officielle
https://php.net/	The PHP Group	Documentation PHP officielle
https://lumen.laravel.com/	Laravel	Documentation Lumen officielle
https://laravel.com	Laravel	Documentation Laravel officielle
https://fr.wikipedia.org/wiki/Git	Collaboratif	Documentation sur git
http://blog.pilotsystems.net/2012/septembre/les-api-rest	Yohann Gabory	Article sur les API REST
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Promise	MDN	Promise JavaScript
https://medium.com/@fishpercolator/implementing-a-global-snackbar-in-nuxt-js-vuetify-using-vuex-a92b78e5651b	Fish Percolator	Inspiration pour la gestion des erreurs avec un composant global
https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP	Collaboratif	Liste des codes HTTP
https://en.wikipedia.org/wiki/Identity_provider	Collaboratif	Explication Single sign-on

https://en.wikipedia.org/wiki/Identity_provider	Collaboratif	Explication Identity provider
http://testingfreak.com/how-to-fix-cross-origin-request-security-cors-error-in-firefox-chrome-and-ie/	Naveen	Désactivation alertes CORS

Sources d'installation de dépendances

Source	Auteur	Utilisation
https://nodejs.org/en/	Node.js Foundation	Téléchargement Node.js
https://cli.vuejs.org/guide/	Evan You et communauté	Installation @vue/cli
http://tequila.epfl.ch/	EPFL Lausanne	Client Tequila EPFL
https://git-scm.com/	git	git
http://www.wampserver.com/	WAMP Server	WAMP Server
https://getcomposer.org/	Nils Adermann, Jordi Boggiano et communauté	Composer

Autres services utilisés

Source	Utilisation
https://mockflow.com/	Shémas API etc.
https://trello.com/	Suivi des tâches
https://www.draw.io/	Diagrammes

6.2 Compte-rendu des discussions

Discussion initiale, le 16.04.2019

Initialisation du TPI, lecture du cahier des charges et confirmation, questions au chef de projet.

Les documents (rapports et code source) devront être envoyé aux deux experts et au chef de projet deux fois par semaine, au moyen d'un mail et d'un lien vers le repository c4science. La planification initiale doit être envoyé à l'issue du premier jour.

Contact des experts, le 16.04.2019 et 17.04.2019

Contact des experts pour leur demander la création d'un compte puis pour confirmer l'attribution des accès au repository C4science.

Discussion seconde visite de l'experte, le 26.04.2019

Discussion sur l'avancée du projet.

Il faudrait ajouter le Modèle Conceptuel de la base de données au rapport

Discussion chef de projet, le 26.04.2019

Discussion sur l'avancée du projet et sur mon point de vue sur l'approche des futures fonctionnalités, petite démonstration de l'application.

Dans l'interface, il faut appliquer deux petites modifications : afficher plus de candidatures à la fois (pagination) et les trier par défaut de la plus récente à la plus ancienne.

Contact des experts, le 01.05.2019

Contact des experts pour raison de maladie, les jours de travail du 01.05 et 02.05 sont reportés respectivement au 09.05 et 14.05.

6.3 Code source

L'entier du code source se trouve sur le repository c4science disponible ici : <https://c4science.ch/source/canapgest/>

Pour disposer du code source en local, il suffit d'exécuter la commande suivante :

```
C:\..\ > git clone https://c4science.ch/source/canapgest.git
```

6.4 Autres

- Documentation administrateur
- Planification initiale du projet
- Déroulement réel du projet
- Journal de travail
- Résumé succinct du projet
- Cahier des charges

Pour des raisons de lisibilité, la planification et le déroulement sont séparés en deux parties.

Annexes

- Documentation administrateur
- Planification initiale du projet
- Déroulement réel du projet
- Journal de travail
- Résumé succinct du projet
- Cahier des charges



Doc. Admin. – TPI Canap-Gest

« CANAP-GEST » APPLICATION WEB DE GESTION DE
CANDIDATURES D'APPRENTISSAGE EPFL –
DOCUMENTATION ADMINISTRATEUR

EPFL | Nicolas Crausaz

CHEF DE PROJET | NICOLAS DUBOIS
EXPERTS | B. FOLOMIETOW, F. ANDOLFATTO

Sommaire

SOMMAIRE	1
1 INSTALLATION	2
1.1 DÉPENDANCES EN DÉVELOPPEMENT	2
1.2 DÉPENDANCES EN PRODUCTION	2
2 CONFIGURATION	2
2.1 ENVIRONNEMENT DE DÉVELOPPEMENT	2
2.2 PRÉPARATION À LA PRODUCTION	4
2.3 STRUCTURE	5
2.4 FICHIERS	5
2.5 BASE DE DONNÉES	7

1 Installation

1.1 Dépendances en développement

L'environnement de développement pour faire tourner l'application nécessite les dépendances suivantes :

- WampServer (avec PHP 7.2) (<http://www.wampserver.com/>)
- Composer (<https://getcomposer.org/>)
- Node.js (<https://nodejs.org/en/>)
- @vue/cli (<https://cli.vuejs.org/>)
- Git (<https://git-scm.com/>)

1.2 Dépendances en production

Le serveur de production devra être équipé de :

- Service Web (IIS, apache etc.)
- PHP 7.2
- Composer

2 Configuration

2.1 Environnement de développement

Pour commencer, clonez la dernière version stable du projet en local :

```
C:\..\> git clone https://c4science.ch/source/canapgest.git
```

Pour faire tourner l'application localement dans le cadre du développement, il faut établir quelques configurations. Pour commencer il faut créer la base de données, pour ce faire il suffit d'exécuter le script `create_db_script.sql` qui se trouve en annexe, puis d'exécuter la requête suivante, en modifiant d'abord la valeur `<password>` :

```
USE canap_db;  
CREATE USER 'canap-gest-user'@'localhost' IDENTIFIED BY '<password>';  
GRANT SELECT, UPDATE, INSERT, DELETE ON * TO 'canap-gest-user'@'localhost';
```

L'utilisateur `canap-gest-user` dispose uniquement des opérations dont il a besoin pour les interactions avec la DB.

Au niveau de l'API, renommer le fichier `_API/.env.example` en `.env`, puis remplir les champs ainsi :

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=<chaîne de caractère aléatoire>
APP_TIMEZONE=UTC

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=canap_db
DB_USERNAME=canap-gest-user
DB_PASSWORD=<mot de passe précédemment choisi>

CACHE_DRIVER=file
QUEUE_DRIVER=sync

JWT_SECRET==<chaîne de caractère aléatoire>
```

La configuration de l'application vue est optionnelle. L'application a originellement été développée en utilisant la configuration suivante, dans le fichier `Site/vue.config.js`

```
module.exports = {
  devServer: {
    host: 'canap-gest-dev.local',
    port: 8080,
    https: true
  }
}
```

Pour utiliser un nom d'hôte comme ici, il faut éditer la configuration DNS locale (fichier `host`) se trouvant dans `C:\Windows\System32\drivers\etc\hosts` et y ajouter les lignes suivantes :

```
127.0.0.1 canap-gest-dev.local
::1 canap-gest-dev.local
```

Pour démarrer le développement Vue.js :

```
C:\..\> cd Site
C:\..\Site> npm install
C:\..\Site> npm run serve
```


Installer ensuite les dépendances de l'API et la démarrer :

```
C:\..\> cd API
C:\..\Site> composer install
C:\..\Site> php -S localhost:8000 -t public
```

2.2 Préparation à la production

Afin de préparer la production, certains éléments définis lors du développement doivent être modifiés.

Pour commencer, clonez la dernière version stable du projet en local :

```
C:\..\> git clone https://c4science.ch/source/canapgest.git
```

Au niveau de l'interface, il suffit uniquement de modifier la configuration de Axios afin d'utiliser l'URL en production de l'API :

Il faut donc éditer le fichier `Site/src/plugins/axios.js`, et de modifier l'instance Axios ainsi :

```
const instance = axios.create({
  headers: { 'Authorization': "Bearer " + localStorage.getItem('stored_token') },
  baseURL: 'https://canap-gest.epfl.ch/api'
})
```

Pour finir, optimiser et générer l'application Vue.js pour la production :

```
C:\..\> cd Site
C:\..\Site> npm install
C:\..\Site> npm run build
```

Le résultat se trouve comme indiqué dans le dossier « dist ».

Au niveau de l'API il faut modifier plusieurs éléments : tout d'abord, modifier l'URL de redirection Tequila. Pour ce faire, éditer le fichier `API/app/http/Controllers/AuthController.php` en remplaçant

```
$oClient->SetApplicationURL('http://canap-gest-dev.local:8080');
```

par

```
$oClient->SetApplicationURL('https://canap-gest.epfl.ch');
```

Renommer ensuite le fichier `_API/.env.example` en `.env`, puis remplir les champs ainsi :

```
APP_ENV=local
APP_DEBUG=false
APP_KEY=<chaîne de caractère aléatoire>
APP_TIMEZONE=UTC

DB_CONNECTION=mysql
DB_HOST=mysql-fac.epfl.ch
DB_PORT=33003
DB_DATABASE=canap_db
DB_USERNAME=<username>
DB_PASSWORD=<password>

CACHE_DRIVER=file
QUEUE_DRIVER=sync

JWT_SECRET==<chaîne de caractère aléatoire>
```

Les informations de connexion MySQL se trouvent dans le KeePass ENAC-IT2.

Installer ensuite les dépendances de l'API :

```
C:\..\> cd API
C:\..\Site> composer install
```

2.3 Structure

La structure du site, depuis sa racine web doit être structuré de cette manière :

- `api`
- `www`

Le dossier `api` contient donc le contenu du dossier `api`, le `www` contient les fichiers obtenus après avoir effectué le *build* de l'application Vue.js (dossier `dist`).

Le dossier `www` doit être configuré en tant que destination par défaut.

2.4 Fichiers

Il faut ensuite ajouter quelques fichiers de configuration pour assurer le bon fonctionnement de l'application. Les exemples suivants sont valables pour une utilisation de IIS en tant que service Web.

Ajouter à la racine du dossier `api`, un fichier `web.config` contenant :

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <handlers>
      <remove name="PHP_7.2" />
      <add name="PHP_7.2" path="*.php"
verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" type=""
modules="FastCgiModule" scriptProcessor="C:\Program Files (x86)\PHP\v7.2\php-
cgi.exe" resourceType="Either" requireAccess="Script" allowPathInfo="false"
preCondition="" responseBufferLimit="4194304" />
    </handlers>
    <rewrite>
      <rules>
        <rule name="Silex Front Controller" stopProcessing="true">
          <match url="^(.*)$" ignoreCase="false" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile"
ignoreCase="false" negate="true" />
          </conditions>
          <action type="Rewrite" url="/api/public/index.php"
appendQueryString="true" />
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

Cette configuration va rediriger toutes les requêtes API vers le routeur Lumen.

Ajouter ensuite à la racine du dossier `www`, un fichier `web.config` contenant :

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <modules runAllManagedModulesForAllRequests="true">
      <remove name="WebDAVModule" />
    </modules>
    <defaultDocument enabled="true">
      <files>
        <clear />
        <add value="index.html" />
      </files>
    </defaultDocument>
    <rewrite>
      <rules>
        <clear />
        <rule name="Redirect to https" stopProcessing="true">
          <match url=".*" />
          <conditions>
```

```
        <add input="{HTTPS}" pattern="off" ignoreCase="true"
/>
        </conditions>
        <action type="Redirect"
url="https://{HTTP_HOST}:8443{REQUEST_URI}" redirectType="Permanent"
appendQueryString="false" />
    </rule>
</rules>
</rewrite>
<handlers>
    <remove name="WebDAV" />
    <add name="PHP_7.2" path="*.php" verb="GET,HEAD,POST" type=""
modules="FastCgiModule" scriptProcessor="C:\Program Files (x86)\PHP\v7.2\php-
cgi.exe" resourceType="Either" requireAccess="Script" allowPathInfo="false"
preCondition="" responseBufferLimit="4194304" />
</handlers>
</system.webServer>
</configuration>
```

Cette deuxième configuration permet de servir l'application si l'URL n'est pas une requête vers l'API.

2.5 Base de données

La base de données en production se trouve sur l'infrastructure MySQL EPFL (*mysql-fac.epfl.ch*). Les informations de connexion se trouvent dans le KeePass ENAC-IT2. Seul le serveur web enacit2web1.epfl.ch est autorisé à se connecter à ce serveur MySQL.

Pour créer la base de données, se connecter à *mysql-fac.epfl.ch*, puis exécuter le script *create_db_script.sql* qui se trouve en annexe.

Planification initiale - partie 1

Nicolas Crausaz
Canap-Gest

Date de début	16.04.2019
Date de fin	08.05.2019

		Date de début	Date de fin	2019-10-05	Avril																																
			08.05.2019		16				17				18					23				24				25				26		30					
TÂCHE					08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30		08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30	8h00-10h		08h00 - 10h00	10h00 - 12h00	13h00 - 15h00			
Initialisation et analyse																																					
Première séance, introduction TPI																																					
Relecture du cahier des charges																																					
Création du repository Git																																					
Réalisation de la planification initiale																																					
Création de templates (rapport, manuel, journal)																																					
Rédaction du rapport avec les informations déjà acquises																																					
Rédaction du rapport et journal																																					
Envoi des documents																																					
Deuxième réunion experts																																					
Réalisation																																					
Génération de la structure de base du site																																					
Mise en place de l'API en local																																					
Mise en place de la charte graphique et des contraintes EPFL																																					
Intégration de l'authentification																																					
Tests																																					
Réalisation de la structure de l'interface « responsable »																																					
Réalisation de la structure de l'interface « formateur »																																					
Réalisation de l'administration des lieux et des corps de métier (responsables)																																					
Modification du modèle (DB)																																					
Réadapter l'API suite au changement de modèle																																					
Ajout des requêtes à l'API																																					
Implémentation dans l'interface																																					
Tests																																					
Réalisation des fonctionnalités de visualisation et actions (responsables)																																					
Changement des statuts (valide, incomplet etc.)																																					
Refus / rejet des postulations																																					
Tests																																					
Réalisation des fonctionnalités de visualisation (formateurs)																																					
Visualisation globale																																					
Visualisation individuelle (postulation unique)																																					
Visualisation des fichiers annexes																																					
Tests																																					
Réalisation de la fonctionnalité d'attribution de notes (formateurs)																																					
Modification du modèle (DB)																																					
Ajout des requêtes à l'API																																					
Implémentation dans l'interface																																					
Tests																																					
Réalisation de l'affichage des nouvelles candidatures																																					
Réalisation de la fonctionnalité d'attribution de commentaires																																					
Réalisation de la documentation administrateur																																					
Finalisation																																					
Tests globaux																																					
Recherche sur les erreurs restantes																																					
Relecture du rapport																																					
Mise en lien des documents																																					
Envoi final des documents																																					

Déroulement du projet - partie 1

Nicolas Crausaz
Canap-Gest

Date de début	16.04.2019
Date de fin	14.05.2019

		Date de fin	14.05.2019	Avril																																1	2					
				16				17				18								23				24				25				26					30				1	2
TÂCHE				08h00 - 10h00	10h00 - 12h00	12h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	12h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	12h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	12h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	12h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	12h00 - 15h00	15h00-17h30	8h00-10h		08h00 - 10h00	10h00 - 12h00	12h00 - 15h00	15h00-17h30									
Initialisation et analyse				x	x	x	x																																			
Première séance, introduction TPI				x																																						
Relecture du cahier des charges				x																																						
Création du repository Git				x																																						
Réalisation de la planification initiale				x	x	x																																				
Création de templates (rapport, manuel, journal)						x																																				
Rédation du rapport avec les informations déjà acquises						x	x																																			
Rédaction du rapport et journal							x			x				x															x						x							
Envoi des documents							x							x														x							x							
Deuxième réunion experts																												x														
Réalisation																																										
Génération de la structure de base du site						x																																				
Mise en place de l'API en local						x																																				
Mise en place de la charte graphique et des contraintes EPFL						x																																				
Intégration de l'authentification							x	x																																		
Tests								x	x																																	
Réalisation de la structure de l'interface « responsable »									x	x	x																															
Réalisation de la structure de l'interface « formateur »									x	x	x																															
Réalisation de l'administration des lieux et des corps de métier (responsables)												x	x	x	x	x	x	x																								
Modification du modèle (DB)												x																														
Réadapter l'API suite au changement de modèle													x	x	x																											
Ajout des requêtes à l'API																			x	x	x																					
Implémentation dans l'interface																			x	x	x																					
Tests																						x	x																			
Réalisation des fonctionnalités de visualisation et actions (responsables)																x	x																									
Changement des statuts (valide, incomplet etc.)																x	x																									
Refus / rejet des postulations																		x																								
Tests																		x																								
Réalisation des fonctionnalités de visualisation (formateurs)																		x	x	x					x	x	x	x				x	x									
Visualisation globale																		x	x																							
Visualisation individuelle (postulation unique)																				x																						
Visualisation des fichiers annexes																																										
Tests																																										
Réalisation de la fonctionnalité d'attribution de notes (formateurs)												x		x	x																			x								
Modification du modèle (DB)												x																														
Ajout des requêtes à l'API														x	x																											
Implémentation dans l'interface																																										
Tests																																										
Réalisation de l'affichage des nouvelles candidatures																																		x								
Réalisation de la fonctionnalité d'attribution de commentaires																																										
Réalisation de la documentation administrateur																																										
Finalisation																																										
Tests globaux																																										
Recherche sur les erreurs restantes																																										
Relecture du rapport																																										
Mise en lien des documents																																										
Envoi final des documents																																										

x = réalisation de la tâche

Déroulement du projet - partie 2

Nicolas Crausaz
Canap-Gest

Date de début	16.04.2019
Date de fin	14.05.2019

TÂCHE	Mai															
	7				8				9				14			
	08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30	08h00 - 10h00	10h00 - 12h00	13h00 - 15h00	15h00-17h30
Initialisation et analyse																
Première séance, introduction TPI																
Relecture du cahier des charges																
Création du repository Git																
Réalisation de la planification initiale																
Création de templates (rapport, manuel, journal)																
Rédaction du rapport avec les informations déjà acquises																
Rédaction du rapport et journal				x				x			x	x	x	x	x	x
Envoi des documents				x								x				x
Deuxième réunion experts																
Réalisation																
Génération de la structure de base du site																
Mise en place de l'API en local																
Mise en place de la charte graphique et des contraintes EPFL																
Intégration de l'authentification																
Tests																
Réalisation de la structure de l'interface « responsable »																
Réalisation de la structure de l'interface « formateur »																
Réalisation de l'administration des lieux et des corps de métier (responsables)																
Modification du modèle (DB)																
Réadapter l'API suite au changement de modèle																
Ajout des requêtes à l'API																
Implémentation dans l'interface																
Tests																
Réalisation des fonctionnalités de visualisation et actions (responsables)																
Changement des statuts (valide, incomplet etc.)																
Refus / rejet des postulations																
Tests																
Réalisation des fonctionnalités de visualisation (formateurs)																
Visualisation globale																
Visualisation individuelle (postulation unique)																
Visualisation des fichiers annexes																
Tests																
Réalisation de la fonctionnalité d'attribution de notes (formateurs)																
Modification du modèle (DB)																
Ajout des requêtes à l'API																
Implémentation dans l'interface																
Tests																
Réalisation de l'affichage des nouvelles candidatures	x	x														
Réalisation de la fonctionnalité d'attribution de commentaires			x	x	x	x										
Réalisation de la documentation administrateur							x	x	x	x						
Finalisation							x	x	x	x	x	x	x	x	x	x
Tests globaux							x	x	x	x	x					
Recherche sur les erreurs restantes								x		x	x					
Relecture du rapport													x		x	
Mise en lien des documents														x	x	
Envoi final des documents																x

x = réalisation de la tâche

TPI - Canap Gest		Journal de travail	Nicolas Crausaz
------------------	--	--------------------	-----------------

Date	Etape - Tâche	Description	Remarques
Mardi 16.04	Initialisation et analyse	8h-8h45: Réunion intro TPI 8h45-13h: Planification initiale, création repo. GIT, création des templates documents 13h-16h: Base du site, charte, API en local 16h- 17h15: Rapport 17h15-17h30: Contact experts et envoi documents	Le repo c4science affiche des erreurs serveur, cela semble être une perturbation globale.
Mercredi 17.04	Réalisation	8h-10h: Intégration authentification 10h-14h: Protection des routes, strcture du site version formateur et responsable, gestion des erreurs 14h-16h30: Ajouts de valeurs de tests en db, affichage des candidatures sur l'interface, changement de la locale pour affichage en français 16h30-17h30: Rédaction du rapport, correction planification (date de rdv erronée)	J'ai contacté le support pour réparer les erreurs sur c4science, c'est ok. L'authentification n'est pas encore fonctionnelle a 100%, probleme CORS La gestion des erreurs n'est pas terminée
Jeudi 18.04	Réalisation	8h-10h15 : Modifications de la DB pour les nouvelles fonctionnalités et insertions de valeurs de tests 10h15- 12h: Modification de l'API suite au changement du modèle DB, au niveau de l'authentification et de la vérification des rôles de l'utilisateur. 13h-16h40 : Modification de l'API suite au changement du modèle DB, à tout les niveaux. Ajouts du controller status 16h45-17h30: Rapport et envoi des documents	Erreur restante dans le controleur fichier (Call to undefined method Symfony\Component\HttpFoundation\BinaryFileResponse::header())
Mardi 23.04	Réalisation	8h-13h45 : Ajouts des requetes API pour les places disponibles et implémentation dans l'interface 13h45-14h15: Rapport 14h15- 15h30: Adaptation de requetes dans l'API (réadaptation de structure et doublon) 15h30-17h: Gestion erreurs, corrections globales 17h-17h30: Rapport, envoi documents	Reste à afficher les erreurs ou success des actions
Mercredi 24.04	Réalisation	8h-8h30: Adaptation requetes postulotions, renvoi données selon role et status 8h30-10h25: Implémentation de la modification des status sur l'interface, édition unique et groupée des statut de candidatures, implémentation de la suppression de candidatures 10h25-11h: Tests 11h-12h00: Vue candidatures version formateur, filtre métier et status (resp), getter global pour le role (vérif si responsable), création de composants pour améliorer la structure de la vue et spliter le code, utilisation des getters vueux 13h-13h45: Affichage des erreurs (interface) 13h45-15h30: Vue postulation unique, mise à jour requete candidature API pour récupérer plus d'infos 15h45-17h30: Rapport	Affichage des erreurs / messages ok
Jeudi 25.04	Réalisation	8h-8h15: Corrections controller JWT 8h15-16h30: Vue candidature, layout, affichages des infos, des fichiers, séparation de la vue en composants, formatage des dates 16h30-17h30: Rapport et envoi des documents	Améliorer la page d'erreur (interface) Tester l'expiration de session
Mardi 30.04	Réalisation	8h-8h30: Modification globales, login au chargement, tri par date par défaut, changements sur vue globale 8h30-10h: Vue individuelle: style général 10h-11h: Correction de l'erreur dans le fileController, fonctionnalité telechargement des fichier et export données perso 11h-11h30: Tests, refactor code, messages en cas de données vides (vue indiv.) 11h30-14h40: Tests, implémentation des notes sur candidatures 14h30-15h30: Affichage des nouvelles candidatures (page accueil) 15h30 - 17h30 : Rapport et envoi des documents	
Mercredi 01.05		Maladie	Reporté au 09.05
Jeudi 02.05		Maladie	Reporté au 14.05
Mardi 07.05	Réalisation	8h-8h15: Mise a jour journal et planification suites aux absences 8h15-8h50: Affichages des nouvelles postulotions pour responsables 8h50-11h30: Affichages des nouvelles postulotions pour formateurs 11h30- 15h30: Fonctionnalité attribution de commentaires (design), récupération des infos et images 15h30-17h15: Rapport en envoi des documents	
Mercredi 08.05	Réalisation	8h-12h: Attributon de commentaires: design, tri par date, sous composants, ajout, édition et suppression 13h-14h15: Test des fonctionnalités de la vue individuelle, corrections esthétiques et de bugs trouvés grâce aux tests. 14h15-14h45: Changement des retours API 403 et 404, amélioration de la page d'erreur 14h45-16h20: Rédaction de la documentation administrateur 16h20-17h30: Rapport	
Jeudi 09.05	Réalisation et finalisation	8h-8h40: Vérification des conventions Vue.js 8h40-9h20: Tests globaux, recherche de bugs / erreurs 9h20-10h40: Documentation admin 10h40-12h: Mise en production de l'application pour permettre les tests du chef de projet 13h- 14h: Essai de correction des erreurs CORS sur la requete login, toujours sans succès 14h-14h15: Relecture des les critères TPI pour le rapport 14h15-17h30: Rapport	Les heures / minutes à un chiffre ne s'affichent pas comme souhaité (7h2, au lieu de 07h20) un peu partout dans l'application, c'est corrigé Le champ de recherche n'a pas de bouton pour vider le champs comme les autre filtre, je l'ai ajouté
Mardi 14.05	Finalisation	8h-8h30: Rédaction du résumé de projet 8h30-8h45: Amélioration des validateurs et tests 8h45-12h: Rédaction du rapport, corrections doc admin, documentation des tests 13h-14h30: Rédaction du rapport 14h30-16h30: Mise en lien des documents 16h30-17h30: Préparation envoi final du projet	Imprimer une version papier et l'envoyer à l'experte 2 heures supplémentaires (le 13.05) pour la réalisation des diagrammes (gestion erreurs et MCD) et relecture du rapport dans son état actuel Réservation de la salle pour la présentation

Mis à jour le 14.05.2019 à 17h00









Le projet « Canap-Gest » consiste à développer une solution de gestion des candidatures d'apprentissage pour l'EPFL.

Un formulaire de candidature d'apprentissage EPFL en ligne a antérieurement été développé par moi-même, cependant, après deux ans d'utilisation, la formation apprentis EPFL et moi avons décidé de faire évoluer notre système en y ajoutant une interface complète d'administration des candidatures, proposant de nombreuses fonctionnalités pour simplifier le travail de recrutement par les responsables et les formateurs EPFL. Des modifications sur le formulaire de postulation ont également été appliquées pour améliorer le confort d'utilisation par les potentiels apprentis et d'appliquer les nouvelles directives de protections de données personnelles (RGPD).

Pour réaliser ce projet, il a été pensé de développer une application web en suivant une architecture RESTful, disposant d'une API, développée avec le framework PHP Lumen qui interagit avec une base de données MySQL. L'interface de l'application a été développée avec le framework JavaScript Vue.js. Cette interface est donc une SPA totalement indépendante du backend au niveau du rendu. Cette interface s'adapte selon les permissions et le rôle de l'utilisateur (responsable ou formateur) et offre de nombreuses fonctionnalités, telles que l'affichage global et unique des postulations regroupant l'entier des données d'un candidat et ses annexes et l'attribution de notes et de commentaires privés ou partagés avec les autres formateurs de la même profession. L'interface dispose également d'outils d'exportation des annexes et des données et de « notifications » pour les nouvelles candidatures. Les responsables de la formation apprentis peuvent désormais gérer les places d'apprentissage disponibles et leur nombre ce qui aura pour but de restreindre les candidatures sur des places non disponibles.

L'utilisation du framework Vue.js offre à l'application une réactivité, fluidité et vitesse améliorant grandement le confort d'utilisation de l'utilisateur. Cela simplifie également le travail de développement, principalement du fait de la séparation backend et frontend sur deux sites. Les fonctionnalités demandées dans le cahier des charges ont été mise en place entièrement, cependant avant de pouvoir déployer ce projet en production, quelques fonctionnalités et modifications supplémentaires devront être ajoutées pour correspondre au mieux aux besoins des formateurs et responsables EPFL.

1 INFORMATIONS GENERALES

Candidat	Nom : CRAUSAZ	Prénom : Nicolas
	 nicolas.crausaz@epfl.ch	 +41 21 69 32813
Lieu de travail :	EPFL ENAC-IT, GC D1 383, Station 18, CH-1015 Lausanne	
Orientation :	<input type="checkbox"/> 88601 Développement d'applications <input checked="" type="checkbox"/> 88602 Informatique d'entreprise <input type="checkbox"/> 88603 Technique des systèmes	
Chef de projet	Nom : DUBOIS	Prénom : Nicolas
	 nicolas.dubois@epfl.ch	 +41 21 693 63 58
Expert 1	Nom : FOLOMIETOW	Prénom : Borys
	 borys@folomietow.ch	 +41 76 366 45 06
Expert 2	Nom : ANDOLFATTO	Prénom : Frédérique
	 Frederique.ANDOLFATTO@cpnv.ch	
Période de réalisation :	Du mardi 16.04.2019 au mercredi 08.05.2019	
Horaire de travail :	Mardi, mercredi, jeudi 8h-12h, 13h-17h30	
Nombre d'heures :	11 x 8 = 88 heures	
Planning (en H ou %)	Analyse : 16h	
	Implémentation : 46h	
	Tests : 10h	
	Documentations : 16h	

2 PROCÉDURE

- Le candidat réalise un travail personnel sur la base d'un cahier des charges reçu le 1er jour.
- Le cahier des charges est approuvé par les deux experts. Il est en outre présenté, commenté et discuté avec le candidat. Par sa signature, le candidat accepte le travail proposé.
- Le candidat a connaissance de la feuille d'évaluation avant de débiter le travail.
- Le candidat est entièrement responsable de la sécurité de ses données.
- En cas de problèmes graves, le candidat avertit au plus vite les deux experts et son CdP.
- Le candidat a la possibilité d'obtenir de l'aide, mais doit le mentionner dans son dossier.
- A la fin du délai imparti pour la réalisation du TPI, le candidat doit transmettre par courrier électronique le dossier de projet aux deux experts et au chef de projet. En parallèle, une copie papier du rapport doit être fournie sans délai en trois exemplaires (L'un des deux experts peut demander à ne recevoir que la version électronique du dossier). Cette dernière doit être en tout point identique à la version électronique.

3 TITRE

« CANAP-GEST » Application web de gestion des candidatures d'apprentissage EPFL

4 MATÉRIEL ET LOGICIEL À DISPOSITION

- 1 PC standard Windows 10
 - Visual Studio Code
 - WAMP server
 - MySQLWorkbench
 - Dépôt Git
-

5 PRÉREQUIS

- Bonnes connaissances HTML / CSS / JS / Vue.js / PHP / MySQL
 - Bonnes connaissances du système d'authentification web EPFL « Tequila »
 - Connaissances Git
-

6 DESCRIPTIF DU PROJET

Lors de sa 2^e année d'apprentissage, Nicolas Crausaz a développé, pour la formation apprentis EPFL, un formulaire web permettant aux candidats de postuler pour des places d'apprentissage. Le but étant de remplacer le formulaire papier envoyé par poste utilisé jusqu'alors. Le formulaire permet de saisir diverses informations (personnelles, scolaire, professionnelles ...) et des fichiers annexes (cv, lettre de motivation ...).

La suite du processus de recrutement n'est pas gérée par une application. Les données du formulaire sont stockées dans des fichiers JSON, rangés avec les documents annexes dans des dossiers par candidatures. Les responsables de la formation valident manuellement les candidatures en les déplaçant dans des dossiers accessibles aux formateurs. Ensuite des tableaux HTML statiques listant les candidats et leurs données sont générés périodiquement avec un script pour « simplifier » le travail des formateurs. Cette gestion « manuelle » n'est pas des plus pratiques.

Le projet TPI consiste à réaliser une interface web destinée aux responsables de la formation et aux formateurs, visant à simplifier le processus de recrutement et à le consolider en matière de protection des données à caractère personnel.

6.1 Travail préliminaire déjà réalisé

- Ajout de la possibilité au candidat d'éditer et de supprimer sa candidature à tout instant
- Mise en place d'une base de données MySQL pour stocker les données du formulaire et ainsi se débarrasser des fichiers JSON
- Mise en place d'une API en php avec quelques fonctions de base (lister les candidats, récupérer la liste des annexes, ...)
- Implémentation de l'authentification des formateurs et responsables formation via Tequila et vérification de leurs appartenance aux groupes de formateurs par corps de métier

6.2 Objectifs

Développer une application web avec interface en français qui offre ces fonctionnalités :

- Pour les responsables de la formation
 - Gérer la liste des sites (lieux) EPFL (Lausanne, Sion, Fribourg, Neuchâtel, ...)
 - Gérer la liste des corps de métier (Métier, nom du groupe des formateurs, nombre actuel de places disponibles)
 - Visualiser les nouvelles candidatures et permettre ces actions :
 - Valider (pour les rendre visibles aux formateurs)
 - Incomplet (le responsable devra alors signaler au candidat par tél ou email les infos/documents manquants)
 - Refuser
 - Visualiser les candidatures en cours
 - Refuser une sélection de candidats (avec suppression des dossiers)
- Formateurs
 - Lister les candidatures (liées à leur corps de métier)
 - Visualiser/télécharger les documents annexes
 - Identifier facilement les nouvelles candidatures
 - Attribuer des notes aux candidats
 - Ajouter des commentaires privés et publics (partagés avec les autres formateurs)

Le choix de la charte graphique est libre hormis le logo EPFL qui doit se situer en haut à gauche et un pied de page avec la mention « © 2019 EPFL, tous droits réservés ».

7 LIVRABLES

Le candidat est responsable de livrer à son chef de projet et aux deux experts, au moyen d'un email (avec un lien vers le Git) :

- Une planification initiale (livrée à la fin du premier jour)
- Un rapport de projet (livré 2 fois par semaine)
- Un journal de travail (livré 2 fois par semaine)
- Code source complet (livré le dernier jour)
- Un manuel pour les administrateurs (livré le dernier jour)

8 POINTS TECHNIQUES ÉVALUÉS SPÉCIFIQUES AU PROJET



La grille d'évaluation définit les critères généraux selon lesquels le travail du candidat sera évalué (documentation, journal de travail, respect des normes, qualité, ...).

En plus de cela, le travail sera évalué sur les 7 points spécifiques suivants (Point A14 à A20) :

1. *Gestion des exceptions : l'application affiche un message approprié lorsque la DB n'est pas accessible, lorsque l'API ne répond pas et lorsqu'un utilisateur n'est pas autorisé à voir une page ou un document*
2. *L'application se connecte sur la DB avec un compte qui a le minimum de droits nécessaires, ceux-ci sont expliqués dans le rapport*
3. *Chaque page affiche le logo EPFL en haut à gauche et la ligne de pied de page*

4. L'application s'affiche correctement sur Firefox et Chrome
5. L'application n'est pas sensible aux attaques de type XSS
6. L'application n'est pas sensible aux attaques de type SQL injection
7. Le code respecte les conventions Vue.js de priorité A (<https://fr.vuejs.org/v2/style-guide/index.html>), les exceptions sont possibles mais doivent être documentées dans le rapport.

9 VALIDATION

	Lu et approuvé le :	Signature :
Candidat :	16.04.2019	
Expert n°1 :	16.04.2019	
Expert n° 2 :		
Chef de projet :	16.04.2019	