

CPES 3

---

# Rapport de stage

## Développement agile d'applications web

---

*Auteur :*  
Barnabé GEFFROY

*Référent :*  
Olivier CAILLOUX

17 juin 2020

# Table des matières

<b>Preliminaires</b>	<b>1</b>
<b>1 Gestion des presences</b>	<b>2</b>
1.1 Attendance . . . . .	2
1.2 JeSuisEnCours . . . . .	2
<b>2 Plaquette-MIDO</b>	<b>3</b>
2.1 L'authentification . . . . .	3
2.2 L'automatisation du lancement du code . . . . .	3
<b>3 CredsRead</b>	<b>6</b>
3.1 Diagramme de classe . . . . .	6
<b>4 Jugement delibere</b>	<b>7</b>
 <b>Annexes</b>	 <b>ii</b>
<b>Annexe 1</b>	<b>ii</b>
1 Partie 1 . . . . .	ii
<b>Annexe 2</b>	<b>iv</b>

# Introduction

# Préliminaires

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

## 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. La connexion internet est cependant nécessaire pour envoyer ces modifications sur le dépôt distant. Ce genre de sauvegarde est appelé *commit*. Celle-ci est une version du code à instant donné. `Git` crée, avec tous les commits, une série d'instantanés qui rend la perte d'information très difficile.

`Git` gère également l'intégrité du code. Il peut y avoir des conflits, des parties identiques du code modifiées par deux utilisateurs. `Git` pointe 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 (ajouté 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

## GitHub

GitHub est un hébergeur de dépôts `Git`. Il offre la gestion de version distribuée, la fonctionnalité de gestion de code source de `Git`, ainsi que 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, 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
```

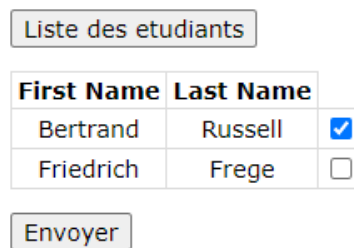
# Projet 1

## Gestion des présences<sup>1</sup>

L'offre de stage<sup>2</sup> auquel 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 présences.

### 1.1 Attendance

Le projet Attendance avait donc pour but de développer une application web liée à la gestion des présences des élèves. Le serveur HTTP Eclipse Jetty,



The screenshot shows a web interface for managing student attendance. At the top is a button labeled 'Liste des etudiants'. Below it is a table with two columns: 'First Name' and 'Last Name'. The table contains two rows of data: 'Bertrand Russell' and 'Friedrich Frege'. To the right of each row is a checkbox. The checkbox for 'Bertrand Russell' is checked, while the one for 'Friedrich Frege' is unchecked. Below the table is a button labeled 'Envoyer'.

First Name	Last Name	
Bertrand	Russell	<input checked="" type="checkbox"/>
Friedrich	Frege	<input type="checkbox"/>

FIGURE 1.1: Enveloppe d'un son

### 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 donc été décidé d'abandonner le projet intiale Attendance pour se tourner vers une application JeSuisEnCours, spécialisée dans la digitalisation les feuilles de présences.

Le but principal du projet est donc 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 ralenti 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](#)

## Projet 2

# Plaquette-MIDO<sup>3</sup>

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 du 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 était 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 dans le code était nécessaire pour qu'un autre utilisateur puisse lancer Plaquette-MIDO. Il fallait donc davantage généraliser le code de façon à rendre accessible le code à d'autres utilisateurs. Le principal aspect à généraliser était l'authentification à l'API de Dauphine, indispensable pour avoir accès aux données de l'université.

### 2.1 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 différentes 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

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

#### 2.1.1 La classe Authentication

Une nouvelle classe a donc 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'instancier aussi bien la valeur d'une chaîne de caractère que l'absence d'une information. Il permet ainsi de

#### 2.1.2 Le projet CredsRead

La généralisation du code permettant l'authentification nous a poussé à le séparer dans un projet bien distinct de Plaquette-MIDO, le projet CredsRead. Celui-ci sera ensuite intégré au code source de Plaquette-MIDO.

### 2.2 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.

#### 2.2.1 Travis-CI

Travis-CI est 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

---

3. <https://github.com/Dauphine-MIDO/plaquette-MIDO>

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 donc 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 peut donc exécuter le code source permettant la création du fichier PDF et ensuite déployer ce fichier vers un dépôt GitHub.

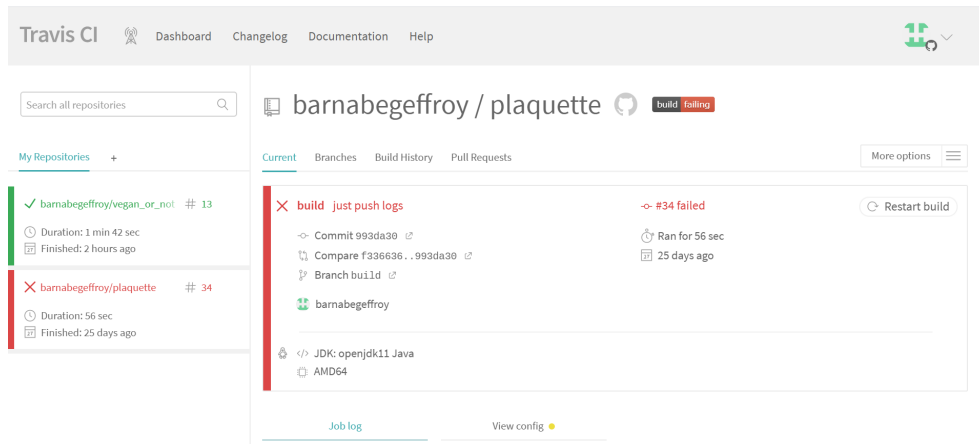


FIGURE 2.1: Enveloppe d'un son

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

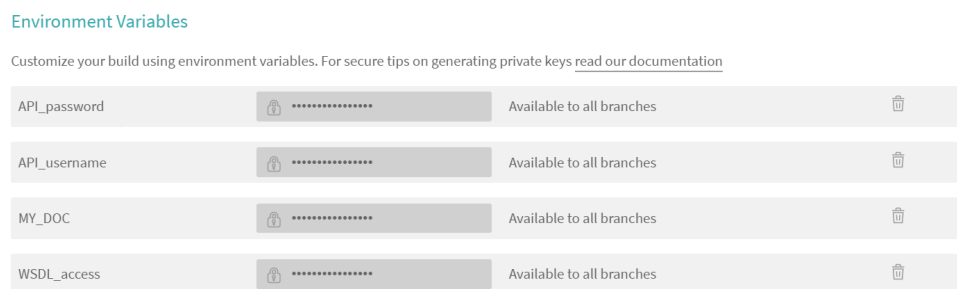


FIGURE 2.2: Enveloppe d'un son

## 2.2.2 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 des identifiants de l'utilisateur. Ce fichier ne peut pas être dans le dépôt car il contient des informations personnelles. Il faut donc un script qui crée le fichier sur la machine virtuelle de Travis-CI. Avant de lancer l'installation des dépendances le fichier doit donc être créé. Dans le `.travis.yml`, on peut ajouter un script qui génère ce fichier.

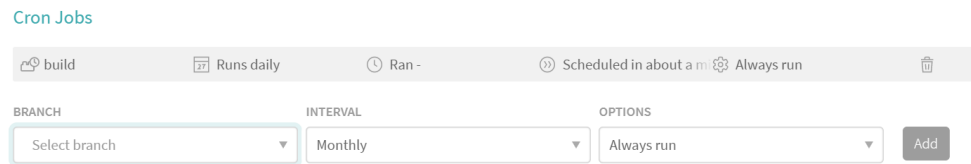
### Création du fichier PDF et déploiement

Une fois que toutes les dépendances du projet Maven sont installées, il faut exécuter le code source qui génère le fichier PDF. La méthode `main` a été exécutée se trouve dans la classe `M1ApprBuilder.java`. Le déploiement vers le dépôt d'arrivée doit également être pris en compte. Voici le script exécuté par Travis-CI pour réaliser ce travail :

### Automatisation

Travis-CI lance initialement la construction du code 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.



The screenshot shows the 'Cron Jobs' section of the Travis CI interface. At the top, there's a header with 'Cron Jobs' in blue. Below it, a horizontal bar contains several status indicators: a build icon, 'Runs daily', 'Ran -', 'Scheduled in about a m', 'Always run', and a trash icon. The main form has three sections: 'BRANCH' with a dropdown menu showing 'Select branch', 'INTERVAL' with a dropdown menu showing 'Monthly', and 'OPTIONS' with a dropdown menu showing 'Always run'. To the right of these sections is an 'Add' button.

FIGURE 2.3: Enveloppe d'un son

En programmant sur *Daily*, Travis-CI va relancer la construction du code tous les jours et ainsi renouveler 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 prévendra de la non-construction du code et le fichier PDF le plus récent sera toujours disponible sur le site.



## Projet 3

# CredsRead<sup>4</sup>

CredsRead, pour Credentials Read, gère comme son nom l'indique la lecture des identifiants d'un utilisateur.

### 3.1 Diagramme de classe

Le diagramme de classe permet de 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.

Le diagramme

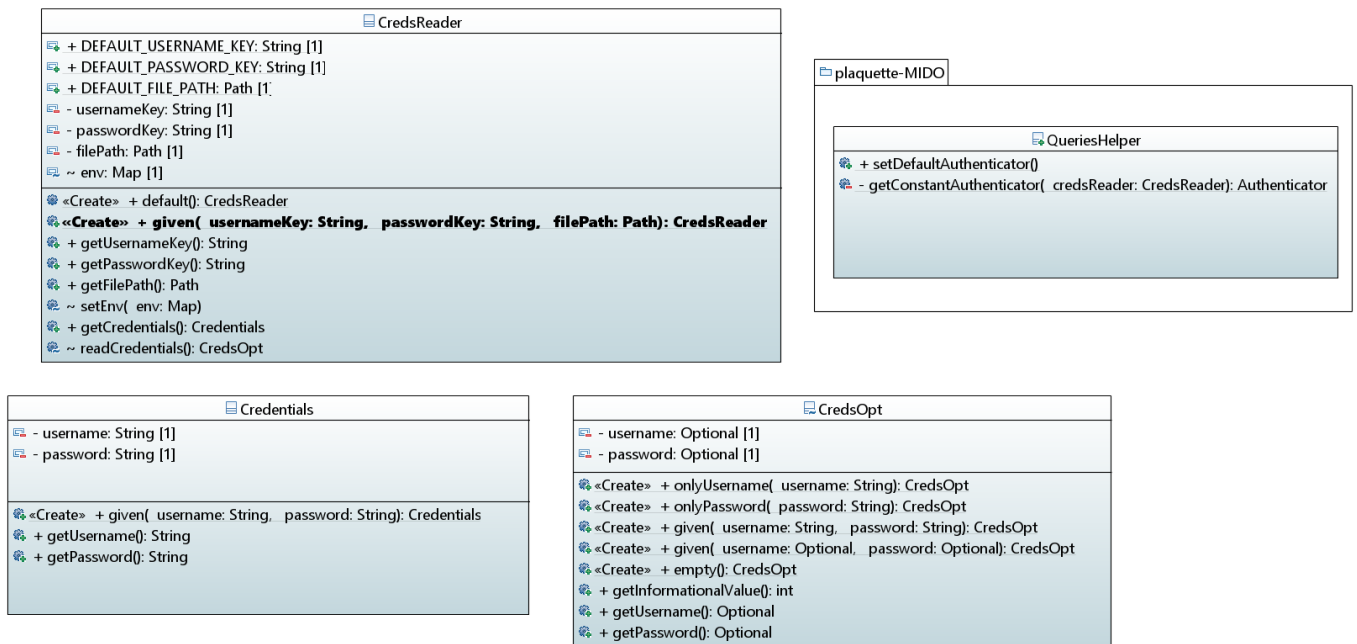


FIGURE 3.1: Enveloppe d'un son

4. <https://github.com/oliviercailloux/creds-read>

## Projet 4

# Jugement délibéré<sup>5</sup>

---

5. [https://github.com/barnabegeffroy/vegan\\_or\\_not](https://github.com/barnabegeffroy/vegan_or_not)

# Conclusion

# Annexes

# Annexe 1

Intro

## 1 Partie 1

Bla

**writeWSDL.sh**

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

**cibuild.sh**

```
1  ##${ACCESS_TOKEN} is a personal access token from GitHub with the repo
    autorisations
2  set -e
3  REPO="receive_plaquette"
4  FILE_LOG="out.log"
5  FILE_PDF="out.pdf"
6  DEPLOY_REPO="https://${ACCESS_TOKEN}@github.com/barnabegeffroy/${REPO}.git"
7  BUILT_EXIT_CODE=1
8
9  function main {
10   clean
11   get_current_deploy
12   build_doc
13   if [ -z "${TRAVIS_PULL_REQUEST}" ]; then
14     echo "except don't publish doc for pull requests"
15   else
16     deploy
17   fi
18 }
19
20 function clean {
21   echo "Cleaning docs."
22   if [ -f "${FILE_LOG}" ]; then rm -f "${FILE_LOG}"; fi
23   if [ -f "${FILE_PDF}" ]; then rm -f "${FILE_PDF}"; fi
24 }
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

```

## Annexe 2

# Bibliographie

- [1] Bash reference manual.
- [2] Jekyll theme for documentation. page 190.
- [3] Stéphane Airiau. Introduction à la programmation en java - cours 11. page 46.
- [4] Olivier Cailloux. Git.
- [5] Olivier Cailloux. Maven.
- [6] Olivier Cailloux and Yves Meinard. A formal framework for deliberated judgment. 88(2) :269–295.
- [7] Olivier Cailloux, Yves Meinard, and Nicolas Salliou. Deliberated diet.