Institut Universitaire de Technologie,
Aix-Marseille Université

RAPPORT DE STAGE
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Conception et réalisation d'un site web

Damien MATHIEU

Jalis

Responsable entreprise : Jean-Christophe DUVIVIER
Responsable académique : Éric WÜRBEL

2019

Table des matières

1. Introduction 6
2. Présentation de l'entreprise 6
3. Introduction au projet 7
3.1 Les bases de création d'un site web 7
3.2 Cahier des charges 9
4. Le projet 9
4.1 Programmer en Orienté Objet 9
4.1.1 Présentation du principe, de l'architecture MVC 9
4.1.2 Travail réalisé 11
4.1.3 Problèmes rencontrés 13
4.2 Séparation des langages, et conception du texte dynamique 14
4.2.1 Présentation 14
4.2.2 Description du travail réalisé 14
4.2.3 Problèmes rencontrés 18
4.2.4 Pour aller plus loin 19
4.3 Communication avec la base de données 19
4.3.1 Comparaison des méthodes disponibles 19
4.3.2 Description du travail réalisé 20
4.4 Les permissions 20
4.4.1 Présentation 20
4.4.2 Description du travail réalisé 21
4.5 Conclusion sur le projet 22
5. Améliorations des outils de développement 22
5.1 Présentation 22
5.2 Description du travail réalisé 22
5.3 Conclusion sur le projet 23
6. Conclusion 25
7. Remerciements 27
8. Sitographie 29

1. Introduction

Du 8 avril au 14 juin, j'ai effectué un stage au sein de l'entreprise Jalis, située à Marseille. L'entreprise Jalis est spécialisée dans la création de sites web et le référencement.
Au cours de ce stage au département développement, j'ai pu m'intéresser à la conception de site web de manière professionnelle.
Ce stage m'a donc beaucoup apporté puisqu'il est en accord avec mon projet professionnel qui est de devenir développeur web.
Mon stage dans le département développement a consisté, en grande partie, à l'élaboration d'un projet de site web recensant des candidatures. Ce projet m'a permis de manipuler tous les aspects du développement web. Puis, un second projet qui porte sur le développement d'outils internes. Ce projet a mis en avant le fonctionnement du travail en équipe sur un même projet.

Ce stage m'a permis d'enrichir mes connaissances et également de découvrir de nouvelles méthodes et technologies, de nouveaux logiciels et de nouveaux langages de programmation. La découverte de ces nouvelles notions a été facilitée grâce aux aides et conseils de toute l'équipe de développeurs. Notamment grâce à mon maître de stage, Jean-Christophe DUVIVIER, qui m'a supervisé.
Ce stage m'a également permis de comprendre comment fonctionne une entreprise ayant un grand nombre d'employés (plus d'une centaine). Et comment les personnes de différents départements travaillent ensemble afin de mener un projet à bien.

Dans ce rapport, nous allons d'abord présenter l'entreprise plus en détail, puis une petite partie sera dédiée aux bases du fonctionnement de site web pour les néophytes, enfin je présenterai le cahier des charges. La suite du rapport consistera à expliquer quels ont été les choix faits afin de satisfaire le cahier des charges, les problèmes rencontrés et les solutions apportées. Enfin, la dernière partie du rapport sera consacrée au second projet réalisé.

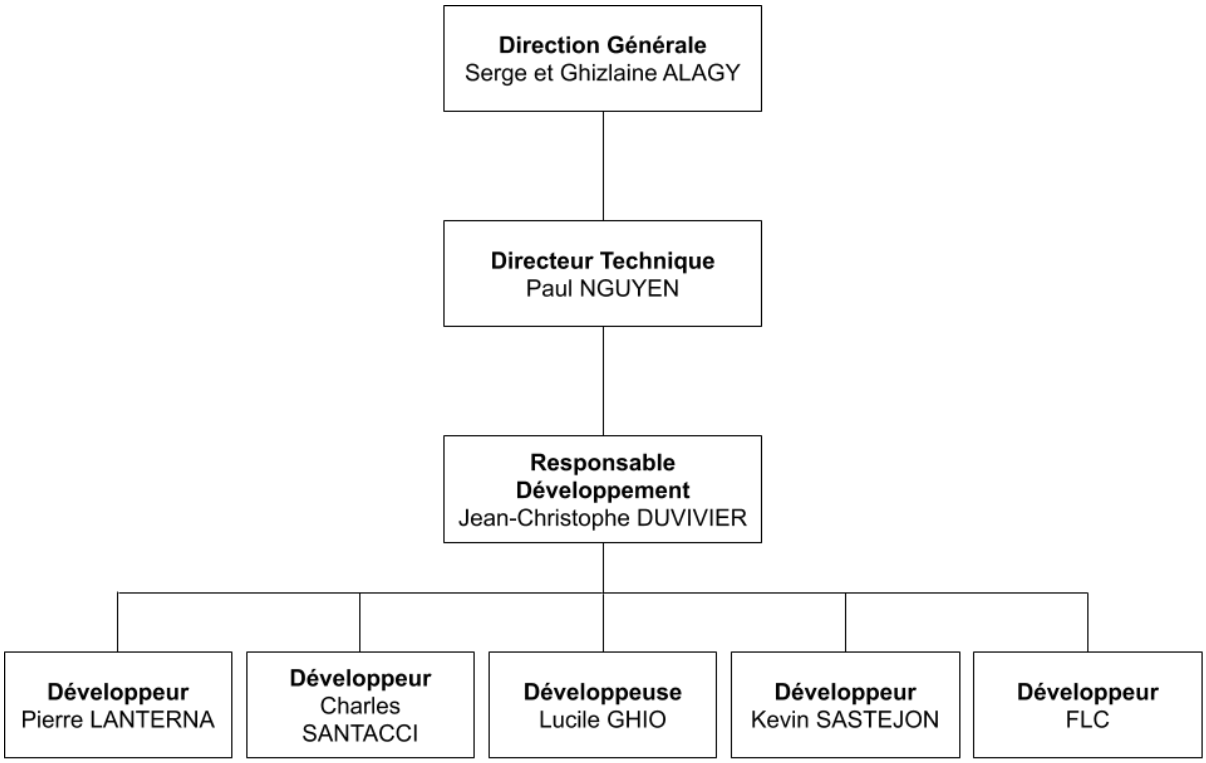
2. Présentation de l'entreprise

Jalis est une entreprise de conception de site web et de référencement, créée en 2002, dont la clientèle est composée exclusivement de professionnels. C'est ce qu'on appelle du B2B, abréviation de Business To Business, que l'on peut traduire par commerce inter-entreprises. Jalis a des clients dans tous les secteurs d'activités : automobile, immobilier, restauration … Les clients sont aussi bien des TPE (Très Petite Entreprise), des PME (Petite et Moyenne Entreprise) et des GE (Grandes Entreprises).

Jalis en quelques chiffres :
- 17 ans d'ancienneté
- 4500 clients
- 5 millions de visites, sur les sites créés, par mois
- 6 agences en France
- 127 certifications Google

La force des sites Jalis est en particulier le référencement, en effet, comme le montre le chiffre des 127 certifications Google, Jalis assure un référencement pour le moteur de recherche Google. D'ailleurs, l'entreprise est certifiée « Google Partner ».
Jalis compte environ 140 employés, répartis dans 8 branches : référencement, graphisme, développement, système d'informations (qui s'occupe des serveurs pour les sites clients), support client, administratif, chefs de projets et commercial.

Jalis accompagne tous ses clients, dès le début du contrat, en faisant une liste de leurs exigences, ce qu'ils veulent mettre en avant sur leurs sites, jusqu'après le lancement du site en assurant les maintenances et en analysant le trafic généré sur le site.
La culture d'entreprise repose sur le partage, des équipes qui travaillent en synergie pour assurer une qualité optimale. Puisque le secteur de l'informatique est en constante évolution, les employés sont continuellement en train de se renseigner sur les nouvelles technologies qui apparaissent. Cela permet de se renouveler tout en proposant un service à la pointe de la technologie.
Le siège social se trouve à Marseille, c'est là que j'ai réalisé mon stage.
Voici l'organigramme des développeurs :



3. Introduction au projet

3.1 Les bases de création d'un site web

Tout d'abord, nous allons parler des bases. Les sites web sont écrits en HTML, HyperText Markup Language et CSS, Cascading Style Sheets. Il faut noter que ce ne sont pas des langages de programmation, HTML est un langage de balisage et CSS permet le contrôle de la mise en page des éléments d'un document. Les fichiers HTML et CSS sont indépendants.
HTML va seulement servir à afficher le contenu de votre site web, par exemple, les titres, les articles, le pied de page, en somme tout le texte visible par l'utilisateur sera contenu dans le fichier HTML.
CSS va quant à lui, mettre en page tout le contenu présent dans le HTML. CSS aussi appelé feuille de style, permet de changer la taille des images, changer la couleur de la police, ajouter un arrière-plan, en somme tout effet visuel d'un site est défini dans le fichier CSS.
En d'autres termes, HTML et CSS sont respectivement le fond et la forme d'un site web.

Pour que les navigateurs affichent correctement les sites, il existe des standards de développement, aujourd'hui ce sont HTML5 et CSS 3. Le nombre correspond simplement à la version du standard.

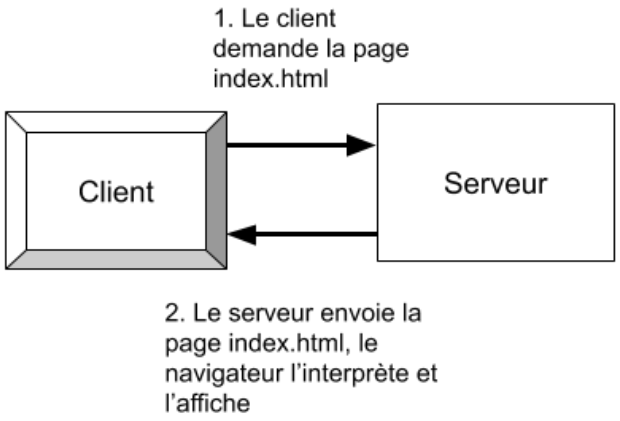


Figure 1 : Schéma simplifié des actions entre le serveur et le client lors d'une requête de page HTML

Cependant, d'autres langages existent et permettent de nombreuses fonctionnalités comme créer des sites dynamiques et pouvoir accéder à une base de données. C'est le cas du langage qui nous intéresse ici : PHP, PHP Hypertext Preprocessor. En effet, dès le début de mon stage, ma première tâche a été de faire un projet : développer un site web en PHP, communicant avec une base de données et qui a pour but de recenser des candidatures. PHP est un langage de script.

Quelle est donc la différence entre PHP et HTML ? HTML est comme on dit « statique », c'est-à-dire que le script HTML renvoie exactement ce qu'on écrit. PHP quant à lui, permet de créer des sites dynamiques, c'est-à-dire que le code est exécuté du côté serveur, qui génère ensuite le code HTML. Pour que cela fonctionne, il faut simplement installer PHP sur le serveur et coder en PHP. En 2018, 80% des sites web utilisent le langage PHP.

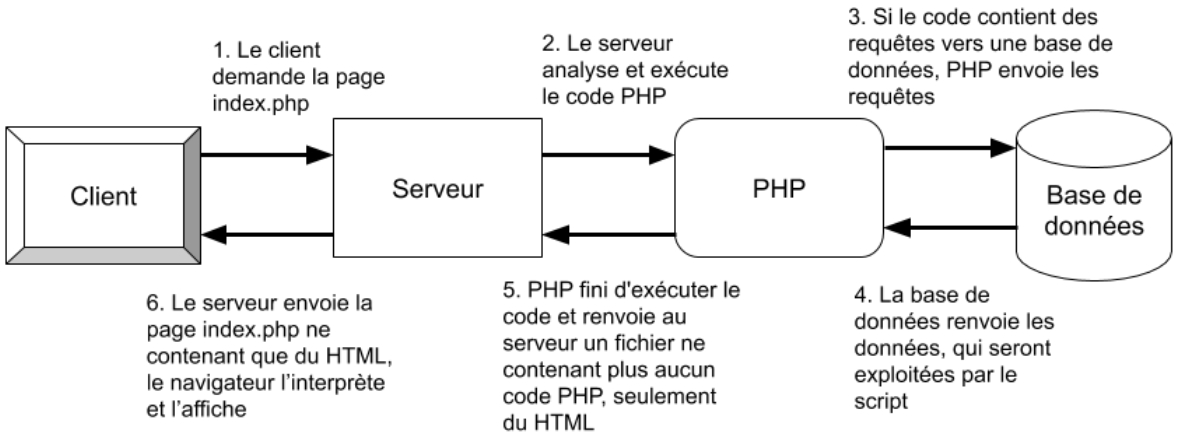


Figure 2 : Schéma simplifié des actions entre le serveur et le client lors d'une requête de page PHP

3.2 Cahier des charges

Voici le cahier des charges qui m'a été donné pour la création du site :

- Le site contiendra 1 seul fichier (index.php) mais présentera 5 pages différentes :
    - Une page qui affiche un tableau de toutes les candidatures avec quelques informations concernant les candidatures.
    - Une page qui affiche toutes les informations liées à la candidature sélectionnée.
    - Une page permettant de créer une candidature.
    - Une page permettant de modifier une candidature.
    - Une page qui permet de se connecter avec un login et un mot de passe
- Chaque candidature comporte ces informations : Nom, prénom, id, adresse, email, numéro de téléphone, photo et C.V.
- Le site doit être mis en forme avec le CSS de Bootstrap.
- Le site doit être développé en POO, Programmation Orientée Objet, avec l'architecture MVC.
- La base de données est une base de données MySQL
- Le site doit séparer le code PHP du code HTML.
- Tous les textes du site doivent être dynamiques.

Des compétences sont également nécessaires : savoir se servir d'un moteur de template, connaître la Programmation Orientée Objet et connaître l'architecture MVC.

Dans la suite du rapport, nous allons voir quelles ont été les techniques utilisées pour satisfaire le cahier des charges, quels ont été les problèmes rencontrés et quelles solutions ont été apportées.

4. Le projet

4.1 Programmer en Orienté Objet

4.1.1 Présentation du principe, de l'architecture MVC

Ici, je vais rapidement présenter le principe de la POO, Programmation Orientée Objet et le principe de l'architecture MVC, car ce sont des techniques très utilisées et elles apparaissent dans le cahier des charges. Et j'expliquerai pourquoi elles sont requises et particulièrement adaptées pour notre site web.

La POO, ou Programmation Orientée Objet, est une manière de coder apparue dans les années 1970. En POO, tout est objet et chaque objet possède des caractéristiques et des capacités.
Prenons comme exemple une voiture, c'est un objet qui a des caractéristiques comme son modèle, sa marque, son kilométrage, etc., et des fonctions comme démarrer, rouler, s'arrêter, etc. C'est comme cela qu'un objet est défini en POO. En fait en POO, les caractéristiques sont appelées **attributs** et les fonctions sont appelées **méthodes.**
Un des avantages de la POO est le fait de pouvoir créer une infinité d'objets de manière autonome, grâce à ce qu'on appelle des **classes.** En effet, c'est à l'intérieur de ces classes que nous allons définir tout ce dont nos objets auront besoin : leurs attributs et leurs méthodes.
Pour continuer avec notre exemple, c'est comme si la classe était l'usine qui fabriquait nos voitures, on commande une voiture d'un certain modèle et d'une certaine marque, et l'usine crée cet objet.

Notre site ne va manipuler qu'un seul type de donnée : des candidatures (qui sont des objets) et avec cela, on va afficher leurs attributs à l'aide de méthodes. Coder le site en POO paraît donc comme le style le plus adapté à nos besoins. De plus c'est la manière de coder la plus populaire.

Un autre avantage de la POO, est l'architecture MVC, Modèle-Vue-Contrôleur. Le principe est de séparer la logique du code en 3 parties, dans 3 fichiers différents.
On a donc :

- le Modèle : qui gère les données du site, il fait la liaison avec l'éventuelle base de données.
- la Vue : qui gère la mise en page des informations.
- le Contrôleur : comme son nom l'indique, qui va contrôler les informations transmises par l'utilisateur, le Modèle et la Vue. C'est en quelque sorte l'intermédiaire entre l'utilisateur et les données du site.

À l'avenir, lorsque je parlerai d'une des parties du modèle MVC, je mettrai la première lettre en majuscule pour une meilleure compréhension.
Voici un schéma (Figure 3) qui explique comment fonctionne le modèle MVC lors d'une requête HTTP.

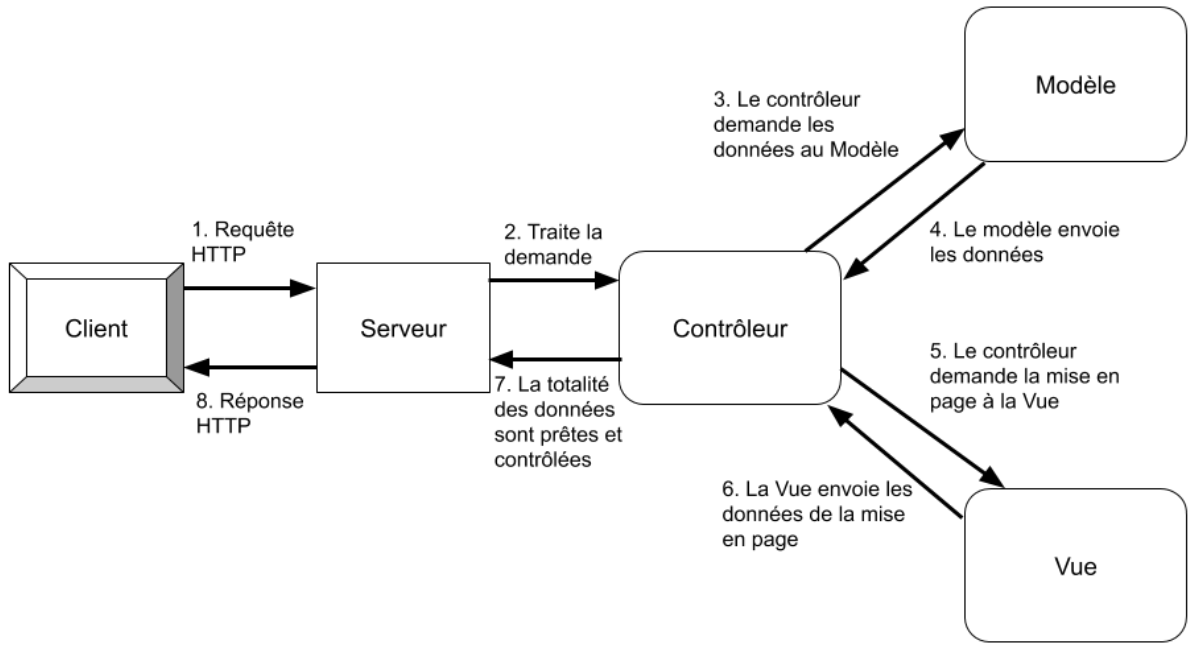


Figure 3 : Représentation de l'architecture MVC

Le modèle MVC va donc permettre d'avoir un code plus clair et plus lisible. De plus dans la réalité, ce sont les Web Designers qui s'occupent de la Vue et les Développeurs s'occupent du Modèle et du Contrôleur, le fait de partitionner le code permet de travailler à plusieurs dessus, en même temps sans qu'il y ait de problème de corruption ou de réécriture.

4.1.2 Travail réalisé

Pour expliquer la programmation en POO, je ne vais pas détailler tous les fichiers de mon projet, car cela serait trop long et inintéressant, je vais plutôt afficher l'arborescence complète (Figure 4) puis présenter rapidement quel fichier rempli quelle fonction.

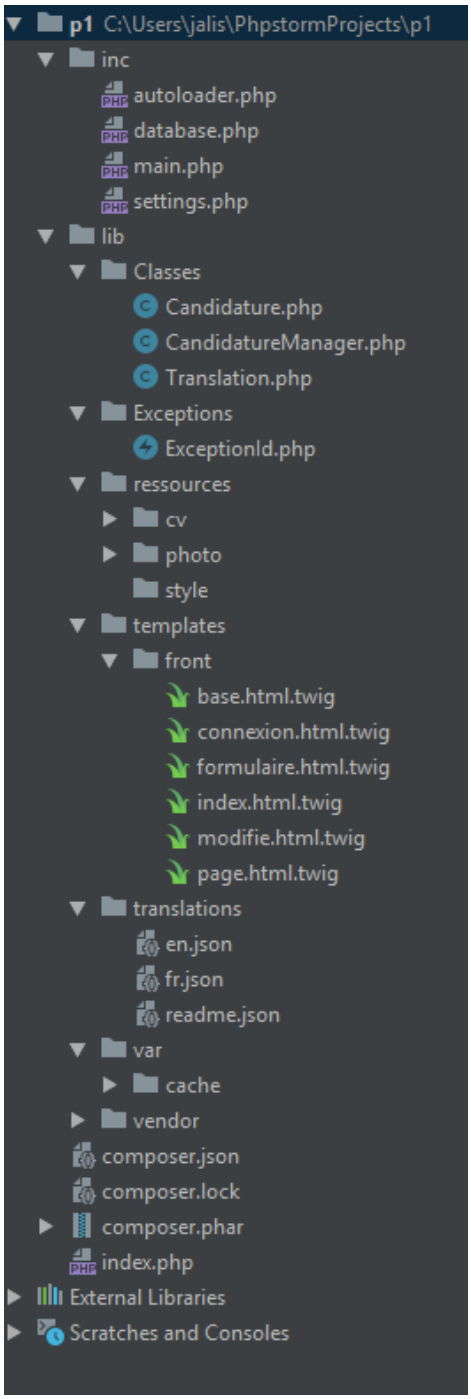


Figure 4 : Arborescence des fichiers du projet

- Dossier inc : Ce dossier possède les fichiers de configuration :
    - autoloader.php : Cette fonction permet de charger les classes utiles au fonctionnement de l'application.
    - database.php : Contiens les mots de passe, l'hôte, l'utilisateur et le nom de la base de données. Cela permet de ne pas écrire ces informations sensibles directement dans le code de la page. Lors de la connexion à la base de données, on utilise alors des alias en pointant vers ce fichier. Cela améliore la sécurité.
    - main.php : Ce fichier s'occupe de la connexion à la base de données en utilisant les informations contenues dans « database.php ».
- Dossier lib -> Classes : C'est ici qu'on regroupe nos classes :
    - Candidature.php : Cette classe définit les attributs et les méthodes de nos objets « candidature ».
    - CandidatureManager.php : Cette classe manipule les objets « candidature ».
    - Translation.php : Cette classe s'occupe de gérer les textes de manière dynamique (voir partie 4.2).
- Dossier lib -> Exceptions :
    - ExceptionId.php : lève une exception lorsqu'il est appelé dans le Contrôleur.
- Dossier lib -> ressources : Contient les C.V. et photos des candidatures ainsi que les fichiers CSS.
- Dossier lib -> templates -> front : Contient les différents gabarits des différentes pages à afficher (voir partie partie 4.2).
- Dossier lib -> translations : Contient le fichier qui rend le texte dynamique (voir partie 4.2).
- Dossier var -> cache : C'est dans ce dossier que le serveur stocke le cache du site.
- Dossier lib -> vendor : Contient les fichiers relatifs à l'installation et au fonctionnement de Composer et Twig. Composer est un gestionnaire de dépendance. Twig s'installe avec ce gestionnaire, c'est pour cela qu'il apparaît dans l'arborescence.
- index.php : C'est la seule page de notre site, et qui va servir de contrôleur. Nous allons voir maintenant qu'est-ce qu'un contrôleur ainsi que le principe de l'architecture MVC.

La première chose que j'ai faite a été de séparer le site selon l'architecture MVC, on a donc :

- le Modèle : Comme vu précédemment, la classe « Candidature » s'occupe de définir les objets « candidature » comme on le souhaite, la classe « CandidatureManager », va s'occuper des fonctions d'appels et de manipulations des données relatives aux candidatures. Par exemple sur la page d'accueil, il nous faut un tableau avec un résumé des informations de chaque candidature (id, nom, prénom et photo). Il existe donc dans ma classe « CandidatureManager » une fonction « resumeInformations() » qui demande ces informations à la base de données. Voici une capture d'écran de cette fonction (Figure 5).

```php
//Demande seulement le Nom, le prénom, la photo et l'Id à la BDD. Utilisé pour la page index.php
public function resumeInformations()
{
    return $this->bdd->query('SELECT photo, nom, prenom, id FROM candidature');
}
```

Figure 5 : Aperçu de la fonction qui affiche un résumé des candidatures.

Le Modèle fait donc le relais entre le site et la base de données, les autres fonctions servent entre autres, à ajouter une candidature, modifier une candidature et supprimer une candidature.

- la Vue : Elle sert donc à la mise en page et contient le texte de chaque page. Puisque la Vue a demandé plus de travail et met en œuvre plusieurs technologies, j'ai consacré une partie entière sur celle-ci (2.4).

- le Contrôleur : Il se trouve dans la page index.php, il a plusieurs rôles, il s'occupe par exemple d'appeler les différentes vues en fonction des besoins.
  En effet, je rappelle que dans le cahier des charges, il est spécifié que le site doit fonctionner sur une seule page : index.php. Pour arriver à afficher différentes vues sur la même page, j'ai choisi d'utiliser la fonction $_GET. Cette fonction permet de faire passer des informations via les paramètres d'URL. Voici à quoi ressemblent les URLs du site et les explications :

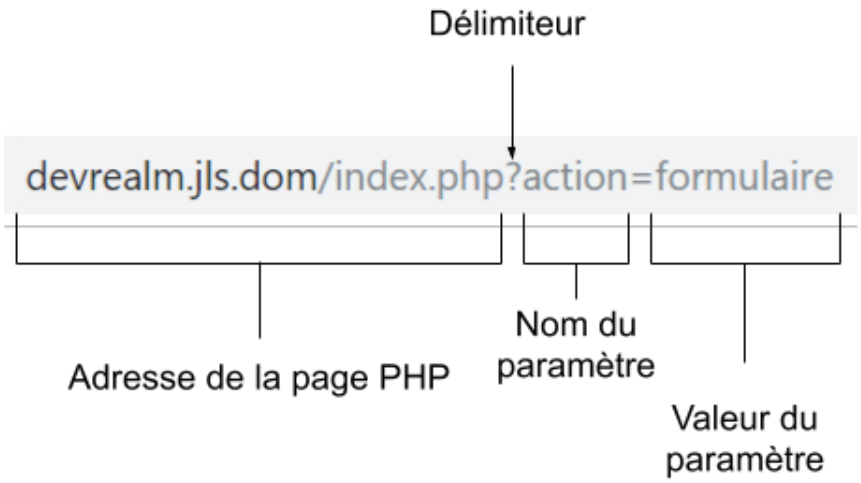


Figure 6 : URL permettant d'afficher le formulaire pour ajouter une candidature.

Donc lorsque le contrôleur reçoit cette URL, il effectue des actions spécifiques, dans la figure 6 , il va chercher la page index.php et affiche la vue correspondant au formulaire. J'ai donc programmé le contrôleur pour effectuer une action précise pour chacune des pages.
Enfin, au cas où la valeur du paramètre ou le paramètre entrés dans l'URL ne correspondent pas à ce qui est prévu, le contrôleur renvoie l'utilisateur sur la page d'accueil.

4.1.3 Problèmes rencontrés

Lors de cette partie du projet, le problème principal a été de coder tout le site en orienté objet, en effet même si cette notion a été abordée lors de ma formation à l'IUT, je n'avais pas les connaissances nécessaires pour développer entièrement un site. Il a donc fallu que je me forme, et pour cela mon maître de stage m'a conseillé de suivre le cours « Programmez en orienté objet en PHP » d'OpenClassrooms. Ce cours m'a permis de mieux comprendre les principes et les avantages du style orienté objet, et des travaux pratiques m'ont permis de mettre en application ce qui était vu dans le cours.

L'architecture MVC était une découverte pour moi, mais comme le découpage des fichiers est fait de manière logique, comprendre ce principe n'a pas été difficile.

4.2 Séparation des langages, et conception du texte dynamique

4.2.1 Présentation

Cette partie est dans la continuité de la partie précédente (2.3), mais j'ai préféré la mettre à part puisque cette partie est importante et relativement longue. Ici nous allons voir comment rendre le site dynamique et une nouvelle technologie : les moteurs de templates. Tout cela est lié à la Vue (de l'architecture MVC), de la partie précédente.

Le fait d'avoir fractionné le code en plusieurs fichiers, nous permet de satisfaire un autre point du cahier des charges : le site doit séparer le code PHP du code HTML. Pour cela, nous avons le choix d'utiliser ce qu'on appelle des moteurs de templates. Un template est un modèle, un gabarit qui gère uniquement la mise en page. PHP propose son propre template, mais la problématique est que les Web Designers ne sont pas forcément familiers avec PHP. Ici, afin d'éviter les anglicismes je vais parler de gabarit à la place de template. Il existe donc d'autres moteurs de gabarit, utilisant d'autres langages voire leur propre langage.
Lors de mon projet, j'ai choisi d'utiliser le moteur de gabarit Twig, qui utilise la technologie Symfony. Il est léger, propose une syntaxe simple à comprendre et concise, une documentation complète, et c'est le moteur de gabarit utilisé par l'entreprise où j'effectue mon stage.

Pour ce qui est de rendre le site dynamique, il existe de nombreux avantages : le code est beaucoup plus clair et concis, on peut mettre en place un système de traduction dans une autre langue, si du même contenu se trouve à plusieurs endroits du site il n'y a pas besoin de réécrire. Le texte est aussi plus simple à ajouter, modifier et supprimer.

4.2.2 Description du travail réalisé

La mise en place de cette partie du projet s'est faite en 2 temps. Premièrement, j'ai installé et configuré mes fichiers avec le moteur de gabarit Twig, puis j'ai changé tout le texte statique du site pour le rendre dynamique.

Comme expliqué dans la sous-partie précédente, un moteur de gabarit gère des gabarits pour la mise en page. Une fois le code compilé par le moteur de gabarit, le résultat est entièrement écrit en HTML. De plus, Twig comme d'autres moteurs de gabarit, propose une solution pour ne pas avoir à réécrire du code inutilement. Il est possible de créer un gabarit parent et des gabarits enfants, ces derniers héritent du gabarit parent spécifié. Dans un gabarit parent, on va mettre tout le code qui est général au site et qui ne changera pas, par exemple le langage par défaut de la page, le chemin du CSS, le chemin du JavaScript, etc. Le gabarit parent contient aussi ce qu'on appelle des "block", ce sont des balises qui indiquent au moteur de gabarit qu'un gabarit enfant peut réécrire cette partie-là. Dans mon cas, puisqu'aucune page n'est la même, j'instancie les balises "block" dans mon fichier parent et je les laisse vides. Ce sont les templates enfants qui auront la tâche de remplir ces balises.

Puisque dans la suite de cette partie je vais donner des exemples concrets, voici un petit résumé de la syntaxe de Twig. Pour plus d'informations, je vous conseille d'aller voir la documentation de Twig, le lien est dans la sitographie.

- Les variables sont entourées de 2 accolades.
  La variable « exemple » s'écrit : {{ exemple }}.

- Les fonctions sont entourées d'une accolade et d'un signe pourcent de chaque côté et est fermée par l'expression « end + nom de la fonction ».
  La fonction « if » s'écrit : {% if … %} … {% endif %}

- Les blocks sont entourés d'une accolade et d'un signe pourcent de chaque côté et est fermée par l'expression « endblock ». Le mot « block » est suivi d'une chaîne de caractère unique qui permet de différencier les différents blocks.
  Le block « contenu » s'écrit : {% block contenu %} … {% endblock %}

Voici donc un exemple concret de mes pages Twig pour la page index.php, on a le template parent « base.html.twig » et le template enfant « index.html.twig" ». Notez que le lien de parenté entre les fichiers est spécifié dans la première ligne du fichier enfant, commençant par « extends » suivi du nom du template parent.

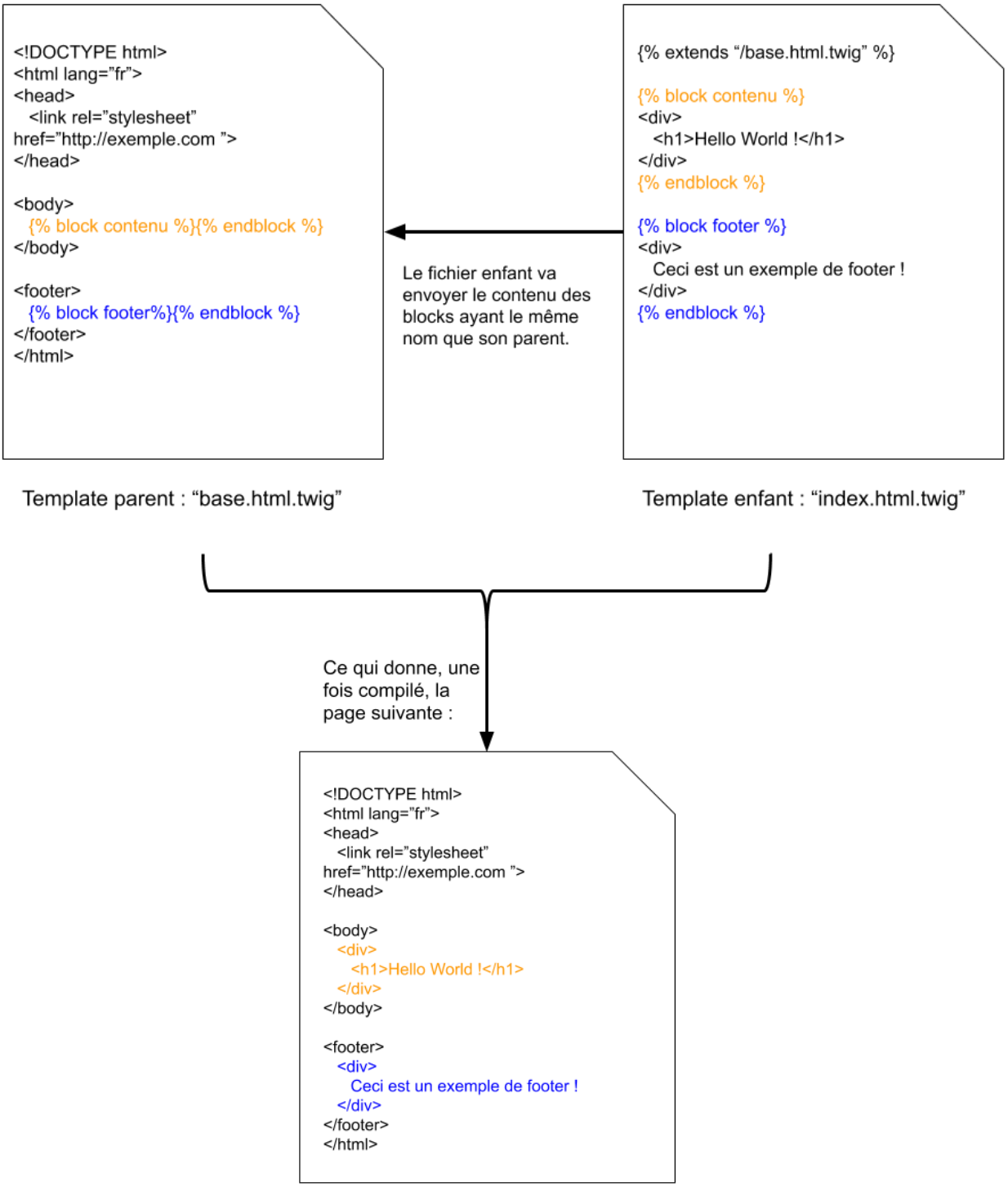


Figure 7 : Principe de parenté de template.

Maintenant, nous allons voir comment rendre tout le texte du site dynamique. Cela veut dire qu'entre les balises des fichiers Twig, il n'y aura plus de texte statique. Pour cela, il faut manipuler des variables globales. À l'inverse des variables classiques qui ne sont accessibles que dans le script où elles sont définies, les variables globales sont utilisables dans n'importe quel fichier. PHP permet d'utiliser des variables globales, mais puisque l'on utilise maintenant un moteur de gabarit et que le texte se trouve dans les fichiers Twig, nous allons utiliser la fonction de Twig : « addGlobal() ».

Le principe est assez simple : nous allons créer un fichier qui associe des alias à chacun des textes du site.
Par exemple le bouton pour ajouter une candidature a pour valeur « Ajouter une candidature », je décide d'utiliser un alias qui le définit, mon alias sera donc « bouton_ajout_candidature ». On aura donc :

Figure 8 : Exemple d'alias et de texte correspondant à celui-ci

Dans ce cas-là, cela ne semble pas très utile puisque l'alias est aussi long que le texte voulu. Mais dans le cas d'un article de plusieurs dizaines de lignes par exemple, le code sera bien plus clair et concis.
Ensuite, pour que le code aille chercher la valeur de cet alias, nous allons définir une variable globale et une fonction qui va récupérer l'alias. Lorsque la variable et la fonction sont appelées, on va chercher dans le fichier le texte correspondant à l'alias.
Dans notre cas, la variable globale s'appelle « translate » (j'explique ce choix dans la sous-partie suivante) et la fonction qui récupère l'alias s'appelle « get ».
Si l'on ajoute à cela la syntaxe de Twig, vue précédemment, on obtient au final :

```
<button>{{ translate.get('bouton_ajout_candidature') }}</button>
```

Figure 9 : Fonction appelant le texte par le biais de son alias, dans le fichier Twig

Voici un schéma récapitulatif de comment tout va fonctionner :

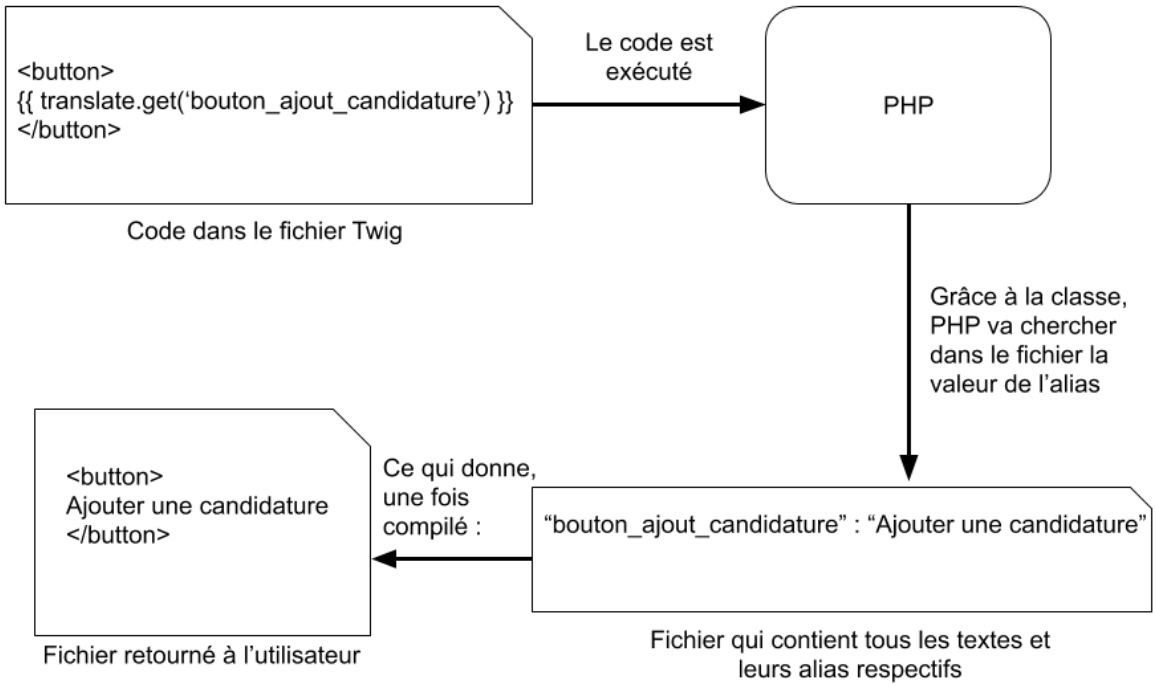


Figure 10 : Schéma de fonctionnement du texte dynamique

Il n'y a donc plus aucun texte écrit en statique. Nous avons bien satisfait le point du cahier des charges spécifiant que tous les textes doivent être dynamiques.

4.2.3 Problèmes rencontrés

Lors de la conception de cette partie du projet, j'ai rencontré plusieurs problèmes, le premier étant que je ne connaissais rien aux moteurs de gabarit. J'en avais déjà entendu parler lorsque je me renseignais sur le métier de développeur web, mais je ne connaissais ni le principe ni le fonctionnement. Il a donc fallu que je me renseigne, et notamment lire la documentation. Là vient le deuxième problème qui est que la documentation officielle de Twig n'existe qu'en anglais. Ma compréhension écrite en anglais est correcte et les explications de mes collègues et mon maître de stage m'ont permis de faciliter la lecture de la documentation.

Enfin, Twig utilise sa propre syntaxe, heureusement, elle est simple à comprendre. De plus la syntaxe est inspirée de Symfony, un des frameworks PHP les plus connus. Un framework est un ensemble de composants qui permettent de créer, dans notre cas, les fondations des sites web. Donc si à l'avenir j'ai besoin d'utiliser Symfony, j'aurai déjà quelques bases.
Pour ce qui est du passage des textes en dynamique, je ne savais pas comment cela fonctionnait non plus, et bien que sur mon site cela ne soit pas très utile car il y a assez peu de texte, cette connaissance est indispensable pour la création de gros sites ou de sites gérant plusieurs langues.

4.2.4 Pour aller plus loin

Comme dit dans l'introduction et dans la sous-partie précédente, la technique de textes dynamiques permet de mettre en place un système de traduction. C'est pour cela que ma variable globale s'appelle « translate », qui veut dire « traduire » en français. En effet, il est possible d'ajouter deux icônes au site, par exemple un drapeau français et un drapeau anglais et lorsque l'on clique sur l'un d'eux la langue du site change. Pour cela, il faut juste rajouter une condition dans la classe gérant le texte, et créer un deuxième fichier comportant exactement les mêmes alias que ce qu'on a fait précédemment et traduire simplement le texte correspondant.

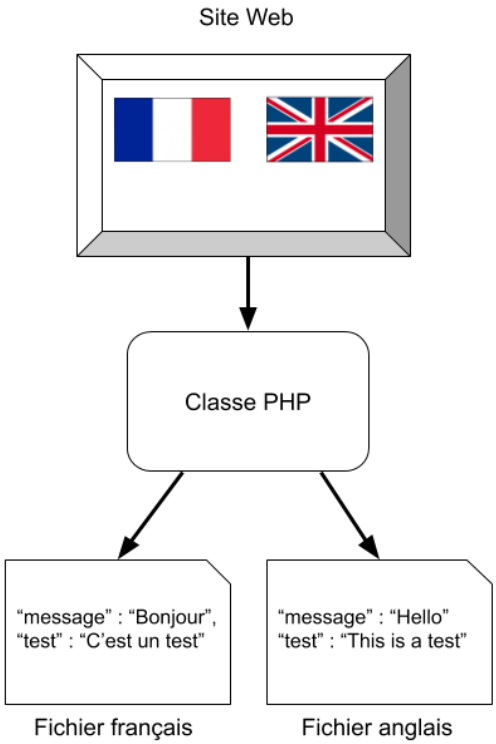


Figure 11 : Schéma du principe de traduction

4.3 Communication avec la base de données

4.3.1 Comparaison des méthodes disponibles

Pour assurer la communication entre le site et la base de données, PHP propose plusieurs interfaces de programmation : MySQL, PDO, et MySQLi. On peut directement oublier l'interface MySQL, puisqu'elle est obsolète depuis la version 5 de PHP alors que j'utilise la version 7 de PHP.
Je vais vous présenter un comparatif des points importants des deux dernières technologies, et j'expliquerai par la suite ce qui m'a orienté dans la prise de décision. Pour plus d'informations, je mets le tableau comparatif complet officiel de PHP dans les annexes.

MySQLi :

- Fonctionne en style procédural et orienté objet
- Fonctionne seulement avec les bases de données MySQL
- Technologie la plus performante

PDO :

- Fonctionne seulement en style orienté objet
- Fonctionne avec tout type de base de données
- Technologie la plus simple
- On peut nommer les paramètres, ce qui permet d'avoir un code plus clair
- Technologie la plus utilisée

Au final, j'ai choisi d'utiliser PDO pour faire mon projet, en effet, puisque le site que j'ai développé devait être en Orienté Objet, le fait qu'il ne supporte pas le style procédural n'est pas un problème. Le point majeur, est le fait que PDO est l'interface de programmation la plus répandue et la plus utilisée, notamment au sein de Jalis, donc c'est le plus pratique si j'ai des questions concernant cette technologie et puisqu'une partie de mon projet a pour but d'être utilisé par les développeurs (Partie 5), il fallait obligatoirement que j'utilise la même interface que tout le monde. Pour compléter, le fait d'avoir un code plus clair en PDO est un avantage non négociable lorsqu'on travaille en équipe. Enfin, bien que l'interface MySQLi soit plus performante que PDO, cette différence ne se ressent que sur de gros projets avec des bases de données ayant des centaines d'entrées, ce qui n'est clairement pas le cas pour mon projet.

4.3.2 Description du travail réalisé

Dès le début du stage, un accès à une base de données MySQL m'a été fourni, et donc la connexion entre ma page PHP et la base de données a été l'une des premières choses que j'ai faites. Comme nous avons vu dans la sous-partie précédente, j'ai décidé d'utiliser l'interface de programmation PDO afin de réaliser la connexion. Bien que la création de sites web communicants avec une base de données a déjà été vue à l'IUT, je n'avais jamais utilisé la technologie PDO, je me suis donc basé sur la documentation. La documentation est trouvable très simplement sur le site officiel de PHP, en français, détaillée, et avec des exemples. Cela m'a beaucoup aidé dans la découverte et la maîtrise de cette technologie.
Le site que j'ai développé utilise la communication avec la base de données sur toutes les pages. La connexion à la base de données est instanciée dans le fichier main.php, les requêtes SQL sont, quant à elles dans le Modèle, qui s'occupe de la connexion avec la base de données.

4.4 Les permissions

4.4.1 Présentation

Maintenant que le site est créé, il reste un dernier point important : la sécurité. Et notamment les permissions attribuées aux personnes naviguant sur le site. Puisque le site est dédié à la gestion et à l'administration de candidatures, il faut s'assurer que les personnes le consultant ont le droit. D'autant plus qu'il est possible d'effectuer des actions d'administration. Ici, quand je parle d'actions d'administration, je parle d'ajouter, modifier et supprimer des candidatures.

Il existe donc plusieurs solutions : soit le site est interne à une entreprise et est seulement visible depuis le réseau et donc ce n'est pas nécessaire d'ajouter des conditions de sécurité. Soit le site est visible par tous et à ce moment-là il faut identifier les personnes pouvant administrer la base de données. Dans le cadre de mon projet, il est intéressant de s'occuper de la partie sécurité du site donc nous allons partir sur le second scénario.
Ici, deux choix s'offrent à nous :

- Rendre la connexion obligatoire pour accéder à n'importe quelle page du site.
- Rendre le site accessible à tous et n'autoriser les actions d'administration qu'aux personnes connectées.

Dans le contexte d'une entreprise, différentes personnes peuvent vouloir avoir accès aux candidatures sans forcément effectuer d'actions d'administration. De plus, la première solution exige de créer des logins et mots de passe pour n'importe quelle personne voulant juste accéder au site. Cela rend donc la tâche de l'administrateur du site plus complexe inutilement.
Finalement, je choisis donc d'utiliser la seconde solution, qui semble la plus adaptée à nos besoins.

4.4.2 Description du travail réalisé

La première chose à faire a été de créer une nouvelle table dans la base de données, contenant les id, login et mots de passe des comptes pouvant effectuer les actions d'administration.
Après avoir choisi la solution la plus adaptée, vient maintenant la question de comment s'y prendre. Pour cela, dans le contrôleur, nous allons simplement mettre en place une boucle « if » qui déclare une variable globale.
- Si l'utilisateur est connecté, la variable aura pour valeur son id et son login.
- Si l'utilisateur n'est pas connecté, la variable sera nulle.

Puisque nous avons créé une variable globale, elle est accessible depuis n'importe quel fichier. Notamment les fichiers Twig. Je rappelle que toute la mise en page se trouve dans les fichiers Twig, de ce fait je peux, à l'aide d'une boucle « if », choisir un affichage différent en fonction de si le bouton est cliquable ou non. Je choisis donc que si l'utilisateur n'est pas connecté, les boutons pour ajouter, modifier et supprimer des candidatures soient désactivés. Et à l'inverse, si l'utilisateur est bien connecté, il aura accès aux actions d'administration.
Cependant, ce système ne garantit pas la sécurité, en effet, désactiver les boutons permet qu'ils ne soient pas cliquables, mais si l'utilisateur modifie le code source depuis son navigateur et supprime le mot clé « disabled », il aura alors accès aux pages qui ne lui sont, normalement, pas autorisées. Donc comme nous avons vu dans la partie 4.1, ce qui va permettre de contrôler les données reçues et envoyées par l'utilisateur, est le Contrôleur de notre architecture MVC. Pour ce faire, je mets en place, sur les pages d'administration, une boucle qui va autoriser l'affichage seulement si la variable globale a pour valeur un id et un login existant.

On notera que l'id est configuré dans la base de données en auto incrémentation, ce qui veut dire que chaque fois qu'une personne est ajoutée à la base de données, son id sera automatiquement calculé et sera supérieur de 1 à l'id de la dernière personne enregistrée. De ce fait, chaque utilisateur a un id unique, ce qui permet de mieux différencier les utilisateurs, même s'ils sont plusieurs à avoir le même login.
Je mets en annexes le comparatif de mise en page pour une personne connectée et une personne déconnectée.

On a donc assuré que les actions d'administration ne seront effectuées que par les personnes ayant la permission.

Cette partie n'a pas posé de problème particulier lors de sa conception.

4.5 Conclusion sur le projet

Ce projet a été réalisé durant la majeure partie du stage. Tous les points du cahier des charges ont été satisfaits, et même plus puisque la partie concernant les permissions ne figurait pas dans le cahier des charges. Une page de connexion devait être créée, mais son usage n'était pas spécifié. J'ai choisi de faire cette tâche, car je pense que manipuler est la meilleure façon de progresser. Et puisque je me trouvais dans un environnement où mes collègues pouvaient me donner des conseils et m'aider lorsque j'étais bloqué, j'ai décidé de faire cette partie supplémentaire, essentielle lors du déploiement d'un site.

En résumé, ce projet m'a permis de découvrir de nombreuses technologies comme les moteurs de gabarits et l'architecture MVC. Et comme j'ai codé tout le site moi-même, sans m'aider de framework, lorsqu'un problème survenait, je savais où et comment le résoudre.

5. Améliorations des outils de développement

5.1 Présentation

Une fois le projet terminé, une nouvelle tâche m'a été assignée, en effet, les développeurs de Jalis utilisent des frameworks, qu'ils ont développés afin de créer les sites web de clients. Ces sites proposent donc une vue classique, pour tout le monde et une vue pour l'administrateur du site. On appelle le Frontend la partie que tous les utilisateurs voient lorsqu'ils arrivent sur le site, et Backend la partie « cachée » du site qui sert à gérer les articles, les sections, etc., et qui est visible seulement par les administrateurs du site. Ici, le problème venait du backend, en effet, lorsqu'un utilisateur télécharge une image vers le serveur, il n'a pas moyen de la faire pivoter, ce qui peut poser problème.
Mon travail consiste donc à créer une fonction permettant de faire pivoter des images et ensuite d'intégrer cette fonction au framework.

5.2 Description du travail réalisé

Pour cette partie, j'ai d'abord créé de mon côté une page permettant de mettre plusieurs images dans un tableau, puis pour rendre l'aspect plus intuitif, j'ai décidé d'utiliser des images cliquables afin de faire pivoter les images. Ce serait simplement une flèche vers la droite et une flèche vers la gauche. Tout comme dans la partie précédente, le site sera fait en POO, afin d'exporter la fonction de rotation le plus simplement possible. Mon application comporte donc une seule page : « index.php » et une classe qui gère les actions apportées sur les images : « ImageManager.php ».
Lorsqu'un client ajoute une image sur son site, l'image est copiée dans 4 dossiers différents. Chaque dossier enregistre l'image avec des dimensions différentes. Il y a : big, medium, small et thumb. Lorsque le client fait pivoter son image, il faut que les 4 images soient pivotées en même temps.

Pour le fonctionnement du code, on va distinguer 2 cas : le cas où l'image téléchargée est un fichier PNG et le cas où c'est un fichier JPG (ou JPEG).
Le fonctionnement est le suivant : lorsque l'on appuie sur le bouton de rotation, on va créer une nouvelle image, identique à celle de base, puis on va appliquer la rotation, et enfin on enregistre la nouvelle image à la place de l'ancienne. Le fonctionnement que je viens de décrire est quasiment identique pour les images PNG et JPG. La seule différence est que, puisque les images PNG peuvent avoir une partie transparente, il faut rajouter une ligne de code pour que la partie transparente ne soit pas dégradée.

Une fois la fonction développée, il a fallu l'intégrer au framework. Pour cela j'ai utilisé Git, Git est un logiciel de gestion de versions décentralisé. Cela signifie que pour un gros projet, où plusieurs personnes travaillent en même temps, et plus particulièrement des personnes de services différents (Développeurs, Designers, Graphistes ...). Le principe de ce logiciel est que le projet complet est sur un serveur, chaque personne peut copier une partie du projet sur son ordinateur afin d'apporter des modifications. Lorsque tout est changé et fonctionnel, on peut copier le code modifié vers le serveur. On dit que Git est un gestionnaire de version car chaque modification apportée au projet est enregistrée. Cela permet de faire un backup en cas de problème. C'est un logiciel très complet et très populaire, je mets dans la sitographie le site officiel présentant les principes.

5.3 Conclusion sur le projet

Ce second projet a été réalisé durant la fin du stage. Le cahier des charges demandait de créer une fonction opérationnelle pour l'intégrer aux outils de développement, c'est ce qui a été fait. Ce projet m'a vraiment permis de comprendre les enjeux du travail en équipe, savoir commenter correctement un code, s'adapter au code déjà existant, utiliser un logiciel de partage de fichiers…

6. Conclusion

En conclusion, ce stage m'a permis de découvrir beaucoup d'aspects du métier de développeur web. En effet, mon projet personnel et professionnel est de devenir développeur web, ce stage m'a permis de découvrir également le travail en entreprise.

Au niveau technique, lors de ce stage j'ai découvert de nombreuses technologies et logiciels, et ma formation s'est fait en grande partie par la lecture de la documentation et la manipulation. Le fait de tester avec un serveur et une base de données dédiés m'a permis de réellement comprendre comment fonctionnent les technologies. Ces connaissances me seront très utiles puisque la totalité des technologies et techniques utilisées lors du stage sont les mêmes que celles qu'utilisent les développeurs de Jalis.

Si on fait la synthèse des deux projets que j'ai réalisés, j'ai acquis beaucoup de connaissances nouvelles, dans toutes les facettes du développement web, de la création complète d'un site web, à la création d'outils pour un framework commun à tous les employés.

Ces connaissances me seront très utiles pour mon projet professionnel. Ce stage m'a permis de mieux cerner en quoi consiste le métier de développeur web. Et ce stage a confirmé mon envie de faire le métier de développeur web.

7. Remerciements

Tout d’abord, j’adresse mes remerciements à Mathieu LAB, qui m’a aidé à trouver ce stage.

Je tiens à remercier mon maître de stage, Jean-Christophe DUVIVIER pour son accueil, le temps qu’il m’a accordé et son aide.

Je remercie Serge ALAGY, le directeur général, de m’avoir accueilli au sein de son entreprise.

Je remercie également toute l’équipe de développement pour leur accueil, et leurs conseils.

8. Sitographie

Bootstrap [En ligne]. Consulté le 16 avril. Disponible sur : https://getbootstrap.com/docs/4.3/getting-started/introduction/

Git [En ligne]. Consulté le 4 juin. Disponible sur : https://git-scm.com/

MySQL [En ligne]. Consulté tout le long du stage. Disponible sur : https://dev.mysql.com/doc/

OpenClassrooms (cours) [En ligne]. Consulté du 30 avril au 13 mai. Disponible sur : https://openclassrooms.com/fr/courses/1665806-programmez-en-oriente-objet-en-php

OpenClassrooms (forum) [En ligne]. Consulté tout le long du stage. Disponible sur : https://openclassrooms.com/forum/categorie/php

Php [En ligne]. Consulté tout le long du stage. Disponible sur : https://www.php.net/manual/fr/

Stack Overflow [En ligne]. Consulté tout le long du stage. Disponible sur : https://stackoverflow.com/questions

Twig [En ligne]. Consulté du 16 avril au 29 avril. Disponible sur : https://twig.symfony.com/

W3C [En ligne]. Consulté tout le long du stage. Disponible sur : https://www.w3schools.com/php/default.asp

Institut Universitaire de Technologie,
Aix-Marseille Université

ANNEXES
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Conception et réalisation d'un site web

Damien MATHIEU

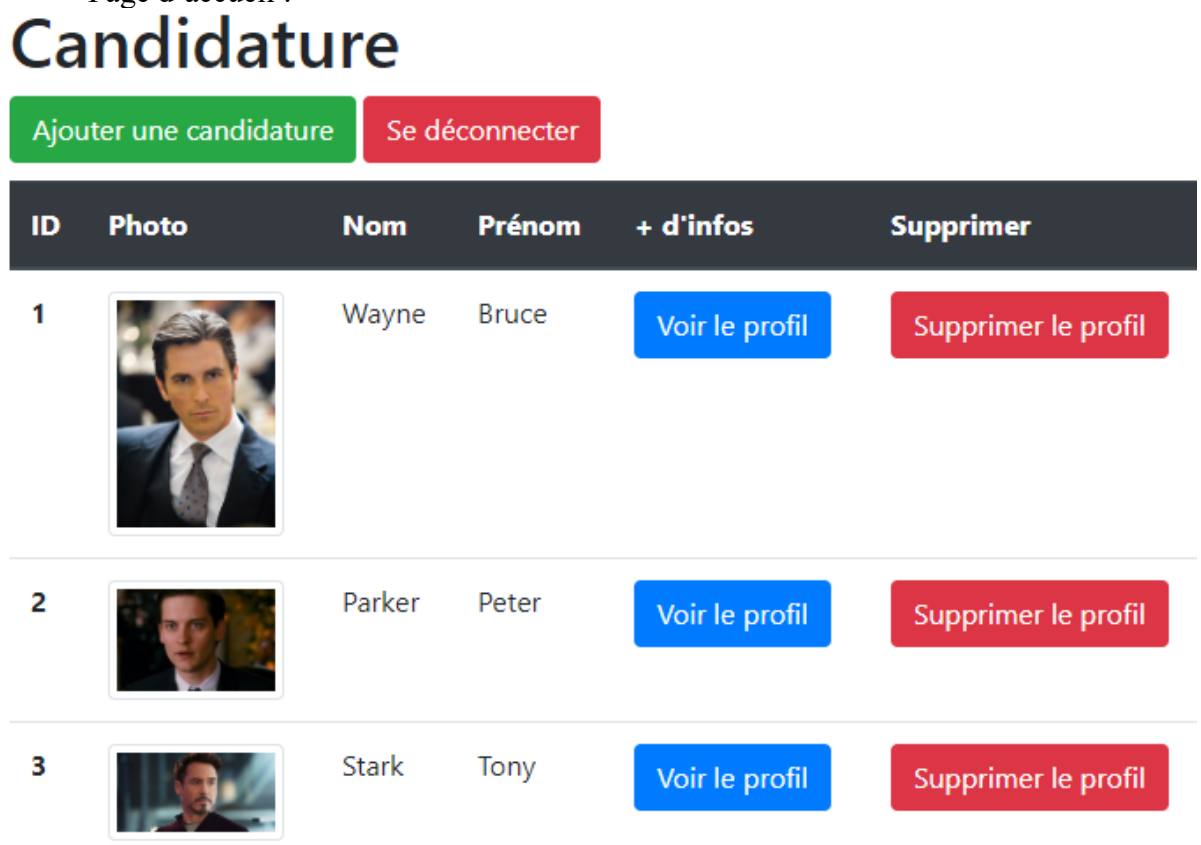
Jalis

Responsable entreprise : Jean-Christophe DUVIVIER
Responsable académique : Eric WÜRBEL

2019

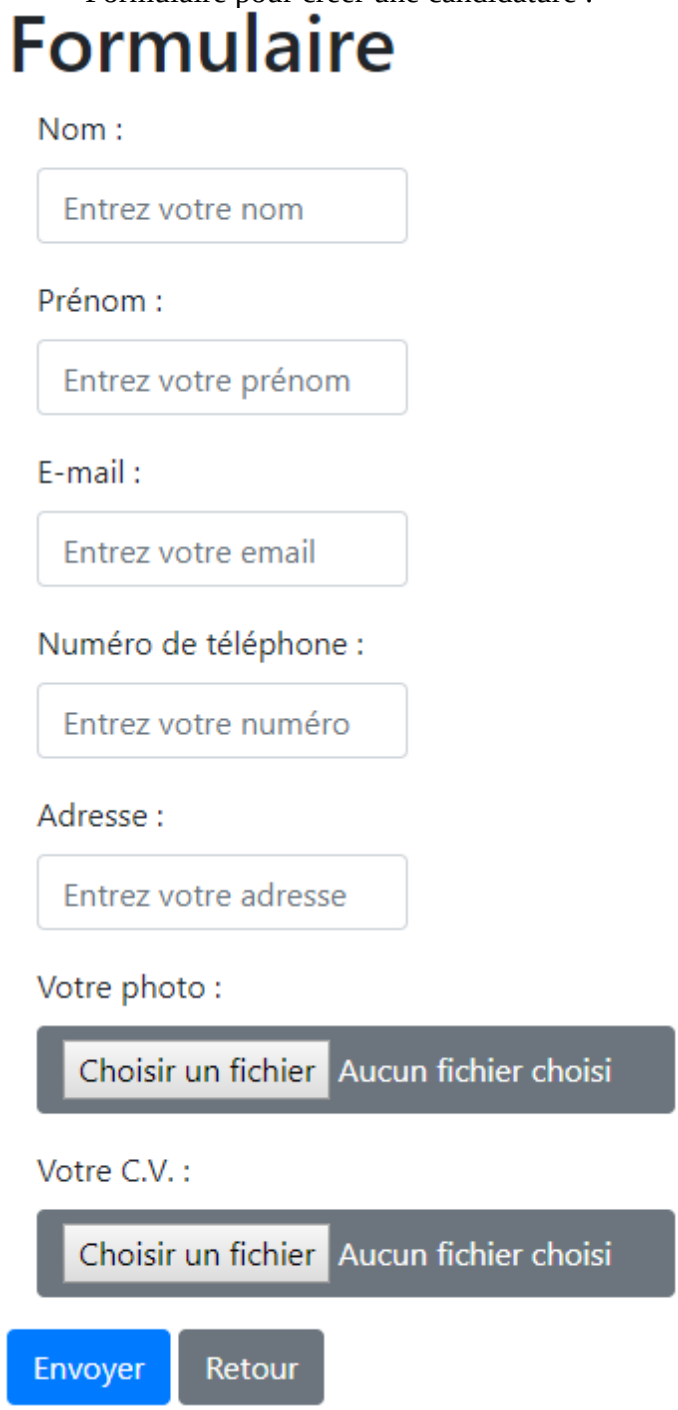
Voici un aperçu des pages du site réalisé :

- Page d'accueil :



- Page contenant les informations d'une candidature :

- Formulaire pour créer une candidature :



- Page de connexion :

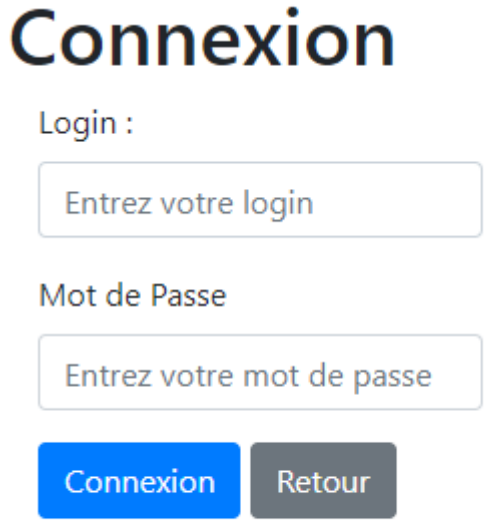
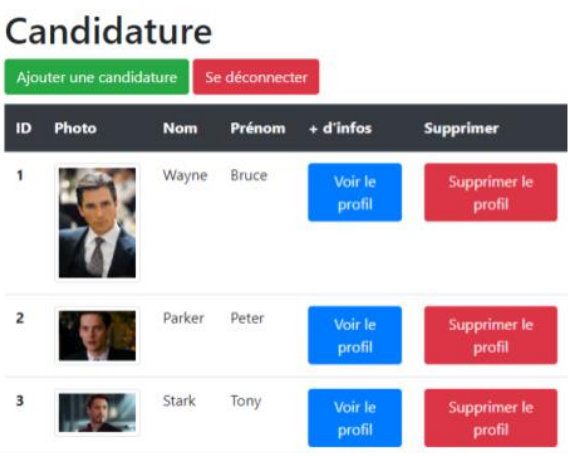


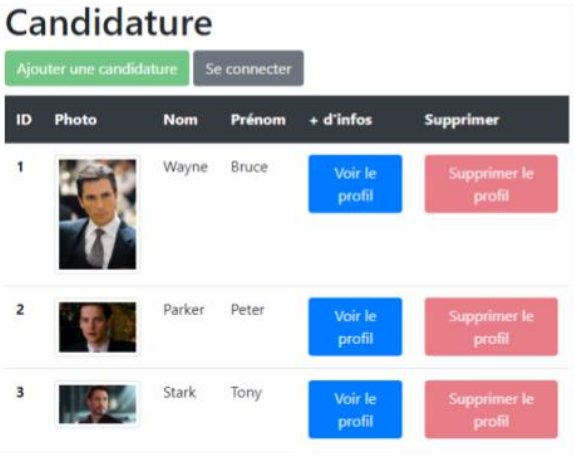
Tableau comparatif officiel des différentes interfaces de programmation de PHP :

	ext/mysqli	PDO_MySQL	ext/mysql
Introduite en PHP version	5.0	5.1	2.0
Inclus avec PHP 5.x	Oui	Oui	Oui
Inclus avec PHP 7.x	Yes	Yes	No
Statut du développement	Active	Active	Uniquement en maintenance en 5.x ; supprimée en 7.x
Cycle de vie	Active	Active	Obsolète en 5.x ; supprimée en 7.x
Recommandé pour de nouveaux projets	Oui	Oui	Non
Interface orientée objet	Oui	Oui	Non
Interface procédurale	Oui	Non	Oui
L'API supporte les requêtes non-bloquantes, asynchrones avec mysqlnd	Oui	Non	Non
Connexions persistentes disponibles	Oui	Oui	Oui
L'API supporte les jeux de caractères	Oui	Oui	Oui
L'API supporte les requêtes préparées côté serveur	Oui	Oui	Non
L'API supporte les requêtes préparées côté client	Non	Oui	Non
L'API supporte les procédures stockées	Oui	Oui	Non
L'API supporte les requêtes multiples	Oui	La plupart	Non
L'API supporte les transactions	Oui	Oui	Non
Les transactions peuvent être contrôlées avec SQL	Oui	Oui	Oui
Supporte toutes les fonctionnalités de MySQL 5.1+	Oui	La plupart	Non

Comparatif de la mise en page de la page d'accueil pour une personne connectée et une personne déconnectée :



Aperçu de la page lorsque l'utilisateur est connecté



Aperçu de la page lorsque l'utilisateur est déconnecté