

ENAC-IT2

EPFL | Nicolas Crausaz

TPI – Canap-Gest

« Canap-gest » Application WEB de gestion de candidatures d’apprentissage epfl

# Sommaire

[Sommaire 1](#_Toc3460783)

[1 Analyse préliminaire 2](#_Toc3460784)

[1.1 Introduction 2](#_Toc3460785)

[1.2 Situation actuelle 2](#_Toc3460786)

[1.3 Objectif 4](#_Toc3460787)

[1.4 Travail préalable 5](#_Toc3460788)

[2 Analyse du projet 7](#_Toc3460789)

[2.1 Points importants du cahier des charges 7](#_Toc3460790)

[2.2 Faisabilité du projet 7](#_Toc3460791)

[2.3 Choix des technologies 7](#_Toc3460792)

[2.4 Planification initiale 7](#_Toc3460793)

[2.5 Matériel et ressources à disposition 7](#_Toc3460794)

[2.5.1 Global 7](#_Toc3460795)

[2.5.2 Services 7](#_Toc3460796)

[2.5.3 Dépendances 7](#_Toc3460797)

[2.5.4 Description & Installation 8](#_Toc3460798)

[2.6 Versionning et sauvegardes 10](#_Toc3460799)

[3 Réalisation 10](#_Toc3460800)

[3.1 Modèle DB 10](#_Toc3460801)

[3.2 Structure de l’application 10](#_Toc3460802)

[3.3 Authentification 10](#_Toc3460803)

[3.4 Requêtes API 11](#_Toc3460804)

[3.5 Interface 12](#_Toc3460805)

[3.5.1 Structure 12](#_Toc3460806)

[3.5.2 Fonctionnalités 12](#_Toc3460807)

[3.6 Tests 13](#_Toc3460808)

[4 Conclusion 13](#_Toc3460809)

[4.1 Atteinte des objectifs 13](#_Toc3460810)

[4.2 Planification 13](#_Toc3460811)

[4.3 Problèmes rencontrés 13](#_Toc3460812)

[4.4 Améliorations possibles 13](#_Toc3460813)

[4.5 Évolution 13](#_Toc3460814)

[5 Annexes 13](#_Toc3460815)

[5.1 Sources 13](#_Toc3460816)

[5.2 Compte-rendu des discussions 14](#_Toc3460817)

[5.3 Journal de travail 14](#_Toc3460818)

[5.4 Code source 14](#_Toc3460819)

[6 Glossaire 14](#_Toc3460820)

# Analyse préliminaire

## 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, scolaire, 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.

## Situation actuelle

La personne désirant postuler pour un apprentissage à l’EPFL se rend sur le formulaire, crée un compte au travers du service « compte guest » de l’EPFL, puis rempli 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 retenu et prochainement engagés

Les responsables de la formation apprentis EPFL s’occupent de procéder à la sélection des dossiers et de transmettre les dossiers aux formateurs qui choisiront leur(s) futur(s) apprenti(s).

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 entrainer 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 entraine des conflits avec la loi RGPD*.*

## Objectif

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

* Eviter 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 valide par les responsables Formation Apprentis EPFL.
* Faciliter le travail de sélection et recrutement des postulants par les formateurs EPFL.

L’idée qui à été retenue pour obtenir ses 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 postulations (sans les fichiers) dans une base de données pour en faciliter l’accès ultérieur.
* Développer une API permettant l’accès à la base de données, faisant le lien entre l’interface et la base de données.

## Travail préalable

Une grande 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 évidement 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 subvenir aux « besoins » de la future interface. J’ai choisi le *Framework « Lumen »* pour son développement.

L’ajout de la base de données à déjà permis de combler la problématique du traitement des fichiers d’information directement sur le partage réseau et d’en simplifier sa structure, puisque que celui-ci ne sera utiliser que pour les fichiers annexes des candidatures et que l’accès direct au serveur ne sera plus utile.

Le diagramme suivant résume la situation :

# Analyse du projet

## Points importants du cahier des charges

Expliquer les différents points demandés du cahier des charges, fonctionnalités demandées et les étapes de réalisation de chacune.

## Faisabilité du projet

Connaissances techniques, charge de travail, temps a dispo etc.

## Choix des technologies

Déjà choisie Lumen etc

Exigé Vue.js

Vue, vuetify

Mon expérience perso avec vue.js

## Planification initiale

Planif initale

Trello

## Structure de l’application

REST et shema

Pourquoi choisir le RestFull ?

Expliquer meme code pour plsieurs appli et séparation front-back

## Matériel et ressources à disposition

### Global

* Un PC sous Windows 10 Education
* Un IDE, Visual Studio Code
* Un client SQL, MySQLWorkbench
* Un client http, Postman

### Services

* Client tequila

### Dépendances

L’utilisation, fonctionnement et l’installation des dépendances suivantes sont décrites au point 2.5.4.

* PHP version 7.2
* Composer (php packet manager)
  + Ajouter php au « path » windows
* Lumen Framework
* firebase/php-jwt
* Node.js (v.11)
  + Npm (node.js packet manager)
* @vue/cli (Vue.js)
* Vue-router
* Vuex
* Vuetify

### Description & Installation

**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’installeur sur : <https://getcomposer.org/doc/00-intro.md#installation-windows>
* Ajouter le chemin vers PHP à la variable d’environnement PATH
* Cmd : > composer, pour valider l’installation

**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 le routing, les middlewares, l’authentification, la validation etc.

Pour créer un projet Lumen avec une structure de base, dans un cmd :

* *composer create-project --prefer-dist laravel/lumen <nom\_projet>*

Puis pour démarrer l’application :

* *php -S localhost:8000 -t public*

Editer le fichier  « .env » avec les informations demandées : APP\_KEY et info de connexion à la DB

**firebase/php-jwt :**

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

Pour importer cette libraire, ouvrir un cmd dans le dossier du projet Lumen, puis :

* *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 crypter/décrypter les tokens JWT. Il suffit de générer une chaine de caractères pour cette valeur.

**Node.js :**

Node.js est un environnement JavaScript permettant d’interpréter du JavaScript côte 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.

* Télécharger et installer Node.js (v.11 actuellement) sur : <https://nodejs.org/en/>

**Vue.js**

*VUEJS*

Orienté composant

Revient en détail dans un autre point ( Dev, code d’exemple, cycle de vie composant, evenement, v-model etc.)

**@vue/cli**

Outil en ligne de commande servant à générer des projet Vue.js avec une structure de base, en choisissant les dépendances souhaitées.

Il faut installer le CLI de manière globale :

* *npm install @vue/cli* ***-g***

L’argument -g permet d’installer le module de manière globale au système. Une fois installé on peut l’utiliser depuis un prompt :

* *vue*

Pour créer un nouveau projet :

* *vue create <nom\_app>*

Choisir manuellement les paquets :

* Babel
* Vuex
* Router
* Linter / Formatter

Use history mode ? N

Il s'agit d'enlever le "#" dans l'url qui est affiché par vue-router. Si l’on choisit oui, il faudra avoir un .htaccess (ou autre) qui redirige toutes les requetes sur index.html

Linter / formatter config :

* ESLint + Standard config
* Lint on save

ESLint permet de valider la manière dont le code est écrit et d’assurer les bonnes pratiques en manière de code (indentation, code superflu etc.) en se basant sur les normes ECMAScript. Ici, j’utilise les standards officiels JavaScript[[1]](#footnote-1).

Les Frameworks JavaScript actuels utilisent la norme ECMAScript Version 6, appelée ES6, alors que les navigateurs ne supportent pas encore totalement cette norme, c’est pour que cela qu’il faut utiliser un transcompilateur (en JavaScript, paquet Babel installé précédemment) qui permet de convertir notre code vers la norme ES5, supportée par les navigateurs.

Configuration des paquets :

* In dedicated config files

Lors de la création du projet, un projet Git est initialisé et un commit initial est effectué.

**Vue-router**

**Vuex**

Fonctiknnement des actions, mutations, getters etc et interaction composants

« Vuex, c'est comme les lunettes : vous saurez quand vous en aurez besoin. »

**Vuetify**

* *vue add vuetify*

## Versionning et sauvegardes

C4science, git

# Réalisation

## Modèle DB

## Structure de l’application

REST

## Authentification

JWT

## Requêtes API

Liste des requêtes par « Controller » :

AuthController (authentification)

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

UsersController (utilisateur)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Route | Auth (token) ? | Rôle | Body (x-form-urlencoded) | | Description |
| GET | api/user | Oui | tous | - | info user connecté | |
| GET | api/user/permittedjobs | Oui | tous | - | Retourne les métiers visibles par l’user | |
| GET | api/user/hascommentedormarked | Oui | tous | - | Retourne les postulation commentées ou marquées par l’user | |
| GET | api/user/data/{sciper} | Oui | tous | - | Retourne les infos LDAP d’un utilisateur selon son sciper | |

ApplicantsController (postulations)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Route | Auth (token) ? | Rôle | Body (x-form-urlencoded) | | Description |
| GET | api/applicants | Oui | tous | - | Retourne les postulation des tous les metiers 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 télécharger les données (json) d’une postulation | |
| PATCH | applicant/{id} | Oui | resp | status | Modifie le statut d’une postulation | |
| DELETE | applicant/{id} | Oui | resp | - | Supprime une postulation | |

CommentsController (commentaires)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Route | Auth (token) ? | Rôle | Body (x-form-urlencoded) | | 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 | |

MarkersController (marqueurs)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Route | Auth (token) ? | Rôle | Body (x-form-urlencoded) | | Description |
| GET | api/applicant/{id}/markers | Oui | tous | - | Récupère les marqueurs d’une postulation | |
| GET | api/applicant/{id}/usermarkers | Oui | tous | - | Récupère les marqueurs d’une postulation d’un user | |
| PUT | api/marker | Oui | tous | type, applicant\_id | Ajoute un marqueur | |
| PATCH | api/marker/{id} | Oui | tous | type | Edite un marqueur | |
| DELETE | api/marker/{id} | Oui | tous | - | Supprime un marquer | |

FilesController (fichiers)

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

StatsController (statistiques)

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

## Interface

### Structure

Structure des pages et description

Routes

### Fonctionnalités

## Tests

# Conclusion

## Atteinte des objectifs

* Objectifs atteints / non-atteints
* Eléments restants

## Planification

* Comparaison du réel avec la planification initiale

## Problèmes rencontrés

* Points positifs / négatifs
* Difficultés particulières

## Améliorations possibles

Amélioration état actuel (fin TPI)

## Évolution

* Suites possibles pour le projet (évolutions long terme)

# Annexes

## Sources

<https://nodejs.org/en/>

<https://lumen.laravel.com/>

<https://cli.vuejs.org/guide/>

<https://vuejs.org/>

<https://github.com/axios/axios>

<http://tequila.epfl.ch/>

<https://vuetifyjs.com/en/>

<https://jwt.io/>

<https://github.com/firebase/php-jwt>

<https://vuex.vuejs.org/fr/>

Services :

<https://mockflow.com/>

## Compte-rendu des discussions

Discussions et échanges avec le formateurs et experts

## Journal de travail

## Code source

# Glossaire

A

API

application programming interface, interface permettant de fournir des services à d'autres applications 4, 5

C

Canap-Gest

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

CLI

Command-line interface 10

compte guest 2

E

ECMAScript

Normes JavaScript en matière de code 9

J

JSON

Format de données dérivé de la syntaxe des object JavaScript 2, 4

JWT

JSON Web Token 8

N

NPM

Node.js Packet Manager 9

R

RGPD

Règlement général sur la protection des données 4

T

transcompilateur

Converti un code source d'un language vers un autre 9

Git

Rest

Framework

<https://www.edoeb.admin.ch/edoeb/fr/home/documentation/bases-legales/Datenschutz%20-%20International/DSGVO.html>

1. <https://standardjs.com/> [↑](#footnote-ref-1)