# Rapport projet PPE

## Sommaire

# Contexte et spécifications du projet

## Contexte

Le système d'information étudié  lors du projet est issu de l'entreprise Galaxy Swiss Bourdin (GSB). Le laboratoire Galaxy Swiss Bourdin est issu de la fusion entre le géant américain Galaxy (spécialisé dans le secteur des maladies virales dont le SIDA et les hépatites) et le conglomérat européen Swiss Bourdin (travaillant sur des médicaments plus conventionnels), lui-même déjà l'union de trois petits laboratoires.

## Spécifications

Le contexte du projet consistait en la réalisation et la mise en place d'un logiciel de helpdesk (dit de gestion d'incidents), dû aux fortes demandes d'assistances de la part du personnel. Le logiciel à développer devait répondre à deux problématiques principales :

- Fournir de l'assistance aux utilisateurs
- Permettre aux utilisateurs de suivre en temps réel la résolution de leurs problèmes

Pour ce faire une étude préalable a été réalisée pour s'inspirer d'applications déjà existantes, ainsi une analyse de l'application GLPI, un projet open-source, nous a permis d'établir les principales fonctionnalités de l'application à concevoir.

Toutefois dans le cadre de notre projet, nous étions amenés à ne développer que la partie utilisateur de l'application. Ce faisant notre projet devait contenir les caractéristiques suivantes :

- Obligation de s'identifier en tant qu'utilisateur
- Possibilité pour les utilisateurs de changer de mot de passe
- Nécessité d'être connecter pour déposer un ticket
- Possibilité  pour un utilisateur de déposer un ticket
- Permettre aux utilisateurs de suivre l'évolution d'un ticket
- Affichage spécifique pour les appareils mobile de type Smartphone et tablette.

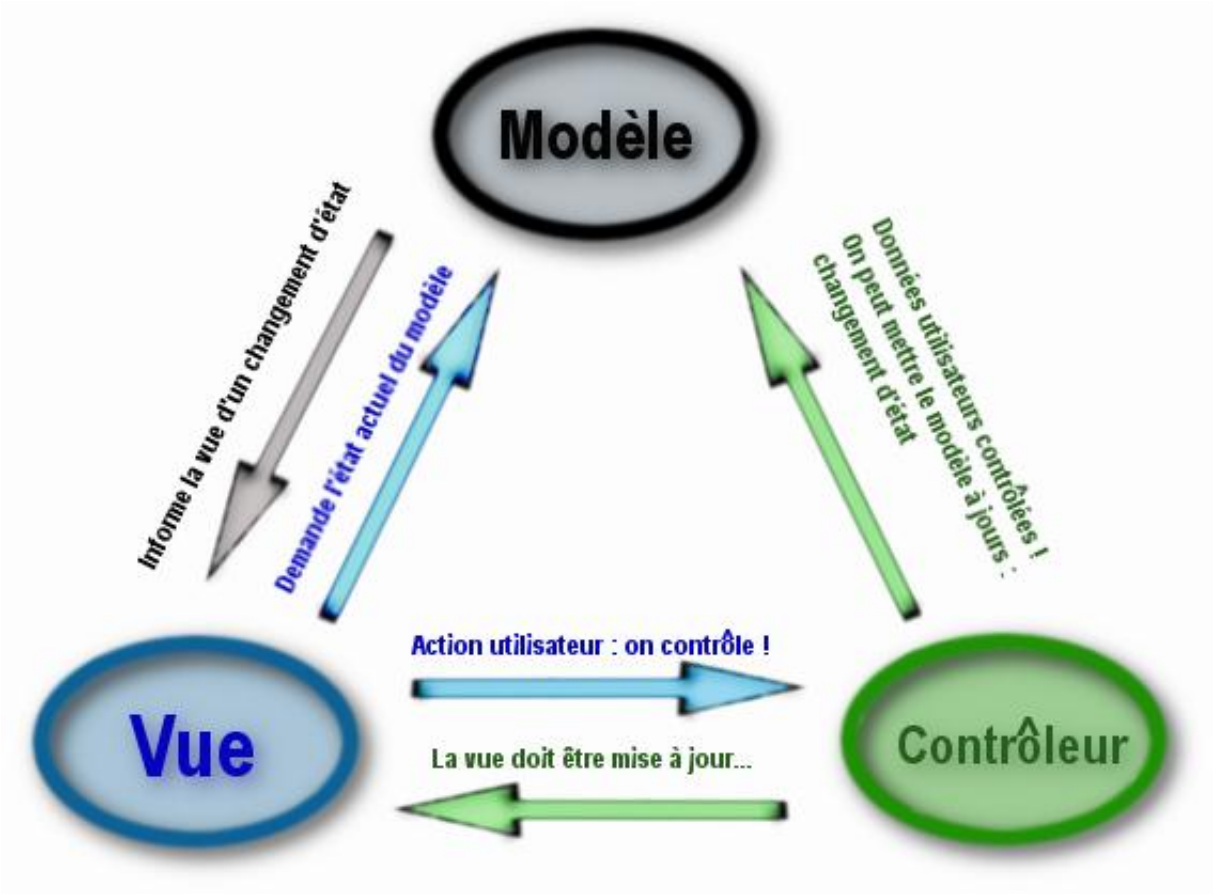
# Choix des solutions

Pour mener à bien ce projet plusieurs choix se sont offerts à nous.

Il nous était possible de réaliser l'application web demandée en utilisant un framework pour PHP comme Zend ou Symphony. Cependant étant donné le temps de développement imparti, l'utilisation de framework aussi puissant aurait constitué un temps d'apprentissage trop important.

## Modèle MVC

Nous nous sommes donc tournés vers la réalisation de l'application en PHP natif tout en utilisant l'architecture MVC (Modèle Vue Contrôleur). Cette architecture qui nous était alors encore peu connu, nous a permis de découvrir de nouvelles méthodes de développement. Il nous a donc été nécessaire de réaliser une veille technologique vis-à-vis de cette architecture particulière. Le modèle MVC peut être modélisé par le schéma suivant :




Le traitement d'une demande d'un utilisateur se déroule selon les étapes suivantes :

- L'utilisateur fait une demande au contrôleur. Ce contrôleur est la porte à toutes les demandes utilisateurs. C'est le C de MVC.
- Le contrôleur va traiter cette demande, il va donc appeler la couche dites métier, le M dans la structure MVC

- Une fois que le contrôleur reçoit la réponse de la couche métier (ou modèle), la demande de l'utilisateur a été traitée. Par exemple selon le traitement une page d'erreur ou de confirmation va être affichée.
- Le contrôleur ayant choisi la réponse il va envoyer à l'utilisateur cette dernière, ce que l'on appelle également la vue. Il s'agit le plus souvent d'une page contenant des éléments dynamiques.
- Enfin la vue est envoyée à l'utilisateur, le V de MVC.

Les principales raisons pour le choix d'une architecture selon le type MVC sont :

- La clarté de l'architecture au sein de l'application.
- Une plus grande facilité pour la distribution des tâches
- L'apprentissage et la mise en place d'une nouvelle méthode de développement.

### Langages et Logiciels

Par la suite nous avons définis les langages et logiciels utilisées lors de la phase de développement de l'application. Nous nous sommes donc tournés vers l'utilisation d'un serveur Apache ainsi que de PhpMyAdmin, la mise en place d'une machine virtuelle servant de FTP a également été utilisée de sorte à ce que, lors du développement en groupe, chacun puisse récupérer les fichiers développés par les autres.

De plus l'utilisation de PDO s'est avérée comme nécessaire pour la connexion à la base de données, PDO offrant une interface pour accéder à une base de données depuis PHP intuitive. PDO offre également la possibilité d'être personnalisé, ainsi que la gestion des exceptions permettant l'intégration d'un système de gestion des erreurs.

Concernant les langages utilisés, l'application est essentiellement codée en PHP Objet. Toutefois l'affichage est réalisée en partie en HTML, et le développement de quelques fonctionnalités ont étés fait en jQuery ou Ajax, de sorte à rendre un affichage dynamique.

## Mise en place des solutions

Suite à l'étude du cahier des charges ainsi qu'à l'analyse de l'application GLPI, l'une des premières réalisations du projet fût la réalisation du modèle Entité-Association (MEA) ainsi que par la suite sa traduction en modèle relationnel pour enfin arriver à la création de la base de données.

Nous avons par la suite réalisés deux maquettes de manière à avoir une vue générale de l'application sur laquelle nous allions nous tourner. Ces deux maquettes sont fournies en tant qu'annexes, l'application ayant évolués de manière importante vis-à-vis de ces maquettes de début de programmation.

Nous avons ainsi décidés de séparer les différentes fonctionnalités de l'application par des onglets, éléments transposées en icônes pour l'affichage mobile.

Concernant ce même affichage mobile, après avoir effectué de nombreuses recherches sur les différentes méthodes de détection des appareils mobiles, nous nous sommes tournés vers une

requête PHP permettant la détection des navigateurs par le serveur. Ainsi selon s'il s'agit d'un navigateur de type Dolphin ou Safari pour mobile, par exemple, le CSS chargé est celui de type « mobile », dans le cas contraire le CSS normal est chargé. Cette solution présentait l'avantage d'une détection directe et permettait d'avoir deux fichiers CSS distinct, solution plus claire si l'un des deux CSS venait à être modifiée. L'inconvénient majeur de cette méthode consiste en la mise à jour continue de la requête si de nouveau système mobile venait à voir le jour.

Nous nous sommes donc tournés par la suite sur la programmation plus en détail de l'application. Ainsi nous avions deux grands axes à traiter, le code relatif à toutes les possibilités de traitement par l'utilisateur d'une part, ainsi que tous les éléments concernant les tickets d'une autre part.

Cette partie du projet a fortement été facilité par l'analyse de la solution GLPI, nous permettant de définir quel type d'action un utilisateur pouvait effectuer et quels éléments étaient nécessaires à la création d'un ticket. Gardant en tête que cette application devait être utilisé au sein d'une entreprise, nous avons fait le choix de ne permettre à l'utilisateur que la modification de son mot de passe, son login (consistant en son mail au sein de l'entreprise) ainsi que son nom et prénom étant pré-remplies et inchangeable. Concernant les tickets, nous avons choisis de permettre à l'utilisateur de donner un nom à ce dernier, de donner un descriptif de l'incident ainsi que le choix de la priorité et de la catégorie de l'incident, la date étant pré-remplie au jour et heure de la création du ticket.

Pour ce qui est de l'affichage des tickets ayant été créés par l'utilisateur, nous nous sommes tournés vers un tableau regroupant les informations générales de chaque ticket, à savoir son nom, sa date de création, le numéro du ticket ainsi que son statut actuel. L'utilisateur peut alors, sur la même page, décider de voir les détails d'un ticket en cliquant sur un lien. Ces mêmes détails sont alors affichés de manière dynamique sur la page, en dessous du tableau, grâce à l'Ajax. L'avantage principal de cette technique consiste à éviter à l'utilisateur de devoir retourner sur l'historique des tickets après avoir regardé un ticket en détail.

Un des ajouts que nous avons réalisés vis-à-vis du cahier des charges consiste en la présence d'un bouton permettant l'envoi d'un mail à l'utilisateur dans le cas où il aurait oublié son mot de passe. Cet élément est affiché sur l'écran de connexion, il suffit alors à l'utilisateur d'entrer son adresse mail (login) dans un champ et l'application retournera un mail contenant un lien permettant à l'utilisateur de réinitialiser son mot de passe.

Nous avons aussi développés un guide d'utilisation de manière à aiguiller l'utilisateur sur l'utilisation de l'application. Ce guide est accessible par le lien situé en pied de page. Il a été réalisé en partie avec du jQuery.

# Conclusion

Ce projet nous auras permis de mettre en application de nouvelles méthodes de développement ainsi que d'améliorer la gestion du travail en équipe. En effet, la conception d'une application sous la méthode MVC comporte des spécificités de développement allant aisément de pair avec un travail en groupe, ainsi nous avons pu mieux organiser notre travail d'équipe ainsi que la gestion du temps imparti.

Ce projet aura également été l'occasion pour nous de mettre en application de nouveaux langages ainsi que d'approfondir ceux que nous connaissions déjà. Nous avons rencontré cependant plusieurs difficultés, notamment sur la gestion des formulaires de création de tickets ou de gestion du compte de l'utilisateur.

Si nous devions être amené à faire évolution l'application, il serait possible de développer la partie administration de l'application. Dans le but de, par exemple, affecter des techniciens à des tickets, ou encore de modifier, voire supprimer, des tickets ayant été mal rédigées.

# Hébergement

L'une des conditions du projet était également l'hébergement de l'application sur un site web, l'application complète est donc disponible à cette adresse :

http://btssio.davidung.fr/

Afin de rendre l'application entièrement fonctionnelle sur l'hébergeur, une modification a dû être apportée au niveau des variables globales déclarées dans le fichier config.php du dossier global. Etant donné qu'en local, la racine retournée par la variable $_SERVER['DOCUMENT_ROOT'] correspond au dossier 'www' et non au dossier 'www/mon_projet', il y a un léger changement à ce niveau par rapport à la version hébergée.

Par exemple, sur le serveur d'hébergement, la variable globale CHEMIN_LIBS (correspondant au dossier contenant les librairies utilisées), elle a été défini de la manière suivante :
*define('CHEMIN_LIBS', $_SERVER['DOCUMENT_ROOT'].'/libs/') ;*

Concernant le changement sur la version locale, il a fallu la définir avec le nom du dossier en plus :
define('CHEMIN_LIBS', $_SERVER['DOCUMENT_ROOT'].'**/mon_projet**/libs') ;

# Annexes

En annexe au rapport sont mis à disposition :

- Le MLD et le MCD de l'application.
- L'arborescence adoptée pour la conception du site.
- Les maquettes de l'application.
- Le script d'insertion de la base de données.