CPES 3

Rapport de stage
Développement agile d'applications web

Auteur :
Barnabé Geffroy

Référent :
Olivier Cailloux

26 juin 2020

Table des matières

Introduction **1**

Préliminaires **2**

1 Gestion des présences **3**
1.1 Attendance . . . 3
1.2 JeSuisEnCours . . . 3

2 Plaquette-MIDO **4**
2.1 L'automatisation du lancement du code . . . 4
2.2 L'authentification . . . 6

3 CredsRead **7**
3.1 Diagramme de classe . . . 7
3.2 Les différentes classes . . . 7
3.3 Intégration à plaquette-MIDO . . . 9

4 Jugement délibéré, les régimes alimentaires **10**
4.1 Protocole de l'expérience . . . 10
4.2 Création d'un site web . . . 10
4.3 Déploiement sur Internet . . . 12

Remerciements **14**

Conclusion **14**

Annexes **i**

Codes Plaquette-MIDO **ii**
1 Scripts pour Travis-CI . . . ii

Codes CredsRead **iv**
2 Classe CredsReader . . . iv

Codes Jugement Délibéré **vi**
3 Jekyll . . . vi

Bibliographie

Introduction

Le développement agile est une méthodologie qui vise à délivrer rapidement une solution fonctionnelle et à améliorer progressivement le code. Cette méthode permet une simplification du code et d'explorer des voies suggérées lors du développement et inenvisagées initialement. L'ajout de tests automatisés fréquents permet de ne pas se perdre dès que le code renvoie une erreur. Une communication régulière entre les développeurs est également nécessaire en développement agile.

Lors de ce stage, plusieurs projets ont été réalisés en suivant cette méthode. Il seront exposés dans ce rapport. Premièrement, quelques détails techniques seront exposés. Ensuite, les quatre projets seront présentés. Le premier projet traite du développement d'une application web gérant la présence des élèves au sein d'un master de l'Université Paris-Dauphine. Le second projet s'intéresse à une application générant un fichier PDF contenant le programme du même master. Le troisième projet est un projet découlant du second. Celui-ci n'était initiallement pas prévu mais la méthodologie agile nous a conduit à développer un système d'authentification. Le dernier projet est une mise en pratique de l'article *A formal framework for deliberated judgment*[1]. Une annexe à la fin du document propose certains de mes codes utilisés dans les projets. Tous les codes sont également disponibles sur mon profil GitHub : `https://github.com/barnabegeffroy`.

Préliminaires

Lors de ce stage `Git` et Maven ont été utilisés sur la plupart des projets. En voici une présentation succinte.

`Git`

Fonctionnement

`Git` est un système de contrôle de version qui permet la collaboration entre développeurs. Le code source est conservé dans un *dépôt* distant. Il suit un modèle distribué, il n'y a pas de serveur central. Le code est donc accessible par plusieurs sources et peut être utilisé sans connexion. Une connexion internet est cependant nécessaire pour envoyer ses modifications sur le dépôt distant. Ce genre de sauvegarde est appelé *commit*. Celle-ci est une version du code à un instant donné. `Git` crée, avec tous les commits, une série d'instantanées évite le risque de perte d'information.

`Git` gère également l'intégrité du code. Il peut y avoir des conflits, des parties identiques du code modifiées par deux utlisateurs. `Git` identifie les régions du code qui sont différentes et les utilisateurs éditent le code pour régler les zones de conflit.

Quelques commandes

- `git init` initialise un nouveau dépôt.
- `git clone` copie un dépôt `Git` déjà existant.
- `git add` ajoute les fichiers que l'on veut sauvegarder dans le commit.
- `git status` affiche l'état des fichiers (ajoutés ou non).
- `git commit` crée un instantané du code en modifiant les fichiers ajoutés avec `git add`.
- `git push` envoie les commits sur le dépôt distant.

Seules les commandes `git clone` et `git push` nécessitent une connexion internet. Il est donc très aisé de travailler sans connexion sur le code et de se connecter simplement pour pousser vers le dépôt `Git` les différents commits.

GitHub

GitHub est un hébergeur de dépôts `Git`. Il permet la gestion de version distribuée, la fonctionnalité de gestion de code source de `Git` et propose ses propres fonctionnalités. Il compte plus de 50 millions d'inscrits et plus de 100 millions de dépôts.

Maven

Maven est un outil de gestion de configuration de projet Java, en particulier de gestion des dépendances. Il permet de ne pas se soucier de l'environnement de compilation. Les dépendances sont indiquées dans un fichier nommé `pom.xml`. Grâce à ce fichier Maven configure les bibliothèque et autres dépendances. Maven propose aussi une structure du projet qui sera la même dans chacun des projets exposés.

Voici l'arborescence de base d'un projet Maven :

```
pom.xml
/src
      /main
         /java
         /resources
      /test
         /java
         /resources
```

Projet 1

Gestion des présences[1]

L'offre de stage[2] à laquelle je me suis porté candidat portait initialement sur le développement d'une application gérant la présence des élèves du Master 1 MIAGE en Apprentissage. L'idée de ce projet était de digitaliser les feuilles de présence.

1.1 Attendance

Le projet Attendance avait donc pour but de développer une application web permettant au professeur de faire l'appel et d'envoyer les données à l'administration. Dans un premier temps, j'ai conçu une application lisant un fichier JSON[3] contenant une liste d'élèves et affichant celle-ci de manière à faire l'appel. Pour cela, le serveur HTTP Eclipse Jetty fournit les services web pour des applications Java. Le projet Attendance contient ainsi une classe `StudentsList` qui créé un fichier JSON contenant la liste des élèves. Une classe `RecordAttendance` récupère la liste et l'affiche dans une page HTML (voir Figure 1.1). Cette classe permet aussi de récupérer les informations entrées par l'utilisateur (les élèves absents cochés par le professeur).

```
1  [
2      {
3          "First Name": "Bertrand",
4          "Last Name": "Russell"
5      },
6      {
7          "First Name": "Friedrich",
8          "Last Name": "Frege"
9      }
10 ]
```

Fichier JSON

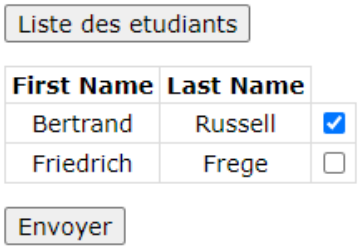

FIGURE 1.1 – Page HTML généré par `RecordAttendance`

1.2 JeSuisEnCours

Après quelques semaines, nous nous sommes rendus compte que la direction du projet était incompatible avec les exigences administratives. En effet, l'application prévoyait un simple appel du professeur, or l'étudiant doit personnellement attester de sa présence en émargeant un document. Il a ainsi été décidé d'abandonner le projet initial Attendance pour se tourner vers une application JeSuisEnCours, spécialisée dans la digitalisation les feuilles de présences.

Le but principal du projet est alors devenu la connexion de l'application JeSuisEnCours aux données de l'université, d'une part, pour accéder aux données (annuaires, emplois du temps,...), d'autre part pour gérer les éléments renvoyés par l'application (absence, justificatif,...).

Malheureusement, la crise sanitaire a fortement ralentit les contacts avec l'équipe de JeSuisEnCours et le projet a finalement été abandonné.

1. `https://github.com/barnabegeffroy/Attendance`
2. Offre de stage: développement agile d'applications web et de bibliothèques open-source
3. JSON est un format réprésentant les données de manière structurée.

Projet 2

Plaquette-MIDO[4]

Ce projet a pour but d'implémenter un code générant un fichier PDF détaillant les différents enseignements du Master 1 MIAGE en apprentissage à partir de la base de données de Dauphine. La finalité est d'automatiser le lancement du code de sorte que quotidiennement le fichier PDF soit mis à jour et publié en ligne.

À mon arrivée, un code permettait déjà la génération du fichier PDF. Certains passages du code étaient cependant à revoir pour améliorer l'esthétique du fichier PDF. De plus, le code initial était très peu généralisé à d'autres utilisateurs et plusieurs changements étaient nécessaire pour qu'un autre utilisateur puisse lancer Plaquette-MIDO. Le principal aspect à généraliser était l'authentification à l'API de Dauphine[5], indispensable pour avoir accès aux données de l'université.

2.1 L'automatisation du lancement du code

La finalité du projet Plaquette-MIDO est de lancer la construction du fichier PDF quotidiennement et de le publier de manière à ce qu'il soit accessible sur le site de Dauphine. Pour réaliser cette automatisation, j'ai utilisé l'outil Travis-CI.

2.1.1 Travis-CI

Travis-CI est un logiciel d'intégration continue qui permet de compiler, tester et déployer le code de dépôts GitHub. Il est configuré à partir d'un fichier nommé `.travis.yml` présent dans la racine du répertoire. Celui-ci est lu à chaque nouveau commit par Travis-CI qui exécute son contenu sur une machine virtuelle. L'exécution peut alors réussir dans le cas où aucune erreur n'a été signalée ou échouer si une erreur est survenue. Travis-CI peut ainsi s'assurer de la bonne compilation d'un projet. Il peut également effectuer des déploiements. En effet, il est possible d'insérer du script que Travis-CI exécute pour déployer des fichiers. Par exemple, ajouter un script avec les commandes git adéquat pour pousser des fichiers vers un dépôt. Travis-CI exécute le code source permettant la création du fichier PDF et ensuite déploie ce fichier vers un dépôt GitHub.

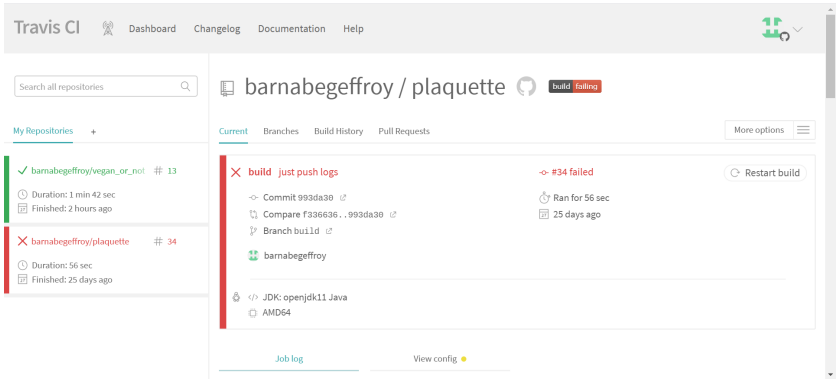

FIGURE 2.1 – Capture d'écran de l'interface de Travis-CI

La figure 2.1 montre que la construction du commit `just push logs` a échoué pour le dépôt `plaquette`, tandis que celle du dépôt `vegan_or_not` a réussi.

Travis-CI permet également de référencer des variables d'environnement sécurisées (clefs d'accès, identifiants, ...).

4. `https://github.com/Dauphine-MIDO/plaquette-MIDO`
5. Une API est une interface de programmation d'application qui permet d'accéder à un ensemble de classes, méthodes, fonctions et autres données. Dans le cas de Dauphine, son API donne accès aux fonctions informatiques permettant de manipuler le programme des cours des différentes formations.

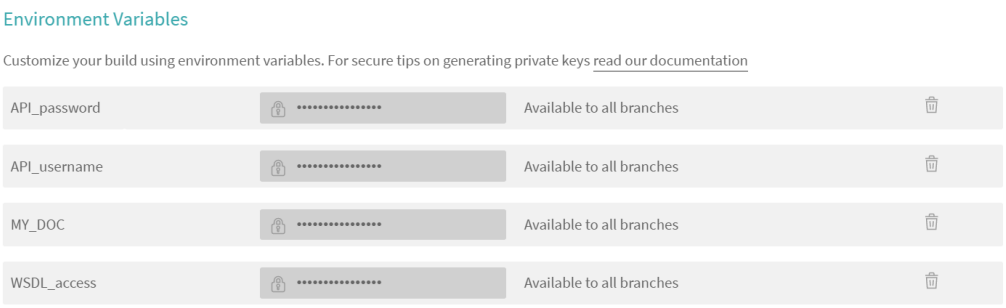

FIGURE 2.2 – Différentes variables d'environnement entréees pour la construction Travis-CI d'un dépôt

2.1.2 L'exécution du code

Les dépendances

Travis-CI identifie automatiquement un projet Maven et installe les dépendances indiquées dans le `pom.xml`. Dans le projet Plaquette-MIDO, la dépendance qui importe le code source de l'API de Dauphine nécessite un fichier texte nommé `WSDL_Login.txt` contenant un URL spécifique aux identifiants de l'utilisateur. Ce fichier ne peut pas être dans le dépôt car il contient des informations personnelles. Il faut un script qui crée le fichier sur la machine virtuelle de Travis-CI. Le fichier doit être créé avant de lancer l'installation des dépendances . Dans le `.travis.yml`, on peut ajouter un script qui génère ce fichier.

Voici ci-dessous un extrait de mon implémentation permettant la création d'un tel fichier. On y retrouve les variables d'environnement présentées dans la figure 2.2.

```
1 API_username = $(urlencode "${API_username}")
2 API_password = $(urlencode "${API_password}")
3
4 echo https://${API_username}:${API_password}@* > WSDL_login.txt
```

Extrait de writeWSDL.sh, annexe 1.1

Création du fichier PDF et déploiement

Une fois que toutes les dépendances du projet Maven sont installées, il faut executer le code source qui génère le fichier PDF. La classe qui permet cette génération est `M1ApprBuilder`. Le script de Travis-CI va exécuter cette classe, il faudra ensuite déployer vers un dépôt d'arrivée le fichier PDF ainsi que le logs de la construction. La construction de Travis-CI doit échouer si le fichier PDF n'est pas généré. Néanmoins, dans tous les cas les logs doivent être poussés vers le dépôt. Ainsi, si le fichier PDF est généré, il est poussé avec les logs vers le dépôt d'arrivée. Si ce n'est pas le cas, la construction échoue, le dernier PDF déployé reste disponible. Nous sommmes avertis immédiatement par e-mail de cet échec. Les logs du code source de Plaquette-MIDO ainsi que ceux de Travis-CI permettront alors d'expliquer de l'échec.

Vous trouverez mon implémentation du script exécuté par Travis-CI dans l'annexe 1.2. Le script exécute plusieurs fonctions :

- `clean` (l.20 à 24), supprime les fichiers déjà existants.
- `get_current_deploy` (l.26 à 30), copie sur la machine virtuelle de Travis-CI le dépôt dans lequel les fichiers vont être déployés. Pour éviter de cloner un dépôt `Git` dans un autre dépôt `Git`, celui-ci est cloné dans le dossier parent de Plaquette-MIDO de la machine virtuelle `"../$REPO"`.
- `build_doc` (l.31 à 41), essaie de générer le document en exécutant `M1ApprBuilder`. Elle met aussi à jour la variable `BUILT_EXIT_CODE` qui prend la valeur 0 si le code est correctement exécuté et que le fichier PDF est généré, ou la valeur 1 sinon.
- `deploy` (l.43 à 60), déplace les logs et le fichier PDF (s'il y en a un) vers le dépôt d'arrivée (`"../$REPO"`). Les différentes commandes `Git` présentées ultérieurement sont alors executées pour déployer les fichiers déplacés.
  La dernière ligne `exit $BUILT_EXIT_CODE` renvoie à Travis-CI la valeur mis à jour dans `build_doc`. Si

la valeur est 0, la construction contiue et s'achève par un succès. Sinon la construction échoue et une notification est envoyée pour prévenir les développeurs.

Automatisation

Travis-CI lance initiallement la construction du code du dépôt à chaque nouveau commit. Il est cependant possible de configurer des *Cron Jobs*. Ceux-ci vont renouveler la construction du code de manière quotidienne, hebdomadaire ou mensuelle.

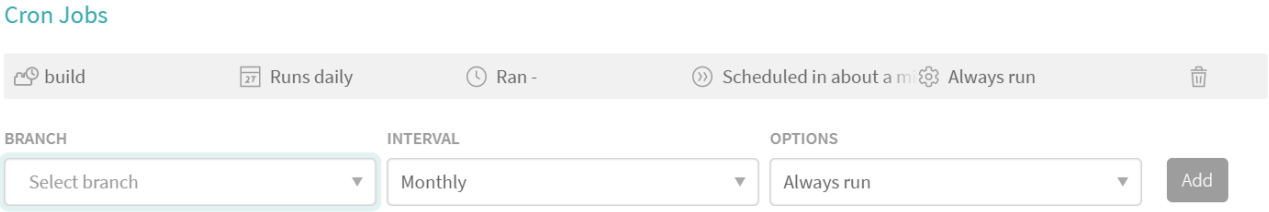

FIGURE 2.3 – Les *Cron Jobs* de Travis-CI, ici la construction est programmée quotidiennement

En programmant sur *Daily*, Travis-CI va relancer la construction du code tous les jours et ainsi renouveller le fichier PDF et les logs si des changements sont effectués. Si une erreur se produit et que le fichier PDF n'est pas généré, une e-mail nous préviendra de la non-construction du code et le fichier PDF le plus récent sera toujours disponible sur le site.

2.2 L'authentification

Pour se connecter à l'API de Dauphine, un nom d'utilisateur et un mot de passe sont nécessaires. Le code initial prévoyait trois manières de fournir ces informations afin de se connecter à l'API :

— les propriétés du système
— les variables d'environnement
— un fichier texte contenant les informations nécessaires

Cependant, ce code ne lisait initialement que le mot de passe et le nom d'utilisateur était une valeur par défaut. Un nouvel utilisateur devait modifier le code pour pouvoir utiliser ses identifiants. L'idée d'une valeur par défaut pour le nom d'utilisateur a donc été abandonnée pour rendre le programme plus accessible. La valeur du nom d'utilisateur serait lue de la même manière que celle du mot de passe.

2.2.1 La classe Authentication

Une nouvelle classe, `Authentication`, a été créée pour permettre la généralisation lecture du code et améliorer sa lisibilité. Celle-ci permet de créer un objet contenant un nom d'utilisateur et un mot de passe de type `Optional`. Ce type permet d'instancer aussi bien la valeur d'une chaîne de caractère que l'absence d'une information. Cette classe `Authentication` est lu dans une autre classe, `QueriesHelper`, qui permet de renvoyer les informations nécessaires pour se connecter à l'API. L'introduction de la classe `Authentication` permet ainsi de détecter si le nom d'utilisateur ou le mot de passe manquent et alors jeté une exception appropriée faisant échoué l'exécution du code.

2.2.2 Le projet CredsRead

La généralisation du code permettant l'authentification nous a poussés à séparer dans un projet bien distinct de Plaquette-MIDO, le projet CredsRead. En effet, les méthodes de `QueriesHelper` ainsi que la classe `Authentication` n'avait, en grande partie, aucun lien spécifique avec plaquette-MIDO et pouvait ainsi être totalement publiées dans un autre projet pour pouvoir réutiliser plus facilement ce code dans d'autres projets. Le projet Creds-Read a ensuite été intégré au code source de Plaquette-MIDO.

Projet 3

CredsRead[6]

CredsRead, pour Credentials Read, gère comme son nom l'indique la lecture des identifiants d'un utilisateur. Celle-ci s'effectue de trois manières, comme pour l'authenfication dans Plaquette-MIDO, les propriétés du système, les variables d'environnement et un fichier texte.

3.1 Diagramme de classe

Le diagramme de classe permet d'avoir une idée précise du code que l'on veut écrire. Papyrus est un outil permettant de réaliser ce genre de diagramme. Son interface intuitive facilite la rédaction d'un diagramme UML(le langage standard des diagrammes de classe) lisible et rigoureux.

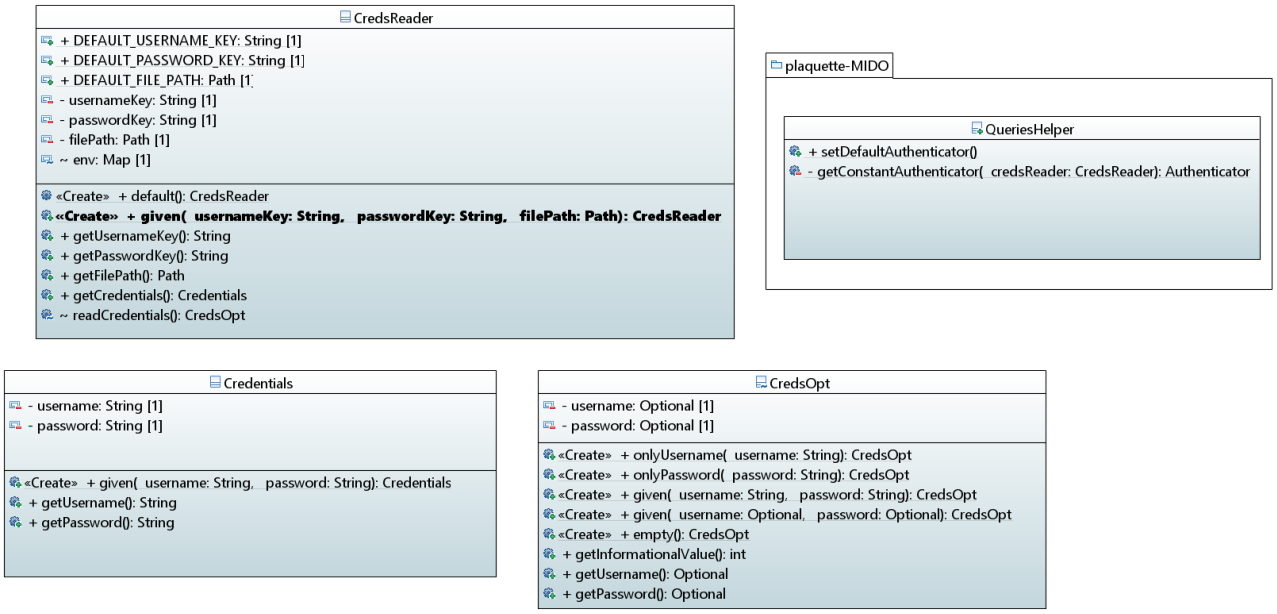

FIGURE 3.1 – Diagramme de classe du projet CredsRead

Le projet est composé de trois classes représentées par les différents blocs. Chaque bloc est constitué de son nom, suivi d'une partie avec l'ensemble des variables de la classe, et d'une partie avec l'ensemble des méthodes de la classe. Les éléments soulignés sont des statiques. Le symbole + précédant les éléments indique qu'ils sont publics, le − privés, le ∼ de visibilité package[7]. Les méthodes sont présentées de cette façon `nomDeMethode(paramètres d'entrée): objet de sortie`. Le préfixe `Create` indique que ce sont des méthodes de construction. Il y a également un sous-dossier de plaquette-MIDO contenant la classe `QueriesHelper`. Celle-ci ne contient plus que deux méthodes spécifiques à la connexion à l'API de Dauphine.

3.2 Les différentes classes

3.2.1 `CredsOpt`

Cette classe remplace la classe `Authentication`, elle crée un objet contenant les identifiants de l'utlisateur, le nom d'utilisateur et le mot de passe, avec le type `Optional`. Celle-ci permet d'instancer également l'abscence

6. `https://github.com/oliviercailloux/creds-read`
7. En dehors du projet, ces éléments ne sont pas accessibles.

d'information avec un `Optional` vide. Une distinction est ainsi faite entre un `Optional` vide (absence d'information) et un `String` vide : `""`. Ce dernier représente un nom d'utilisateur ou un mot de passe vide. La méthode `getInformationValue()` renvoie le nombre d'informations présentes (différente d'un `Optional` vide) dans l'objet `CredsOpt` (0, 1 ou 2).

Cette classe est de visibilité package, elle n'est utilisée que par les autres classes du package et n'est pas accessible publiquement.

3.2.2 `Credentials`

Cette classe crée également un objet contenant les identifiants, le nom d'utilisateur et le mot de passe. Ceux-ci sont cette fois de type `String`. Par rapport à `CredsOpt`, cette classe, ne permet pas d'instancer l'abscence d'information.

Cette classe est publique, elle peut être utilisée en dehors du package par les utilisateurs.

3.2.3 `CredsReader`

`CredsReader` est la classe principale du projet. Elle est instancée par trois variables : une clé pour le nom d'utilisateur `usernameKey`, une clé pour le mot de passe `passwordKey` et un chemin vers un fichier `filePath`. Les deux clés sont utilisées pour lire les propriétés du système et les variables d'environnement. Le chemin correspond au fichier texte contenant les informations d'identification.

L'utilisateur peut créer un objet `CredsReader` en utilisant ses propres noms pour les variables d'instances. Il peut aussi utiliser les valeurs par défaut de la classe : `API_username`, `API_password`, `API_login.txt`. Ces derniers sont ceux utlisés dans plaquette-MIDO.

3.2.3.1 `readCredentials()`

Voir annexe 2.1.

`readCredentials()` est la méthode la plus importante de la classe. Elle est de visibilité package, elle n'est pas accessible aux utilisateurs. Elle renvoie un `CredsOpt`.

Elle permet de lire les informations d'identification : nom d'utilisateur et mot de passe. Pour chaque information, on distingue l'information manquante et la `String` vide. La méthode prend en compte les sources possibles suivantes :

1. l.3 à 11 : Propriétés du système. Chaque propriété peut être définie, y compris pour la chaîne vide, ou non définie. Une information est considérée comme manquante (dans la source des propriétés) si la propriété correspondante n'est pas définie.
2. l.13 à 21 : Variables d'environnement. Chaque variable peut être définie, y compris par une chaîne vide, ou non définie. Une information est considérée comme manquante (dans la source des variables d'environnement) si la variable d'environnement correspondante n'est pas définie.
3. l.23 à 55 : Fichier texte. Les deux informations sont considérées comme manquantes (de la source des fichiers) si le fichier n'existe pas. Si le fichier existe, aucune information n'est considérée comme manquante. La première ligne du fichier donne le nom d'utilisateur, la seconde le mot de passe. Si le fichier n'a qu'une seule ligne, le mot de passe (provenant de la source des fichiers) est mis à vide. Si le fichier est vide, les deux informations (provenant de la source des fichiers) sont fixées à la `String` vide. Les lignes vides ne sont pas prises en compte du tout. Si le fichier ne contient pas de ligne vide après la deuxième ligne, une exception est lancée.

l.51 à 61 : La méthode a une logique de meilleure information de connexion. La source utilisée pour renvoyer un `CredsOpt` contenant les informations est celle qui a la valeur informationnelle la plus élevée, telle que déterminée par `CredsOpt.getInformationalValue()` (ce qui signifie que les sources sont classées par ordre croissant du nombre d'informations manquantes), et, dans le cas d'un fichier ex-æquo, l'ordre de priorité, tel que présenté ci-dessus, détermine quelle source est choisie.

3.2.3.2 `getCredentials()`

Voir annexe 2.2

`getCredentials()` est la classe publique renvoyant un `Credentials`. Elle utlise la méthode `readCredentials()` pour récupérer la meilleure information d'identification (logique de priorité). Elle lance une exception si une (ou les deux) information manque.

3.3 Intégration à plaquette-MIDO

Le projet CredsRead a d'abord été édité dans le projet Plaquette-MIDO. Il a ensuite été séparé du projet initial. Il fallait donc l'intégrer. Pour cela, le projet CredsRead est publié sur Maven Central. Le projet peut alors être intégré à plaquette-MIDO en l'ajoutant dans le `pom.xml` et de cette façon être utilisé dans la classe `QueriesHelper` de plaquette-MIDO.

```
1  public class QueriesHelper {
2
3   public static void setDefaultAuthenticator() {
4    final Authenticator myAuth = getConstantAuthenticator(CredsReader.
        defaultCreds());
5    Authenticator.setDefault(myAuth);
6   }
7
8   private static Authenticator getConstantAuthenticator(CredsReader
       credsReader) {
9    Credentials credentials = credsReader.getCredentials();
10   final PasswordAuthentication passwordAuthentication = new
        PasswordAuthentication(credentials.getUsername(),
11     credentials.getPassword().toCharArray());
12   final Authenticator myAuth = new Authenticator() {
13    @Override
14    protected PasswordAuthentication getPasswordAuthentication() {
15     return passwordAuthentication;
16    }
17   };
18   return myAuth;
19  }
20 }
```

La classe `QueriesHelper`, utilisant le projet CredsRead pour plaquette-MIDO.

Projet 4

Jugement délibéré, les régimes alimentaires[8]

Ce projet a pour but de mettre en pratique les théories exposées dans l'article *A formal framework for deliberated judgment*[1]. Cette article de théorie du choix s'intéresse à l'influence des arguments dans la prise de décision. Le jugement délibéré est le fait de prendre une décision en ayant pris en compte différents arguments qui ne vous feront pas changer d'avis. Pour avoir des données empiriques sur ce modèle, il a été convenu de s'intéresser aux régimes alimentaires proposés dans une cantine.

4.1 Protocole de l'expérience

L'idée est de proposer aux utilisateurs un site web sur lequel l'influence des arguments dans leur assentiment à une cantine végane ou non est suivie. Pour cela le site web proposera des vidéos de deux experts, un en faveur des régimes véganes, l'autre contre ce genre de régimes alimentaires. L'utilisateur aura accès à une bibliothèque de vidéos d'arguments des deux experts. Cette bibliothèque sera dynamique et évoluera en fonction des vidéos vues. Si l'utilisateur a vu l'argument 1 de l'expert A, il aura alors accès à la réponse de l'expert B, le contre-argument. Si ensuite, il visionne ce contre-argument, il aura accès au contre-contre-argument de l'expert A, et ainsi de suite jusqu'à ce que le débat sur cet argument soit clos par l'un des experts. Le site web doit également proposer des formulaires, en fonction des vidéos visionnées, afin de capturer la tendance de l'opinion de l'utilisateur et le moment où celui-ci émettra un jugement délibéré. Ces formulaires permettront d'étudier la puissance des arguments des deux experts et la tendance des utilisateurs dans leur prise de décision. Outre ces formulaires, le site web doit fournir des informations techniques sur le parcours de l'utilisateur, notamment le temps de visionnage des différentes vidéos.

4.2 Création d'un site web

Le site web doit répondre aux exigences du protocole :
- proposer une bibliothèque de vidéos dynamique.
- fournir des formulaires adaptés aux vidéos visionnées.
- récupérer les données de l'utilisateur sur le visionnage des vidéos.

En ce qui concerne le premier et dernier point, ils sont fortement liés. En effet, une fonction pourrait permettre de débloquer la réponse à un argument une fois que l'information "vidéo lue jusqu'à la fin" serait transmise. Pour réaliser cela, *video.js*, lecteur open source, permet d'obtenir de nombreuses données nécessaires pour le projet.

Ce projet a été réalisé avec une étudiante en nutrition, il fallait alors également trouver une interface opensource permettant d'éditer un site web facilement.

4.2.1 WordPress

WordPress est un logiciel de conception web (CMS). Il est utilisé par 35% des sites web dans le monde. WordPress permet de réaliser un site web de qualité sans avoir besoin de compétences techniques importantes (HTML, JavaScript). Il propose différents modèles adaptables facilement et plus de 30 000 extensions permettant d'accéder facilement à de multiples fonctionnalités (vidéos, formulaires, analyse de fréquentations, ...). WordPress semble ainsi être l'interface opensource parfaite pour éditer le site web facilement tout en respectant le protocole.

Une contrainte s'ajoutait tout de même en utilisant WordPress. Le fait de dépendre d'extensions pouvant évoluer et ne plus être compatible avec nos exigeances était en effet un risque à éviter. De plus, le code source est très

8. `https://github.com/barnabegeffroy/vegan_or_not`

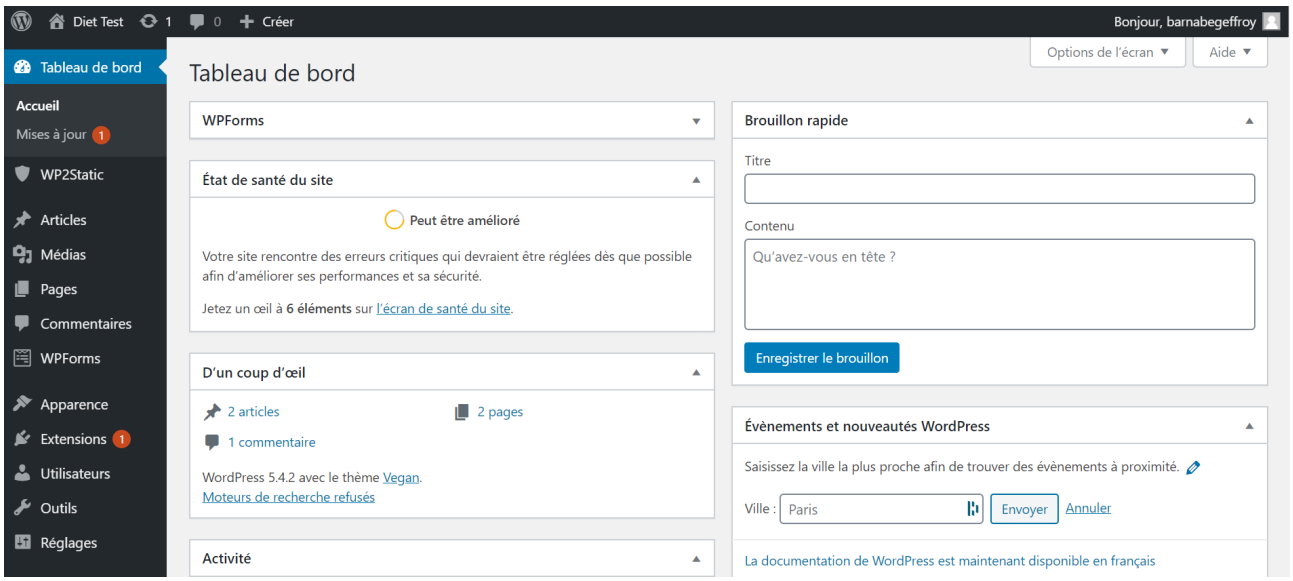

FIGURE 4.1 – Capture d'écran de l'interface de WordPress

peu accessible sur WordPress, il est très compliqué de réaliser un site web sur mesure lorsque celui-ci exige des points techniques très précis.

Il a donc été décidé de concevoir l'esthétique du site sur WordPress (affichage, thème, polices, images). La partie technique (dynamisme, récupération des données) serait développée sur Jekyll.

4.2.2 Jekyll

4.2.2.1 Fonctionnement

Jekyll est un générateur de site statique [9]. Ce logiciel permet de générer facilement une architecture web convaincante. Jekyll propose un système de modèle de page. Celui-ci permet d'obtenir des pages suivant les mêmes caractéristiques (styles, en-tête, pied de page, barre de navigation, ...) sans recopier sur chaque page le code HTML nécessaire. Le code permet également la conversion de fichier Markdown [10], éditable facilement, en page HTML. Chaque fichier doit er par une en-tête lue par Jekyll (voir ci-dessous). Celle-ci contient les informations permettant à Jekyll de générer une page HTML. Le fichier ne peut contenir que cette en-tête, comme ci-dessous.

```
1 ---
2 layout: post
3 title:  "Le premier argmuent de l'expert A!"
4 date:   2020-03-27
5 excerpt: "S1a"
6 image : {{page.image}}
7 ---
```

Contenu d'un fichier suivant le modèle `post` pour générer une page vidéo

Jekyll va, en suivant ces quelques lignes suivrent, générer la page web (voir figure 4.2) à partir du modèles post. L'annexe 3.1 détaille le code HTML généré par Jekyll et permet de se rendre compte de l'efficacité de Jekyll, seulement sept lignes de code ont suffit à générer 190 lignes de code HTML. La page contient une vidéo qui est généré par l'`excerpt` (l.72 à 75).

9. Une page web statique est une page web dont le contenu ne varie pas en fonction des caractéristiques de la demande.
10. Markdown est un langage de balisage offrant une syntaxe facile à lire et à écrire.

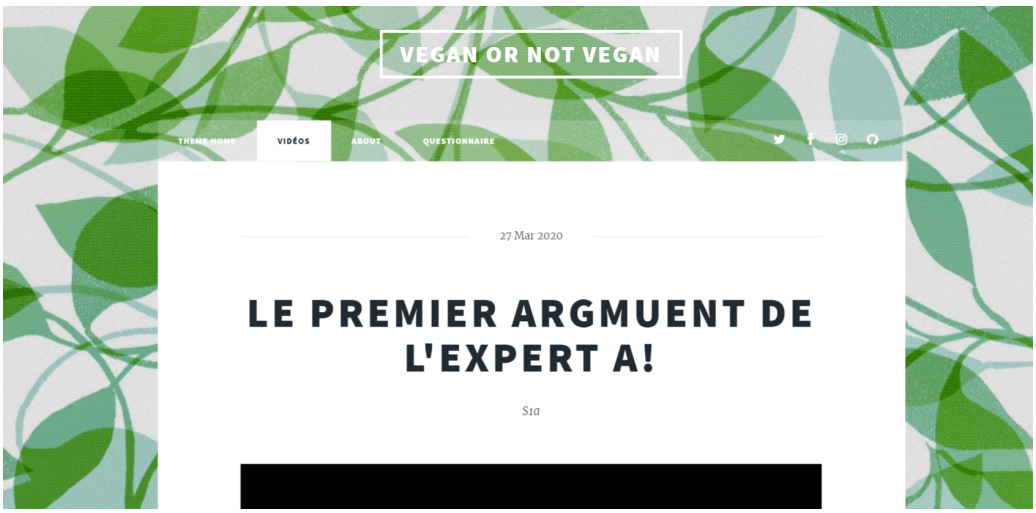

FIGURE 4.2 – Capture d'écran de la page web généré par les lignes de codes ci-dessus

4.2.2.2 Thèmes

Jekyll possède une bibliothèque de thème opensource avec des nombreux modèles préexistants. Le thème de la figure 4.2 avait d'abord été choisi en attendant de récuprer les données esthétiques du site WordPress. Cependant ce thème présentait quelques disfonctionnements et le thème de la figure a finalement été choisi

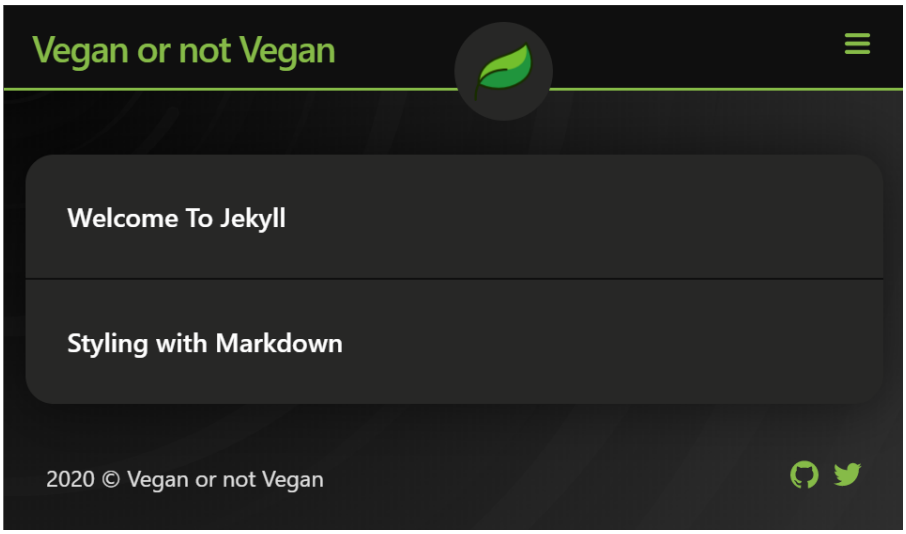

FIGURE 4.3 – Nouveau thème Jekyll utilisé provisoirement dans l'attente de l'esthétique WordPress

4.3 Déploiement sur Internet

4.3.1 Jekyll

Jekyll a été créé par le fondateur de GitHub, Tom Preston-Werner. Son déploiement sur Jekyll est relativement simple. Il suffit de déposer les fichiers générés par Jekyll sur un dépôt GitHub pour que ceux-ci soient déployés sur un site web, une page GitHub.

La génération des fichiers par Jekyll est automatisé de la même manière que plaquette-MIDO avec Travis-CI (voir section 2.1). Le dépôt vers lequel les fichiers sont déployés est vegan_or_not. GitHub va donc créer une page GitHub à partir du contenu de ce dépôt. On obtient ainsi notre site web, Vegan or not vegan.

4.3.2 WordPress

Le déploiement de la page WordPress sur Internet est un peu plus complexe. Pour un déploiement optimal, WordPress préconise de passer par un hébergeur web adapté et payant. Une extension WordPress existe cependant pour convertir le contenu WordPress en statique, lisible par GitHub pour générer une page GitHub. Cette extension s'appelle WP2Static. Elle crée un répertoire contenant les fichiers statiques. Il suffit de pousser ces derniers vers un dépôt GitHub pour que le site web soit déployé. Un exemple de page WordPress déployé sur une page GitHub est disponible sur ce lien.

Remerciements

Je remercie chaudement Olivier Cailloux qui m'a suivi tout au long de ce stage. Premièrement, il a accepté de m'encadrer malgré mon peu d'expérience en développement. Secondement, très pédagogue et à l'écoute, il a su me transmettre son goût pour l'informatique. Le partage de ses cours m'a été d'une grande aide à la compréhension des travaux à réaliser. Je le remercie également pour sa patience dans la relecture de mes codes parfois archaïques et pour nos périodes de riches échanges. Je garderai un très bon souvenir de cette période pratique.

Conclusion

Ce stage m'aura ainsi permis de découvrir davantage le domaine du développement informatique. À travers les différents projets menés, j'ai pu explorer différentes voies du développement : site web, projets Maven/Java, intégration continue... et de maîtriser de nombreux outils et langages informatiques : `Git`, Travis-CI, Jekyll, Maven, WordPress, HTML, JavaScript... Ce stage m'a ainsi fortement conforté dans le choix d'une formation axée sur l'informatique et le développement.

Annexes

Codes Plaquette-MIDO

1 Scripts pour Travis-CI

1.1 writeWSDL.sh

```
#code from https://gist.github.com/cdown/1163649
urlencode() {
    old_lc_collate=$LC_COLLATE
    LC_COLLATE=C

    local length="${#1}"
    for (( i = 0; i < length; i++ )); do
        local c="${1:i:1}"
        case $c in
            [a-zA-Z0-9.~_-]) printf "$c" ;;
            *) printf '%%%02X' "'$c" ;;
        esac
    done

    LC_COLLATE=$old_lc_collate
}

API_username = $(urlencode "${API_username}")
API_password = $(urlencode "${API_password}")

echo https://${API_username}:${API_password}@* > WSDL_login.txt
```

1.2 cibuild.sh

```
#${ACCESS_TOKEN} is a personnal access token from GitHub with the repo
    autorisations
set -e
REPO="receive_plaquette"
FILE_LOG="out.log"
FILE_PDF="out.pdf"
DEPLOY_REPO="https://${ACCESS_TOKEN}@github.com/barnabegeffroy/${REPO}.git"
BUILT_EXIT_CODE=1

function main {
 clean
 get_current_deploy
 build_doc
 if [ -z "${TRAVIS_PULL_REQUEST}" ]; then
     echo "except don't publish doc for pull requests"
    else
  deploy
 fi
}

function clean {
 echo "Cleaning docs."
 if [ -f "${FILE_LOG}" ]; then rm -f "${FILE_LOG}"; fi
 if [ -f "${FILE_PDF}" ]; then rm -f "${FILE_PDF}"; fi
}
```

```
25
26 function get_current_deploy {
27  echo "Getting latest target deployment repository."
28  git clone --depth 1 ${DEPLOY_REPO} "../${REPO}"
29 }
30
31 function build_doc {
32  echo "Trying to generate document."
33  mvn dependency:build-classpath -Dmdep.outputFile=.classpath
34  if java -cp "target/classes:$(cat .classpath)" "io.github.oliviercailloux.
       plaquette_mido_soap.M1ApprBuilder"; then
35   echo "Document generation succeeded."
36   BUILT_EXIT_CODE=0
37  else
38      echo "Document generation failed."
39   BUILT_EXIT_CODE=1
40  fi
41 }
42
43 function deploy {
44  echo "Deploying changes."
45  mv -f "${FILE_LOG}" "../${REPO}"
46  if test -f "${FILE_PDF}"; then
47   mv -f "${FILE_PDF}" "../${REPO}"
48  fi
49  cd "../${REPO}"
50  git config user.name "Travis CI"
51     git config user.email barnabe.geffroy@psl.eu
52  git add "${FILE_LOG}"
53  if test -f "${FILE_PDF}"; then
54   git add "${FILE_PDF}"
55  fi
56  git status
57  git commit -m "Lastest doc built on travis build $TRAVIS_BUILD_NUMBER auto-
       pushed to github (exit code ${BUILT_EXIT_CODE})"
58  git push ${DEPLOY_REPO} master
59  exit ${BUILT_EXIT_CODE}
60 }
61
62 main
```

Codes CredsRead

2 Classe CredsReader

2.1 Méthode readCredentials()

```
1  CredsOpt readCredentials() throws IOException, IllegalStateException {
2   final CredsOpt propertyAuthentication;
3   {
4    final String username = System.getProperty(this.usernameKey);
5    final String password = System.getProperty(this.passwordKey);
6    propertyAuthentication = CredsOpt.given(Optional.ofNullable(username),
        Optional.ofNullable(password));
7    final int informationalValue = propertyAuthentication.
        getInformationalValue();
8    LOGGER.info(
9      "Found {} piece" + (informationalValue >= 2 ? "s" : "") + " of login
          information in properties.",
10     informationalValue);
11  }
12
13  final CredsOpt envAuthentication;
14  {
15   final String username = env.get(this.usernameKey);
16   final String password = env.get(this.passwordKey);
17   envAuthentication = CredsOpt.given(Optional.ofNullable(username),
        Optional.ofNullable(password));
18   final int informationalValue = envAuthentication.getInformationalValue();
19   LOGGER.info("Found {} piece" + (informationalValue >= 2 ? "s" : "")
20     + " of login information in environment variables.", informationalValue
          );
21  }
22
23  final CredsOpt fileAuthentication;
24  {
25   final Optional<String> optUsername;
26   final Optional<String> optPassword;
27   final Path path = filePath;
28   if (!Files.exists(path)) {
29    optUsername = Optional.empty();
30    optPassword = Optional.empty();
31   } else {
32    final List<String> lines = Files.readAllLines(path);
33    final Iterator<String> iterator = lines.iterator();
34    if (iterator.hasNext()) {
35     optUsername = Optional.of(iterator.next());
36    } else {
37     optUsername = Optional.of("");
38    }
39    if (iterator.hasNext()) {
40     optPassword = Optional.of(iterator.next());
41    } else {
42     optPassword = Optional.of("");
43    }
44    while (iterator.hasNext()) {
45     if (!iterator.next().isEmpty()) {
46      throw new IllegalStateException(
```

```
47          "File " + filePath + " is too long: " + lines.size() + " lines");
48        }
49       }
50      }
51      fileAuthentication = CredsOpt.given(optUsername, optPassword);
52      final int informationalValue = fileAuthentication.getInformationalValue()
            ;
53      LOGGER.info("Found {} piece" + (informationalValue >= 2 ? "s" : "") + "
            of login information in file.",
54        informationalValue);
55    }
56
57    final TreeMap<Double, CredsOpt> map = new TreeMap<>();
58    map.put(propertyAuthentication.getInformationalValue() * 1.2d,
            propertyAuthentication);
59    map.put(envAuthentication.getInformationalValue() * 1.1d,
            envAuthentication);
60    map.put(fileAuthentication.getInformationalValue() * 1.0d,
            fileAuthentication);
61    return map.lastEntry().getValue();
62  }
```

2.2 Méthode getCredentials()

```
1   public Credentials getCredentials() throws IllegalStateException,
        UncheckedIOException {
2    final CredsOpt credsOpt;
3    try {
4     credsOpt = readCredentials();
5    } catch (IOException e) {
6     throw new UncheckedIOException(e);
7    }
8
9    if (credsOpt.getUsername().isEmpty() && credsOpt.getPassword().isEmpty())
         {
10    throw new IllegalStateException("Login information not found.");
11   }
12   if (credsOpt.getUsername().isEmpty()) {
13    throw new IllegalStateException("Found password but no username.");
14   }
15   if (credsOpt.getPassword().isEmpty()) {
16    throw new IllegalStateException("Found username but no password.");
17   }
18   final Credentials credentials = Credentials.given(credsOpt.getUsername().
         get(), credsOpt.getPassword().get());
19   return credentials;
20  }
```

Codes Jugement Délibéré

3 Jekyll

3.1 Exemple de code HTML généré par Jekyll

```
1
2  <!DOCTYPE HTML>
3  <!--
4   Massively by HTML5 UP
5   html5up.net | @ajlkn
6    Jekyll integration by somiibo.com
7   Free for personal and commercial use under the CCA 3.0 license (html5up.net
       /license)
8  -->
9  <html>
10  <head>
11      <meta charset="utf-8" />
12 <meta name="viewport" content="width=device-width, initial-scale=1, user-
       scalable=no" />
13
14 <title>Le premier argmuent de l'expert A!</title>
15 <meta name="description" content="">
16
17 <link rel="apple-touch-icon" sizes="180x180" href="http://localhost:4000/
       vegan_or_not/assets/icon/apple-touch-icon.png">
18 <link rel="icon" type="image/png" sizes="32x32" href="http://localhost:4000/
       vegan_or_not/assets/icon/favicon-32x32.png">
19 <link rel="icon" type="image/png" sizes="16x16" href="http://localhost:4000/
       vegan_or_not/assets/icon/favicon-16x16.png">
20 <link rel="manifest" href="http://localhost:4000/vegan_or_not/assets/icon/
       manifest.json">
21 <link rel="mask-icon" href="http://localhost:4000/vegan_or_not/assets/icon/
       safari-pinned-tab.svg" color="#5bbad5">
22 <link rel="shortcut icon" href="http://localhost:4000/vegan_or_not/assets/
       icon/favicon.ico">
23 <meta name="msapplication-config" content="http://localhost:4000/
       vegan_or_not/assets/icon/browserconfig.xml">
24 <meta name="theme-color" content="#ffffff">
25 <link href="http://localhost:4000/vegan_or_not/assets/css/video-js.css" rel=
       "stylesheet" />
26   <script src="http://localhost:4000/vegan_or_not/assets/js/videojs-ie8.min.
         js" ></script>
27 <!-- CSS -->
28 <link rel="stylesheet" href="http://localhost:4000/vegan_or_not/assets/css/
       main.css" />
29 <noscript><link rel="stylesheet" href="http://localhost:4000/vegan_or_not/
       assets/css/noscript.css" /></noscript>
30
31  </head>
32  <body class="is-loading">
33
34   <!-- Wrapper -->
35    <div id="wrapper" class="fade-in">
36
37     <!-- Header -->
38         <header id="header">
```

```
39            <a href="http://localhost:4000/vegan_or_not/" class="logo">Vegan
                 or not Vegan</a>
40          </header>
41
42      <!-- Nav -->
43       <nav id="nav">
44
45              <ul class="links">
46    <li class=""><a href="http://localhost:4000/vegan_or_not/">Theme Home</a><
        /li>
47    <li class=" active "><a href="http://localhost:4000/vegan_or_not/videos/">
        Videos</a></li>
48    <li class=""><a href="http://localhost:4000/vegan_or_not/about/">About</a>
        </li>
49    <li class=""><a href="http://localhost:4000/vegan_or_not/form/">
        Questionnaire</a></li>
50  </ul>
51
52  <ul class="icons">
53    <li><a href="https://twitter.com/default" class="icon fa-twitter" rel="
        nofollow"><span class="label">Twitter</span></a></li>
54    <li><a href="https://facebook.com/default" class="icon fa-facebook" rel="
        nofollow"><span class="label">Facebook</span></a></li>
55    <li><a href="https://instagram.com/default" class="icon fa-instagram" rel=
        "nofollow"><span class="label">Instagram</span></a></li>
56    <li><a href="https://github.com/default" class="icon fa-github" rel="
        nofollow"><span class="label">GitHub</span></a></li>
57  </ul>
58
59
60       </nav>
61
62      <!-- Main -->
63      <div id="main">
64            <section class="post">
65          <header class="major">
66            <span class="date">27 Mar 2020</span>
67            <h1>Le premier argmuent de l'expert A!</h1>
68            <p>S1a</p>
69           </header>
70           <div id="testdiv"></div>
71              <div align="center">
72              <video autoplay="" controls=""  class="video-js vjs-default-skin
                   " id="S1a" width="100%" height="100%" data-setup='{}' poster=
                   "{}">
73                  <source src="/vegan_or_not/videos/S1a.webm" type='video/webm
                       '>
74                  <source src="/vegan_or_not/videos/S1a.mp4" type='video/mp4'>
75              </video>
76              </div>
77           <p>
78  </p>
79          </section>
80
81            <div class="comments-wrapper">
82            <div id="disqus_thread"></div>
83            <script>
84                /**
85                 *  RECOMMENDED CONFIGURATION VARIABLES: EDIT AND UNCOMMENT
                       THE SECTION BELOW TO INSERT DYNAMIC VALUES FROM YOUR
                       PLATFORM OR CMS.
```

```
86                 *  LEARN WHY DEFINING THESE VARIABLES IS IMPORTANT: https://
                      disqus.com/admin/universalcode/#configuration-variables
87                 */
88
89                 var disqus_config = function () {
90                     this.page.url = '/videos/s1a/';  /*Replace PAGE_URL with
                          your page's canonical URL variable*/
91                     this.page.identifier = '/videos/s1a/'; /*Replace
                          PAGE_IDENTIFIER with your page's unique identifier
                          variable*/
92                     this.page;image = "http://localhost:4000/vegan_or_not/
                          images/pic02.jpg"
93                 };
94
95                 (function() {  /* dont endit below this line */
96                     var d = document, s = d.createElement('script');
97
98                     s.src = 'https://default.disqus.com/embed.js';
99
100                    s.setAttribute('data-timestamp', +new Date());
101                    (d.head || d.body).appendChild(s);
102                })();
103            </script>
104            <noscript>Please enable JavaScript to view the <a href="https://
                  disqus.com/?ref_noscript" rel="nofollow">comments powered by
                  Disqus.</a></noscript>
105         </div><!-- /.comments-wrapper -->
106
107
108       <!-- Footer -->
109        <footer>
110                <ul class="actions">
111                  <li><a href="http://localhost:4000/vegan_or_not/videos/"
                         class="button">Nos videos</a></li>
112                </ul>
113        </footer>
114       </div>
115
116     <!-- Footer -->
117         <footer id="footer">
118   <section>
119     <form method="POST" action="https://api.slapform.com/barnabe.geffroy@psl
          .eu">
120       <div class="field">
121         <label for="name">Name</label>
122         <input type="text" name="name" id="name" />
123       </div>
124       <div class="field">
125         <label for="email">Email</label>
126         <input type="email" name="slap_replyto" id="email" /> <!--
              slap_replyto will set the reply-to as the submitter's email! -->
127       </div>
128       <div class="field">
129         <label for="message">Message</label>
130         <textarea name="message" id="message" rows="3"></textarea>
131       </div>
132       <ul class="actions">
133         <li><input type="submit" value="Send Message" /></li>
134       </ul>
135       <input type="hidden" name="slap_redirect" value="http://localhost
             :4000/thank-you" /> <!-- slap_redirect allows you to set a custom
```

```
            redirect/thank you page -->
136       </form>
137     </section>
138     <section class="split contact">
139       <section class="alt">
140         <h3>Location</h3>
141         <p>Paris, France</p>
142       </section>
143       <section>
144         <h3>Phone</h3>
145         <p><a href="tel:xxxxxxxxxxx">xxxxxxxxxxx</a></p>
146       </section>
147       <section>
148         <h3>Email</h3>
149         <p><a href="mailto:barnabe.geffroy@psl.eu">barnabe.geffroy@psl.eu</a><
              /p>
150       </section>
151       <section>
152         <h3>Social</h3>
153         <ul class="icons alt">
154           <li><a href="https://twitter.com/default" class="icon fa-twitter"
                 rel="nofollow"><span class="label">Twitter</span></a></li>
155           <li><a href="https://facebook.com/default" class="icon fa-facebook"
                 rel="nofollow"><span class="label">Facebook</span></a></li>
156           <li><a href="https://instagram.com/default" class="icon fa-instagram
                 " rel="nofollow"><span class="label">Instagram</span></a></li>
157           <li><a href="https://github.com/default" class="icon fa-github" rel=
                 "nofollow"><span class="label">GitHub</span></a></li>
158         </ul>
159       </section>
160     </section>
161   </footer>
162   <!-- Copyright -->
163   <div id="copyright">
164     <ul><li>&copy; HTML5 UP</li><li>Design by <a href="https://html5up.net"
          rel="nofollow">HTML5 UP</a></li><li>Jekyll Integration by <a href="
          https://slapform.com">Slapform</a> </li></ul>
165   </div>
166
167
168     </div>
169
170         <!-- Scripts -->
171       <!-- DYN -->
172   <script src="http://localhost:4000/vegan_or_not/assets/js/jquery.min.js"></
        script>
173   <script src="http://localhost:4000/vegan_or_not/assets/js/jquery.scrollex.
        min.js"></script>
174   <script src="http://localhost:4000/vegan_or_not/assets/js/jquery.scrolly.min
        .js"></script>
175   <script src="http://localhost:4000/vegan_or_not/assets/js/skel.min.js"></
        script>
176   <script src="http://localhost:4000/vegan_or_not/assets/js/util.js"></script>
177   <script src="http://localhost:4000/vegan_or_not/assets/js/main.js"></script>
178
179     <!-- Global site tag (gtag.js) - Google Analytics -->
180   <script async src="https://www.googletagmanager.com/gtag/js?id=UA
        -166631473-1"></script>
181   <script>
182     window.dataLayer = window.dataLayer || [];
183     function gtag(){dataLayer.push(arguments);}
```

```
184     gtag('js', new Date());
185
186     gtag('config', 'UA-166631473-1');
187   </script>
188
189    </body>
190   </html>
```

Bibliographie

[1] O. Cailloux et Y. Meinard, "A formal framework for deliberated judgment," *Theory and Decision*, vol. 88, n°. 2, p. 269–295, mars 2020.

[2] O. Cailloux, "Git," Université Paris-Dauphine, sept. 2018.

[3] ——, "Maven," Université Paris-Dauphine, mars 2020.

[4] O. Cailloux, Y. Meinard et N. Salliou, "Deliberated diet," juin 2020.

[5] S. Airiau, "Introduction à la programmation en Java - Cours 11," Université Paris-Dauphine, p. 20–45, déc. 2019.

[6] S. Chacon et B. Straub, *Pro Git*, 2e éd. New York, NY : Apress, nov. 2014.

[7] "Bash Reference Manual." [En ligne]. Disponible : https://www.gnu.org/software/bash/manual/bash.html

[8] "Jekyll Docs Quickstart." [En ligne]. Disponible : https://jekyllrb.com/docs/